CS 2230,Fall 2016, Trenary

Final Project

“You’ve Got Teal !”

Parker Loomis

I) This program detects whether or not the button is pushed. If it is it interrupts the flow of the light and changes the color. It also has a dimmer which dims the light, and should change the display on the seven segment.

II) The hardware used would be a seven segment display, the light, dimmer, and of course the MSP 430

a. I used the MSP processor, with the light, dimmer, and seven segment.

b. The MSP send power to the light, dimmer and seven segment. The code tells the MSP what to do with these three.

c. 

III) <Software Organization>

a) The algorithm I didn’t figure out for the seven segment.

b) Had an interrupt module, and a check button method along with the main.

c) Used a delay I order to have the ADC ref to settle.

IV) Began from basically scratch since I didn’t get assignment 8 done. I figured out how to use interrupts and change the color. I had a lot of trouble figuring out how to get the seven segment to work. I couldn’t get it to work at all.

V) My final project in basically assignment 8. I got the light to dim and change colors when hitting the button. I was unable to fully get the seven segment to display number or change when using the dimmer. It does change what is displayed when you hit the button.

VI) #include <msp430.h>

unsigned int color = 1;

int main(void)

{

WDTCTL = WDTPW | WDTHOLD; // wizard words

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

P1DIR = BIT6 | ~(BIT3);

P2DIR = BIT5 | BIT1; //when commented its red // BIT6 alternate output

P1REN |= BIT3;

P2SEL = BIT5 | BIT1;

P1SEL = BIT6;

P1OUT |= BIT3;

P1IE |= BIT3;

P1IES |= BIT3;

P1IFG &= ~BIT3;

TA0CTL = TASSEL\_2 | MC\_1 | ID\_3; // use TA0.1 for PWM on P1.6

TA0CCR0 = 0x3FF; // 10-bit maximum value

TA0CCR1 = 0; // start off

TA0CCTL0 = CCIE;

TA0CCTL1 = OUTMOD\_7; // reset/set output mode

TA1CTL = TASSEL\_2 | MC\_1 | ID\_3; // use TA1.1 for PWM on P1.6

TA1CCR0 = 0x3FF; // 10-bit maximum value

TA1CCR1 = 0; // start off

TA1CCTL1 = OUTMOD\_7; // reset/set output mode

TA1CCR2 = 0x3FF; // 10-bit maximum value

TA1CCTL2 = OUTMOD\_7; // reset/set output mode

ADC10CTL1 = INCH\_4 | ADC10DIV\_3; // ADC10 channel 4, clock divider 3

ADC10CTL0 = SREF\_0 | ADC10SHT\_3 | // VCC/VSS ref, 64 x ADC10CLKs

ADC10ON | ADC10IE; // ADC10 enable, ADC10 interrupt enable

ADC10AE0 = BIT4; // analog enable channel 4

\_\_enable\_interrupt(); // interrupts enabled

while(1)

{

\_\_delay\_cycles(10000); // wait for ADC ref to settle

ADC10CTL0 |= ENC + ADC10SC; // sampling and conversion start

//\_\_bis\_SR\_register(CPUOFF | GIE); // go to sleep with interrupts enabled

//TA0CCR1 = ADC10MEM & 0x3F8; // assigns the value held in ADC10MEM to the TA0CCR1 register

if(color == 1){

TA0CCR1 = ADC10MEM & 0x3F8; // assigns the value held in ADC10MEM to the TA0CCR1 register

}else if(color == 2){

TA1CCR1 = ADC10MEM & 0x3F8; // assigns the value held in ADC10MEM to the TA1CCR1 register

}else if(color == 3){

TA1CCR2 = ADC10MEM & 0x3F8; // assigns the value held in ADC10MEM to the TA1CCR2 register

}

}

return 0;

}

#pragma vector=PORT1\_VECTOR

\_\_interrupt void portinterrupt (void){

P1IFG &= ~BIT3;//this wont let me switch without it bit4 is yello??

if(color == 1){

color = 2;

}else if(color == 2){

color = 3;

}else if(color == 3){

color = 1;

}

}

// ADC10 interrupt service routine

#pragma vector=ADC10\_VECTOR

\_\_interrupt void ADC10\_ISR (void)

{

\_\_bic\_SR\_register\_on\_exit(CPUOFF); // wake up

}