Application Note Template

Ardit Pranvoku and Thai Nghiem
Rowan University

October 17, 2017

1 Abstract

Creating the Stranger Things Christmas light wall in the assignment allows for more to be discovered about the MSP430. Knowledge and background information on the UART system is needed to complete the assignment. CodeComposer is used to code the behavior of the MSP430. The board needs to be able to take an input of characters to change its own RGB values while allowing the next board to receive the leftover message.

2 Introduction

UART or Universal Asynchronous Receiver Transmitter is widely used in many technological applications. Its essential for allowing devices to communicate in the growing technological world. Networks, local and otherwise, rely heavily on communication interfaces such as UART. One prominent application where UART is used is in GPS. This goes to show how UART is still applicable in today's society, and will continue to be used for years to come.

3 Background

UART is a communication interface that transmits bytes of data bit by bit. A transmitting and receiving UART is able to to convert between the byte and bit forms of the data through the use of a shift register.Because it is cheaper and simpler to transmit bits serially through a wire rather than transfer a byte in parallel, UART becomes useful.

A UART device will automatically start high, with lows being significant. A character usually consists of a logical low start bit, data bits, and a stop bit. Sometimes a parity

bit is added between the data bits and a stop bit. This is used to make sure that the transmission always ends up with an even number of bits by adding a bit if the total is an odd number. A clock signal running 8 times faster than the bit rate governs the UART. Any signal transmitted that lasts at least half the bit rate is valid. If not, the signal is deemed to be a spurious signal, and will not be transmitted. After the character has been fully processed, which is typically for 8 bits, the UART device will make the data available to the processor in a register. A shift register is usually used. An interrupt may be generated and flag will be set informing the system that new data is ready to be received.

3.1 Equations

Formula to find the duty cycle for each color of the RGB LED. This is applied in the Timer Interrupt method.

$$dutycycle = \frac{red/green/blue PWM}{255} * 100 \tag{1}$$

Find the rate at which the UART transmit the message. Bit rate is found based on clock rate. This equation is applied in the UART initialization method.

$$bitrate = \frac{clockrate}{8} \tag{2}$$

General equation to find bit rate, which based upon the bit transmitted.

$$bitrate = \frac{bitstransmitted}{second} \tag{3}$$

3.2 Why FR5594?

The FR5994 was chosen due to its lack of limitations and its advantages in the project in compare to the other 4 boards. Firstly, it can easily implement PWM, as it has both Timer A and B, unlike the FR2311. Secondly, from the experience collected from lab 1 (using example ”Echo” code), the FR5594 can communicate over UART (using PuTTy as the user interface) much faster than the other 4 boards. Thirdly, the FR5994 belongs to the Ferroelectric Random Access Memory family, which is a nonvolatile memory that combines the speed, ultra-low-power, endurance and flexibility of SRAM with the reliability and stability of Flash. In conclusion, it had no problem achieving all goals laid out for this milestone.

4 Figures

```
..
#include <msp430.h>

int TimerCount = 0;
unsigned int byteCount = 0;
unsigned int numOfBytes = 0;
volatile unsigned int i = 0;
int redPWM, greenPWM, bluePWM;
char Message[80]; //UART Message;

void timerIntialize(void){ // Void function that intiializes the Timer A
    TA0CCTL0 = (CCIE); //Timer A capture/control Interrupt Enable
    TA0CCR0 = 0x0001; // Initialize CCRO
    TA0CTL = TASSEL_2 + MC_1 + ID_3; // SMCLK / Upmode / Divider of 8
}
void ledInitialize(void) { // Void function that intiializes the Pins to LEDs
    P1DIR |= BIT0;        // Set P1.0 to output direction
    P1OUT &= ~BIT0;       // Switch LED off

    P1DIR |= BIT1;       //set Port 1.1 output ---LED
    P1OUT &= ~BIT1;       // Swich LED off

    P1DIR |= BIT3;       // P1.3 to output
    P1DIR |= BIT4;     // P1.4 to output
    P1DIR |= BIT5;      // P1.5 to output
}
```

Figure 1: Initialization for LEDs output and Timer A.

```
    // USCI_A0 UART operation and intialization (inspired by William Goh in his/her code of
    //                                          "eUSCI_A0 UART echo at 9600 baud")
void uartInitialize(void){
       CSCTL0_H = CSKEY_H;                     // Unlock CS registers
       CSCTL1 = DCOFSEL_3 | DCORSEL;           // Set DCO to 8MHz
       CSCTL2 = SELA__VLOCLK | SELS__DCOCLK | SELM__DCOCLK;
       CSCTL3 = DIVA__1 | DIVS__1 | DIVM__1;   // Set all dividers
       CSCTL0_H = 0;                           // Lock CS registers
       UCA0CTLW0 = UCSWRST;                    // Put eUSCI in reset
       UCA0CTLW0 |= UCSSEL__SMCLK;             // CLK = SMCLK
       UCA0BRW = 52;                           // 8000000/16/9600
       UCA0MCTLW |= UCOS16 | UCBRF_1 | 0x4900;
       UCA0CTLW0 &= ~UCSWRST;                  // Initialize eUSCI
       UCA0IE |= UCRXIE;CSCTL0_H = CSKEY_H;
```

Figure 2: Initialization for UART.

```
114     case USCI_UART_UCRXIFG:
115         if (byteCount == 0)
116         {
117             P1OUT |= BIT0; //Turn on P1.0 LED
118             numOfBytes = UCA0RXBUF; // Copy total number of bytes in Rx Buffer
119             byteCount++; // Increase to the next byte
120         }
121         else if ((byteCount > 0) && (byteCount < 4))
122         {
123             switch (byteCount)
124             {
125             case 1:
126                 redPWM = UCA0RXBUF; //Assign value of the PWM control Red color
127                 break;
128             case 2:
129                 greenPWM = UCA0RXBUF; //Assign value of the PWM control Green color
130                 break;
131             case 3:
132                 bluePWM = UCA0RXBUF; //Assign value of the PWM control Blue color
133                 break;
134
135             }
136             byteCount++; // Continue to increase to the next byte
137         }
```

Figure 3: Assigning duty cycle to PWM to control the color of LED, using UART buffer.

```
138         else if ((byteCount > 3) && (byteCount < numOfBytes))
139         {
140             Message[byteCount + 1] = UCA0RXBUF; //Copy the information from Rx Buff to Message variable
141             UCA0TXBUF = UCA0RXBUF; //
142             byteCount++; // Continue to increase to the next byte
143         }
```

Figure 4: Transmitting message to the next MSP430 device.

```
__interrupt void Timer0_A0_ISR(void)
{

    if (TimerCount == 255) //Maximum value, turn all color on -> White color
    {
        P1OUT |= BIT3; //turns on BIT3 led
        P1OUT |= BIT4; //turns on BIT4 led
        P1OUT |= BIT5; //turns on BIT5 led
        TimerCount = 0;
    }
    if (TimerCount == redPWM)
    {
        P1OUT &= ~BIT3; //turns off BIT3 led
    }
    if (TimerCount == greenPWM)
    {
        P1OUT &= ~BIT4; //turns off BIT4 led
    }
    if (TimerCount == bluePWM)
    {
        P1OUT &= ~BIT5; //turns off BIT5 led
    }

    TimerCount++; //Increase timer count by 1
    TA0CCTL0 &= ~BIT0;  //clears BIT0
```

Figure 5: Change LED color using Timer A interrupt.

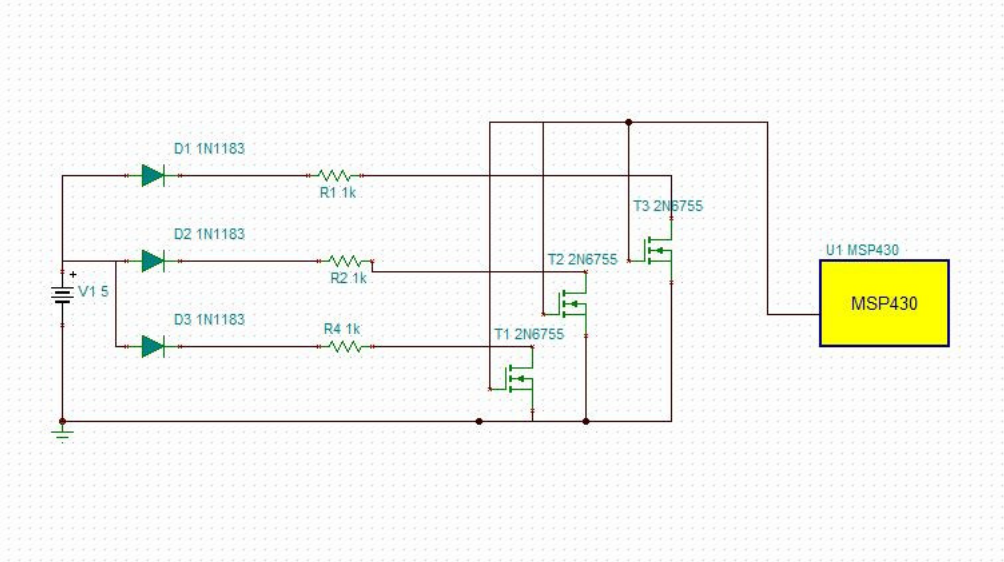

Figure 6: Schematic for breadboard RGB LED.

5 Evaluation and Results

Our program receives an input from the transmitting UART, in the form of an ASCII character. This character is represented by a byte. Our program takes the byte value of the next three characters and converts each to a number which ends up being between 0 and 255. The RGB LED will use these numbers to determine the brightness of the red, green, and blue component of the RGB LED using a PWM.(Figure 3) If there are more than 3 characters, the rest are stored into a fixed size array of size 80. These extra characters are then sent as a message on the TX line for the next board in line to use.

In order to send the message to the FR5594, a PuTTy terminal need to be set up. First, the COM port need to be determined by opening up the ”Device Manager” (for Windows). Look for ”Ports” -¿ ”MSP430 Application UART1” to get the COM number. After determining what port the FR5994 is connected to, one need to input this information into PuTTy by switching to ”Serial” -¿ ”Serial Line”. From here, a terminal will be opened and message can be sent directly to the processor. The message, which is a string of characters, should be ”echo” back to the terminal once it’s done processing.

In Figure 1 we declare and initialize our fields to 0. We also initialize our timer A to use SMCLK, upmode, and we use a divider of 8. We also set P1.0 and P1.1 to outputs. Both P1.0 and P1.1 are LEDS. Finally, P1.3, P1.4, and P1.5 are all set to general purpose outputs. In figure 2, we initialized the fields for the UART. One of the key features of the initialization was setting UART to use the SMCLK in line 56.

In figure 3, we assign values to redPWM, greenPWM, and bluePWM using the RX buffer. Figure 5 shows the interrupt being generated by the timer. If the timer value is equal to redPWM, greenPWM, or blue PWM, those LEDs will be turned off, resulting in the duty cycle.

After each assignment, byteCount is incremented and the next byte is received from the UART. After the values have been all been assigned, the remaining bytes are loaded from the RX buffer onto the message array for the next board to receive. (Figure 4)

Figure 6 shows the schematic for our common anode RGB LEDs. The MSP430 controls 3 Mosfets, each of which are connected to an LED. When we turn on the red LED inside the processor, what actually happens is that a signal is sent to activate the Mosfet connected to the red LED providing a path for current to flow from the 3.3v power source into ground.

6 Conclusions

The code was run using Putty as the transmitter and also as a receiver. The board processed a message using the least significant three characters to determine the values of its red, green, blue LED duty cycles. Based off of the different messages inputted, the RGB values of the LED changed. Finally, Putty was then used to see whether or not the message was passed on by echoing it back to the terminal.

From this project, a great deal of information was obtained about the communication in between devices using UART. Moreover, knowledge about Mosfets and RGB LEDs was also put to practice. ”Communicating with Will Byers” milestone is a great fit for learning the MSP430 families and the UART applications.