Embbeded Milestone 1

Richard Stelts & Robert Michaelchuck
Rowan University

November 2, 2019

1 Design Overview

This device is designed to be a programming a single addressable (Red, Green, Blue)RGB node on an MSP430G2553 using features such as pulse width modulation control, UART, interrupts,and compare and contrast registers. The program takes in a hexadecimal code initiated by the length, takes the most significant bits in as the pulse width modulation values for the RGB LED then adjusts the length bit and outputs the new sequence to the next node. Note this device is capable of being connected anywhere in the chain of microprocessors and therefore can take in/send out the code via the USB or the pins.

1.1 Design Features

- Receive and send hexcode via usb or pin locations on board
- Programs RGB LEDs
- Knows when to stop if hexcode is missing values
- UART

1.2 Featured Applications

- RealTerm
- Ti code composer 9.1.0
- Github
- TI Resource Center

1.3 Design Resources

- link to code: https://github.com/Intro-To-Embedded-Systems-RU09342/milestone-1-tacorich
- link to real term: https://sourceforge.net/projects/realterm/
- link to code composer: http://software-dl.ti.com/ccs/esd/documents/ccs$_{d}ownloads.html$

1.4 Block Diagram

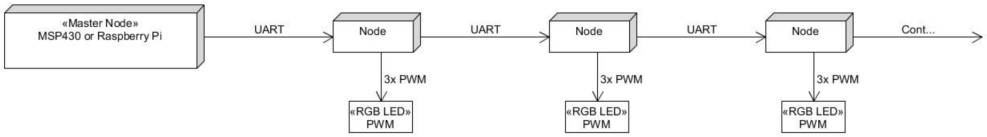
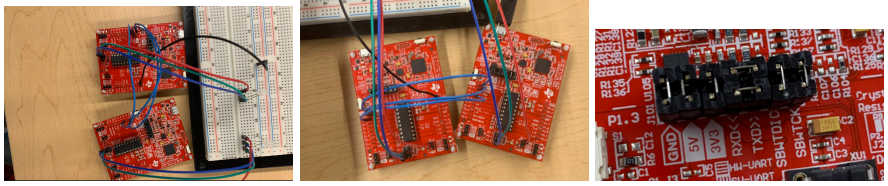


Figure 1: Upper Level Block Diagram

images all from provided lab instructions

1.5 Board Image



2 Key System Specifications

Parameters	Specifications	Details
Read hexcode in from USB	Happens when USB signal is available	
Return unused hex to USB	Happens when USB signal is available	
Read Hexcode from RX input pin	Happens when USB no signal is available	P1.1(RX)
Send Hexcode to TX output pin	Happens when USB no signal is available	P1.2(TX)
Output PWM signal from hexcode	Happens if Hex is available to convert/send	send them to pin 1.6,2.5,and 2.6

3 System Description

The purpose of this device is to essentially use an MSP430G2553 as a single node in an assembly of addressable RGB light strands. To do this the board needed to be able to detect and handle the data length input to it, knowing whether if it had the right type of data left to run, take off the bytes it ran, adjust the length, and send the signal to the next node. Then take the received data and send it to an RGB light as a PWM and correctly display the desired colors.

3.1 Detailed Block Diagram

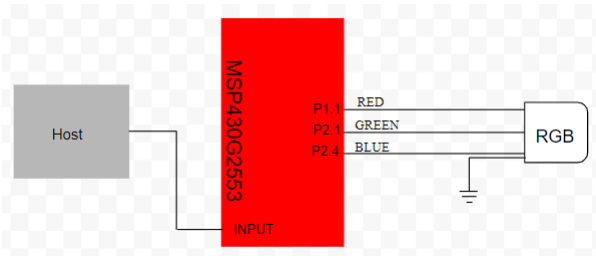


Figure 2: Nodal Model

3.2 Highlighted Devices

- MSP430G2553 - board was chosen to be used to act as the node
- Realterm - Interface Used to power the MSP430G2553 and send a hex string to the board.

3.3 MSP430G2553

The MSP430G2553 Micro controller is a 16-bit MCUs with ultra-low-power and integrated analog and digital peripherals for sensing and measurement applications. MSP430 boards have a large amount of clear example code and secondary devices available from TI. this board also has a very detailed data sheet available at TI's website at anytime. The other option for this project was to use the MPP430FR6989 but this boards data sheet was a lot more difficult to navigate. There was an issue with the TA0CCR3 register not being accessible. This may be an option to be used in a future and may be worth exploring it this project is redone. The UART code would need to be altered in order for that to work.

3.4 RGB LED

The MSP430G2553 addresses the values of an RGB LED. An RGB LED is a combination of three LED's packaged into one LED. RGB LED's have individual addressable pins for each color and one for ground. The duty cycle of each color can be altered by using hardware PWM from the microprocessor. Adjusting the duty cycles for each color can provide almost any color combination.

3.5 UART

UART stands for Universal asynchronous receiver-transmitter, which allows a system to interface with its serial devices. The UART allows for the data format and the transmission speeds to be tailored for your specific needs. It is included in most microprocessors including the MSP430G2553.

4 SYSTEM DESIGN THEORY

4.1 UART

Byte Number	Contents	Example
Byte 0	Number of bytes (N) including this byte	0x50 (80 bytes)
Bytes 1-(N-2)	RGB colors for each node	0xFF (red) 0x00 (green) 0x88 (blue) ...
Byte N-1	End of Message Character	0x0D (carriage return)

Figure 3: UART Format

For this project the UART protocol follows the format of the above figure, at a BAUD rate of 9600. The UART reads the most important byte to the least important byte (left to right). The first byte denotes the length of the incoming string of data and itself. This is used to make certain the data is long enough to fully address the LED and see if any data will be left to send out to a secondary component. Next, the next 3 bytes are processed and send to their corresponding LED pins, first red, then green, then blue. the length is then decreased by three and the data is sent to the TX pin to continue to the next node.

NOTE: In order to this UART with the MSP430G2553 you must turn the pin caps connecting the center board RX and TX pins horizontally as shown on the board.

4.2 Important Pin Locations

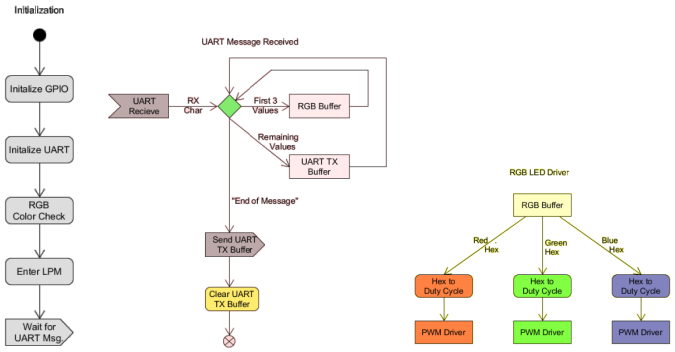
For this project the RGB is controlled with PWM values from onboard pins. The Red-value is P1.6, the Green-value is P2.1, and the Blue-value is P2.5. To connect with other boards the RX, and TX pins on the side of the board will be used. RX is bound to pin P1.1 and TX is bound to pin P1.2.

5 Getting Started Software/Firmware

5.1 Communicating with the Device

1. Download code composer or make an account with TI to acces the cloud version.
2. Download RealTerm as you'll want it for testing later on.
3. Download the code from Github repository (link can be found in 1.3 Design Resources).
4. From here open a new project in Code Composer and past the code from the Github.
5. Click RUN which is a small green arrow in the upper left.
6. If no error occur click the red square(stop) button, this loads the code onto the board. from here the software on the board is ready to be tested.

5.2 Hierarchy Chart



6 Test Setup/ Hardware Setup

Steps- only apply to Board to Board connection

1. Plug MSP430G2553 into computer using USB to Micro-USB cable. After debugging the code, push the code to the processor. This will now stay on the processor after it is removed from power.
2. Put RGB LED onto breadboard with a resistor on each color pin and connect ground pin to ground on the breadboard.
3. With a jumper wire connect the MSP430G2553 ground pin to the breadboard ground to achieve a common ground.
4. With a jumper wire connect P1.6, P2.1, and P2.5 to their respective pin on the LED with an appropriate resister between them. The P1.6 to red, P2.1 to green, and P2.5 to blue.
5. With a jumper wire connect P1.2(TDX) to P1.1(RXD) of the succeeding board.
6. In RealTerm under the display tab change the display to ”Hec[Space]” and check off ”Half Duplex.” Then, click the change button. (Can be seen below on Figure 5) Then, go to the port tab and change baud rate to 9600 and port to the port you plugged your USB to Micro-USB cable into.Finally, click the change button. (Can be seen below on Figure 6)
7. To find the port you are using on your computer, go into device manager and click Communication(COM) port. The port you are using will be listed below as UART.
8. In Realterm under the send tab in the first space, write the hex string 0x08 0x7E 0x00 0xFF 0x40 0xFF 0x00 0x0D. This will output a purplish color on the first LED and yellowish color on the second LED. (Can be seen below on Figure 7)

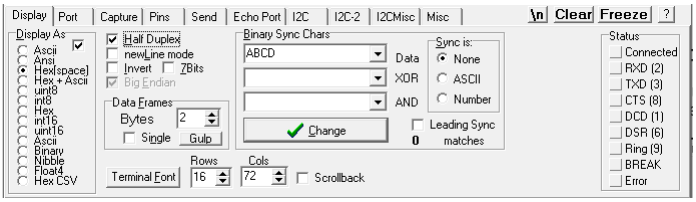


Figure 4: Display Settings

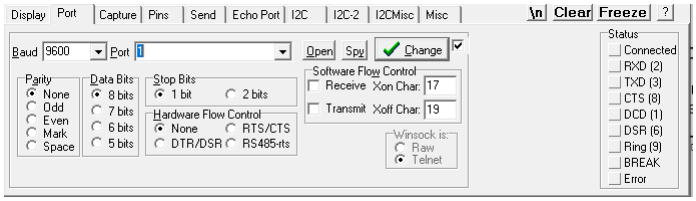


Figure 5: Port Settings

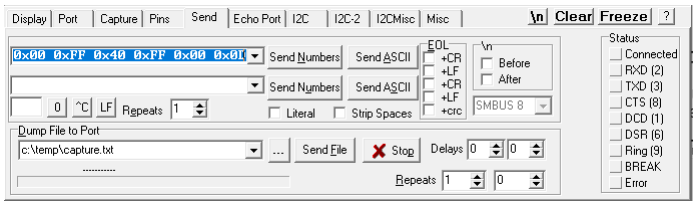


Figure 6: Hexcode Format

6.1 Test Data

Running the following hexcode should result with the first board in the series producing a purple light and the second board producing a yellowish tint.

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 7: TestData

7 Design Files

7.1 Schematics

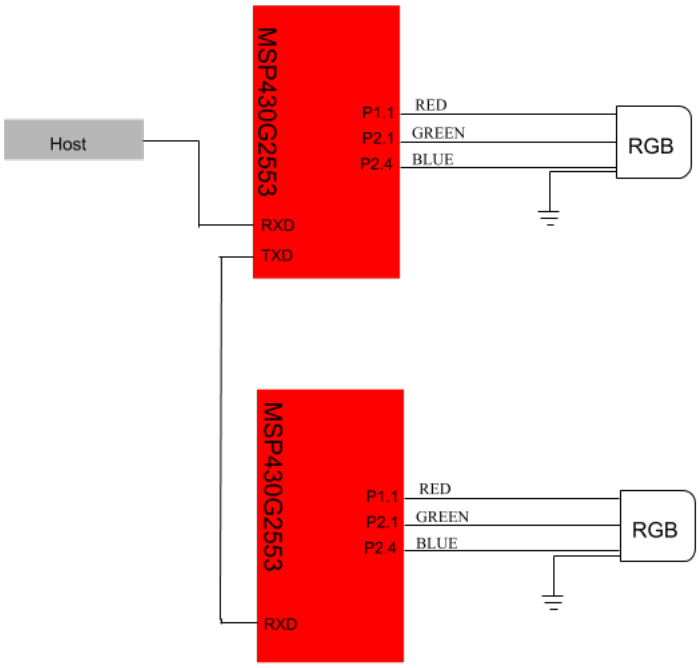


Figure 8: Board to Board setup