Milestone 1 mr. meeseeks code explanation

The code of this lab runs on the functionality of having an input over UART that will designate a number of two bit hex values. Each of these 2 bit hex values will have a range of 0-F. In this case each of these hex values indicate the value of the voltage that will pass through the processor to the LED and will result in 1 of 3 lights in a RGB LED having a voltage and in turn having a high signal. A set of 3 2bit hex values will be used to correlate to each node of the RGB LED in order to put each of those LEDS to high. Meaning an input value of 00 FF FF will only provide voltage to the Red LED so the LED will appear Red. A fourth variable indicates the pressing of the button of the databoard, this value indicates when the next input from UART should be processed.

Another aspect of this lab is that UART can send multiple strings of colors, meaning multiples of 3 2bit Hex values, which will take the initial set of 2 bit hex for the LEDs of the current processor and “pass on” the remaining sets of hex values in order to light the other LEDS in order.

For the Code each color was designated to a specific bit.

|  |
| --- |
| #define RED BIT5 |
|  |

|  |
| --- |
| #define GREEN BIT6 |
|  |

#define BLUE BIT7

With RxCount being how many bytes have been received and nBytes being the number of bits stored in the current packet.

An important distinction to be made here is that 1) the color indicated in the code only applies to that color of the LED if the output pin of the board correlates to the correlating pin of the desired color on the LED. Meaning if BIT7 (blue) was wired to the red light pin, then all values for the “Blue” led would light up the red LED. This is important for understanding the relationship between hardware and software. The second distinction is that correlating colors of the final output of the RGB LED (example blue and red make purple) only occur when red and blue have the same value output and green is off.

The code actually works by initiating the relationship between RxCount and nBytes.

|  |
| --- |
| else if (RxCount > 3 & RxCount <= nBytes -1) { // Pass these along after the first LED color segments have been seen |
|  |

|  |
| --- |
| if (!(RxCount == (nBytes-1))){ //if segments is more than half of total segments then dont change anything |
|  |

|  |
| --- |
| UCA0TXBUF = UCA0RXBUF; |
|  |

RxCount ++;

This final function will keep track of the desired usable bytes (4 button, and each of the three RGB colors) and when there are 3 sets used then to pass on the remaining number of 2 bit hex values. This is done so the databoard does not try to process several desired RGB LED output signals at the same time.

The code actually works by initializing the first of 3 2bit hex values and subtracting that value from 255. (FF would be off and 00 would be 100% strength). The code shown before shows that after this value has been solved, then the code will incriminent to the next of the 2bit hex values.

|  |
| --- |
| if (RxCount == 0 ) { //start at bit segment 0 (0x08) and subtract 3 this is how many are being sent to the next processor |
|  |

|  |
| --- |
| UCA0TXBUF = UCA0RXBUF - 3; //subtract 3 bits |
|  |

|  |
| --- |
| nBytes = UCA0RXBUF; |
|  |

|  |
| --- |
| RxCount ++; //move on to the next segment |
|  |

}

After completing this cycle for 3 2bit hex values. The code shown below will send the code to the next processor.

|  |
| --- |
| } |
|  |

|  |
| --- |
| else if (RxCount > 3 & RxCount <= nBytes -1) { // Pass these along after the first LED color segments have been seen |
|  |

|  |
| --- |
| if (!(RxCount == (nBytes-1))){ //if segments is more than half of total segments then dont change anything |
|  |

|  |
| --- |
| UCA0TXBUF = UCA0RXBUF; |
|  |

|  |
| --- |
| RxCount ++; |
|  |

|  |
| --- |
| //if (RxCount == nBytes) // EOM bad code we kept just in case |
|  |

|  |
| --- |
| // RxCount = 0; |
|  |

}