Milestone 1: Stranger Things RGB LED

Karlie Naphy and Anwar Noel
Rowan University

October 22, 2018

1 Introduction

With the use of wireless communication, the creation of universal asynchronous receiver / transmitters has increased. UART is used in many different types of wireless communication including Bluetooth, WiFi, computers, and cellular devices. With it's prevalence in our everyday lives, having an understanding of its basic functions is very useful because of the impact that it has on technology in the past decade with giving us cellular phones and wireless modems. Due to the prevalence of this UARTs, having an understanding of how they work and can be manipulated is essential for the technology of the future.

2 Background

This lab consists of three major aspects of these microcontrollers: timers, pulse width modulation, and UART. In embedded systems, timers are what controls the clock speed of the microcontroller. A slower clock will result in less commands being completed per clock cycle. With these timers, there are also different timers that are incorporated within the microcontroller. These clocks include the auxiliary clock (ACLK) and sub-system master clock (SMCLK). auxiliary clock has a frequency of 32 KHz and the sub-system master clock has a frequency of 1 MHz. The speed of these clocks can be varied by using the an internal divider. The MSP430 series of the microcontroller contains a 2 divider, a 4 divider, and a 8 divider. These shorten the speed of the clock by 2, 4, and 8, respectively. The timers can also be set to different modes. The modes consist of Up mode, Continuous mode, Up/Down mode, and Stop mode. These different modes result in different functions. Stop mode makes the timer stop while up mode continuously counts up from zero to the value stored in the compare and capture register 0 (CCR0). Continuous mode allows the timer to repeatedly count from zero to 65535 and Up/Down mode repeatedly counts up from zero to the value stored in CCR0 and returns to zero. The block diagram for the timer a is shown in Figure 4. The capture and compare registers mentioned before are registers assigned to the timer that holds the data for the comparison to the timer value in the timer register. These timers also have their own output modes. The mode that will be focused on is Toggle/Reset mode. This mode starts out at a low value then shifts to a high value once the timer reaches the value in CCR1. It then resets once the timer reaches the value stored in CCR0. This is a repeated process.

While incorporating the use of the timers and their output modes, a varying output can be made. This output can be defined as pulse width modulation. Pulse width modulation, or PWM, is a way of describing a binary signal which encodes a message into a digital signal. With a timer set to up mode and the output of the timer set to toggle/reset mode, the timer can create an output that depicts pulse width modulation. The timer counts up to the value stored in CCR0, the output toggles high when the timer counts to the value stored in CCR1 and resets low when it counts to the value stored in CCR0. This allows the the duty cycle to be varied depending on the value stored in CCR1. That value can be changed in many different ways which include ways, but more specifically can be changed using UART.

Universal Asynchronous Receiver/Transmitter, or UART, is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. UART works by sending bytes a single bit at a time, starting with a start bit and a stop bit. This byte is then stored in a buffer and transmitted to the next device. This function is controlled by an internal clock within the UART device which controls the speed at which the transfers are completed. The block diagram for the UART circuit of the microprocessor is displayed in Figure 2.

3 Design Overview

This code is capable of programming the MSP4305529 microcontroller, receive and transmit a UART message, and display the output of the received message on a RGB (Red, Green, and Blue) LED. The UART messages are sent as hexadecimal numbers that are received by the microcontroller and used as the PWM output of a pin to power a common anode RGB LED. The message is then forwarded to the a chain of other microprocessors to form a chain of glowing LEDs.

3.1 Device Description

The microprocessor that was used for the circuit was the MSP430F5529. This device is manufactured by Texas Instruments. The device has 63 input/output pins. The high-level output voltage for the MSP430F5529 is 3V with a high-level output current of -5mA. The driving voltage for the device is between 3.6V and 1.8V. It also runs at 8MHz. The device has four 16-bit timers and counters, one with five capture/compare registers, one with seven, and two with three. The device also has two universal serial communication interfaces. USCI A0 and USCI A1 each support automatic baud rate detection and synchronous serial peripheral interfacing. USCI B0 and USCI B1 each support inter-integrated circuits and synchronous serial peripheral interfacing Some applications of this device are analog and digital sensor systems and data loggers. An image of the board is displayed in Figure 5 and a block diagram of the microprocessor is displayed in Figure 6.

3.2 Circuit Description

The circuit used to demonstrate the output of the microcontroller consisted of a RGB LED and three resistors of different values. The values of the resisters were calculated by using Ohm's law, dividing the driving voltage of the specific color diode by the maximum driving current of 20mA. The RGB LED that was used in this circuit was a common anode LED, meaning that all of the LEDs have a common positive node. The resistors used for the red, green, and blue LEDs are 82 Ohms, 110 Ohms, and 160 Ohms, respectively. The inputs to these LEDs are connected to the outputs of the capture/compare registers. This allows us to directly receive the PWM directly from the pin without any extra software. The pins for the red, green and blue LEDs are pin 1, 3, and 4, respectively. Pin 2 of the RGB LED is connected to the 5 volts pin of the microcontroller, since it is what's powering the circuit. The circuit is displayed in Figure 7.

3.3 Design Features

- Use of Timer A
- Pulse Width Modulation
- UART
- Interrupts
- RGB LED

3.4 Featured Applications

- Christmas Lights
- LED Driver
- String Lights

4 Design Theory

4.1 Design Requirement 1: Using Pulse Width Modulation

To implement pulse width modulation into the circuit, we set timer A0 to SMCLK and in up mode. We then set CCR0 to 255, which is the period of the PWM. We then set CCR1, CCR2, and CCR3 to 0 so that the duty cycle for the red, green, and blue LEDs are all zero, respectively. We then set CCTL0, CCTL1, and CCTL2 all to OUTMOD 2, which sets the output mode to toggle/reset mode. This allows the output of the timer to alter its duty cycle based on the value set to each specific capture/compare register. The outputs for each of the LEDs were set to the outputs of each of the compare/capture registers for timer A0. Following that, the select bits for the special function registers for each LED is set to 1 to allow the output selected to be that of the capture/compare registers. The capture/compare registers were then set to the receiving buffer of the UART system. This means that each time a new set of bytes is received, a new duty cycle is determined.

4.2 Design Requirement 2: Using UART

To set up UART, we set the P4SEL function to BIT4 and BIT5 which allows BIT4 to become the transfer output and BIT5 to become the receiving input. This is done to be able to receive and transfer bits that we receive from other devices and from the UART cable. The USCI clock source is set to SMCLK which runs at 1MHz. That allows us to complete one million operations per clock cycle. The baud rate is set to 9600 which sets the speed of communication over the data channel to 9600 pulses per second. A switch statement was used inside of the USCI interrupt to allow the microcontroller to select which byte is to be used for the size and each LED. The first byte received is for the size of the array being sent, the second byte is the value that's stored in CCR1 for the red LED, the third byte is the value that's stored in CCR2 for the green LED, and the fourth byte is the value that's stored in CCR3 for the blue LED. The size byte is the first byte of the array. We allowed a variable to be equal to the size of the array + 1 and then transfer that value subtracted by 4 to the next device. This allows all of the bytes to be transferred without effecting the size byte. The next case sets the duty cycle for the red LED, followed by the duty cycle of the green LED, and finally followed by the duty cycle of the green LED. We defaulted our switch statement to transmit whatever is in the receiving buffer if there is no transmit interrupt and no USCI interrupt. We also increment by byte and reset our count when the number of bytes is equal to the size of the array.

4.3 Design Requirement 3: RGB LED Circuit

The RGB LED circuit, which is displayed in Figures 7 and 8, is essential for displaying the PWM output of the microcontroller. Since the RGB LED is of the common anode configuration, the 5V line from the microcontroller is connected to pin 2 of the RGB LED, which is the driving pin of the LED, displayed in Figure 3. The output pins used for the PWM is connected to their specific pins of the RGB LED (1 for red, 3 for green, 4 for blue) which is interconnected by their specified resistors which were predetermined prior to building the circuit. That was done to keep the LEDs from short circuiting due to receiving a very high current rating.

5 Getting Started

To power the device, connect the USB port to your computer and the other end to the microcontroller kit and run the set program. This programs the microcontroller with the code. Power the RGB circuit by placing a wire from the 5V pin on the microcontroller kit to the second pin of the RGB LED. Next, attach wires to pins P1.2, P1.3, and P1.4, to the pins 1, 3, and 4 of the RGB LED, respectively. This allows the outputs of the PWM for each capture/compare register to power the three colors of the LED.

To interface with this device using software, the code must be written in Texas Instrument's Code Composer Studio software. This software has the ability to flash the microcontroller with your written code and program the microcontroller accordingly. To test this software, it must be done using the software RealTerm. This is done by setting the display as to hexadecimal space. This allows the user to send hexadecimal numbers to the microcontroller. Following that, the port chosen must be the UART port of the computer. Following that, you can now send values in groups of four to the microprocessor and the LED will change colors based on the outputs given. Also, the values that are sent back depend on the values that are given.

5.1 Hierarchy Chat

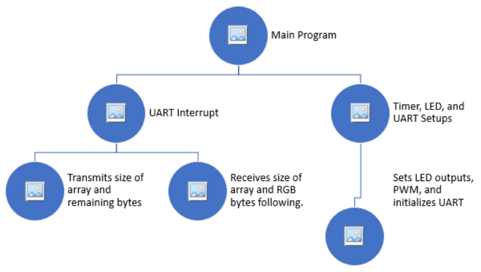
Main Program
UART Interrupt
Timer, LED, and UART Setups
Transmits size of array and remaining bytes
Receives size of array and RGB bytes following.
Sets LED outputs, PWM, and initializes UART

Figure 1: Hierarchy Chart for Milestone 1

6 Test Setup

To demonstrate functionality, we used RealTerm to send UART signals over USB to the microcontroller. Each signal consists of a start bit which denotes how long the signal is in terms of bytes. For example, a start bit ”0x03” means there are 3 following bytes in the serial signal. The following bytes in the signal determine the pulse width modulation of each color LED within the RGB LED. The first byte after the start bit is for the red LED’s PWM, then the second is for the green, and the third for the blue. If the start bit denotes a longer signal, still only the first 3 bytes will be processed by the micro controller and the following bytes will be transmitted back to RealTerm.

RealTerm must be properly configured to function with the microcontroller. In the Display tab, Hex[Space] must be selected and Half Duplex checked. Under the Port tab, the baud rate of 9600 must be entered and the proper COM port of the computer you’re testing on must be selected. Remember to click ”Change”. In the Send tab is where you can send signals to the microcontroller.

6.1 Test Data

The following signals were sent through RealTerm to test the functionality of the microcontroller and the RGB LED.

```
0x03 0xFF 0x00 0x00 //Red LED on
0x03 0x00 0xFF 0x00 //Green LED on
0x03 0x00 0x00 0xFF //Blue LED on
0x03 0xFF 0xFF 0xFF //White LED on
0x03 0x7F 0x7F 0x7F //Half white LED on
0x06 0x00 0xFF 0xFF 0xFF 0x00 0x00 //Cyan LED on with 255 0 0 output
```

As each signal was sent over UART from RealTerm, our microcontroller output the expected values. When the start bit was 0x03, the microcontroller would output 00, meaning it received the full signal and recognized there was no more data in the signal. The microcontroller effectively received all the data and controlled the LED accordingly for each signal tested.

7 Design Files

7.1 Schematics and Figures

7.2 Link to Code

https://github.com/RU09342-F18/milestone-1-clout/blob/master/main.

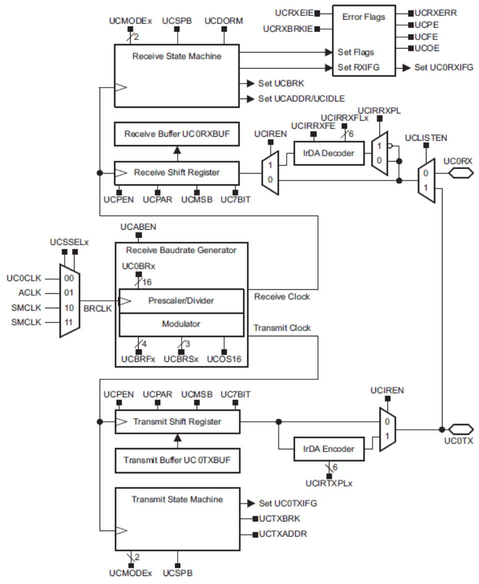

Figure 2: UART Block Diagram

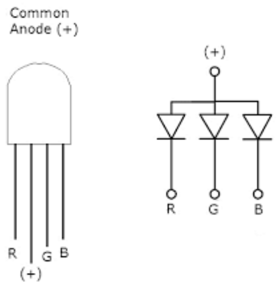

Figure 3: Breakdown of Common Anode RGB LED

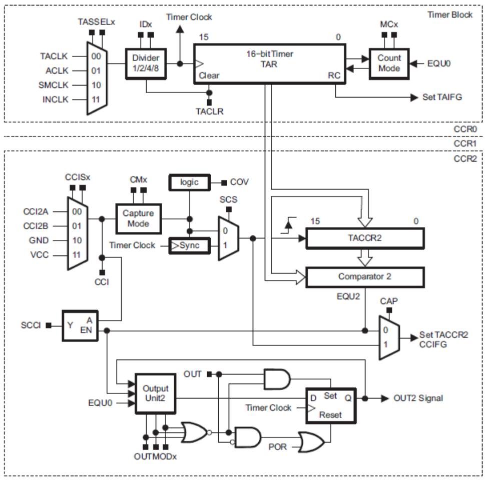

Figure 4: Timer A Block Diagram

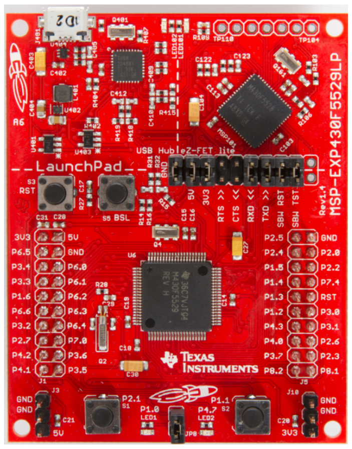

Figure 5: Image of MSP430F5529

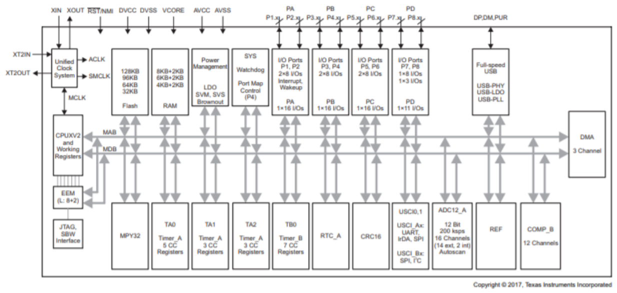
XIN
XOUT
RST/NMI
DVCC
DVSS
VCORE
AVCC
AVSS
DP,DM,PUR
XT2IN
XT2OUT
Unified Clock System
ACLK
SMCLK
MCLK
128KB 96KB 64KB 32KB Flash
8KB+2KB 6KB+2KB 4KB+2KB RAM
Power Management LDO SVM, SVS Brownout
SYS Watchdog Port Map Control (P4)
I/O Ports P1, P2
I/O Ports P3, P4
I/O Ports P5, P6
I/O Ports P7, P8
Full-speed USB USB-PHY USB-LDO USB-PLL
CPUXV2 and Working Registers
MAB
MDB
DMA 3 Channel
EEM (L: 8+2)
JTAG, SBW Interface
MPY32
TA0 Timer_A 5 CC Registers
TA1 Timer_A 3 CC Registers
TA2 Timer_A 3 CC Registers
TB0 Timer_B 7 CC Registers
RTC_A
CRC16
USCI0,1
ADC12_A
REF
COMP_B 12 Channels
Copyright © 2017, Texas Instruments Incorporated

Figure 6: MSP430F5529 Block Diagram

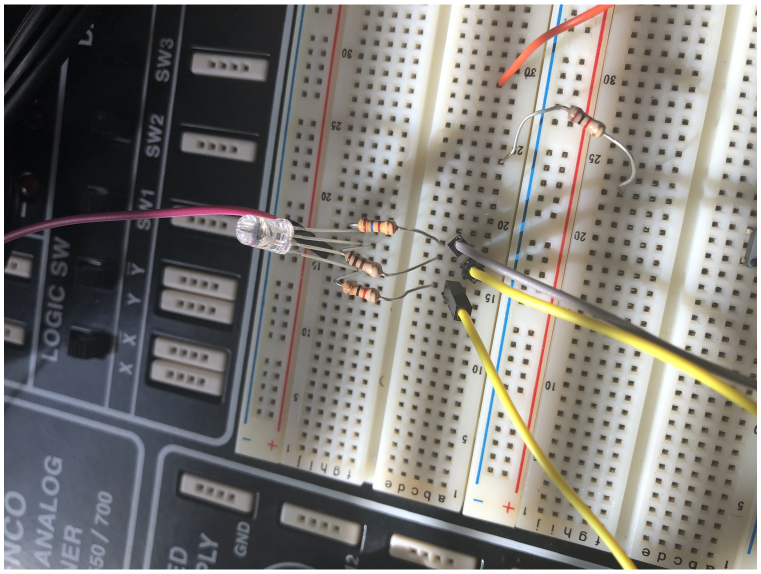

Figure 7: Circuit For RGB LED

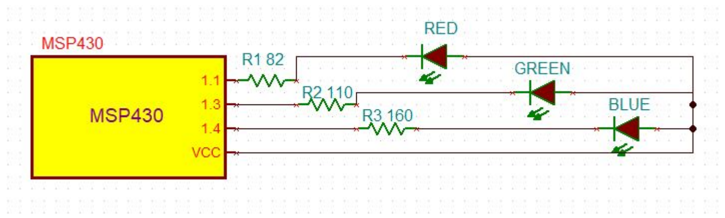

Figure 8: Block diagram of RGB LED circuit