Assignment 7

Kyle Rosenthal 5/8/17 S17

Youtube Link

https://www.youtube.com/watch?v=JNTAt26K9u8

Code

main.c

```
//******************************************************************************
//
// MSP432 main.c template - Empty main
//
//******************************************************************************

#include "msp.h"
#include "dac.h"
#include "uart.h"

void main(void) {

    WDTCTL = WDTPW | WDTHOLD;           // Stop watchdog timer


    Setup_DAC();
    Setup_UART();

    while (1) {
        if (readFlag()) {
            clearFlag();
            int val = readVal();
            if (val <= 4095 && val >= 0) {
                Drive_DAC(val);
            }
        }
    }
}
```

uart.h

```
/*
 * uart.h
 *
 *  Created on: May 8, 2017
 *      Author: kmrosent
 */

#ifndef UART_H_
#define UART_H_

int statusFlag;
```

```
int val;
void Setup_UART();
int readFlag();
void setFlag();
void clearFlag();
int readVal();
unsigned char UART0Rx(void);
unsigned char UART0Tx(unsigned char c);
void EUSCIA0_IRQHandler(void);

#endif /* UART_H_ */
```

uart.c

```
/*
 * uart.c
 *
 *  Created on: May 8, 2017
 *      Author: kmrosent
 */

#include "uart.h"
#include "dac.h"
#include "msp.h"

void Setup_UART() {
    val = 0;
    statusFlag = 0;

    __disable_irq();

    EUSCI_A0->CTLW0 |= BIT0;
    EUSCI_A0->MCTLW = 0;
    EUSCI_A0->CTLW0 = 0x0081;
    EUSCI_A0->BRW = 26;
    P1SEL0 |= (BIT2 + BIT3);
    P1SEL1 &= ~(BIT2 + BIT3);
    EUSCI_A0->CTLW0 &= ~BIT0;
    EUSCI_A0->IFG |= EUSCI_A_IFG_RXIFG;
    EUSCI_A0->IE |= EUSCI_A_IE_RXIE;
    //NVIC_SetPriority(EUSCIA0_IRQn, 4);
    NVIC_EnableIRQ(EUSCIA0_IRQn);
    __enable_irq();
}

int readFlag() {
    return statusFlag;
}

void setFlag() {
    statusFlag = 1;
}
void clearFlag() {
    statusFlag = 0;
```

```
}
int readVal() {
    int temp = val;
    val = -1;
    return temp;
}

/* read a character from UART0 */
unsigned char UART0Rx(void) {
    char c;
    while(!(EUSCI_A0->IFG & 0x01)) ;
    c = EUSCI_A0->RXBUF;
    return c;
}

/* write a character to UART */
unsigned char UART0Tx(unsigned char c) {
    while(!(EUSCI_A0->IFG&0x02)) ;
    EUSCI_A0->TXBUF = c;
    return c;
}

void EUSCIA0_IRQHandler(void) {
    char c = EUSCI_A0->RXBUF;
    static int intVal = 0;

    if (c >= '0' && c <= '9') {
        intVal *= 10;
        intVal += c - '0';
    }

    if (c == '\r') {
        while(!(EUSCI_A0->IFG&0x02)) ;
        EUSCI_A0->TXBUF = c;
        c = '\n';
        val = intVal;
        intVal = 0;
        statusFlag = 1;
    }


    while(!(EUSCI_A0->IFG & 0x02)) {}
    EUSCI_A0->TXBUF = c;
}
```

dac.h

```
/*
 * dac.h
 *
 *  Created on: May 8, 2017
 *      Author: kmrosent
 */
```

```
#ifndef DAC_H_
#define DAC_H_


void Setup_DAC(void);
void Drive_DAC(unsigned int level);

#endif /* DAC_H_ */
```

dac.c

```
/*
 * dac.c
 *
 *  Created on: May 8, 2017
 *      Author: kmrosent
 */

#include "dac.h"
#include "msp.h"

void Setup_DAC(void) {
    // Configure port bits for SPI
    P4->DIR |= BIT1;                    // Will use BIT4 to activate /CE on the DAC
    P1SEL0 |= BIT6 + BIT5;              // Configure P1.6 and P1.5 for UCB0SIMO and
UCB0CLK
    P1SEL1 &= ~(BIT6 + BIT5);           //

    // SPI Setup
    EUSCI_B0->CTLW0 |= EUSCI_B_CTLW0_SWRST;   // Put eUSCI state machine in reset

    EUSCI_B0->CTLW0 = EUSCI_B_CTLW0_SWRST |   // Remain eUSCI state machine in reset
                      EUSCI_B_CTLW0_MST   |   // Set as SPI master
                      EUSCI_B_CTLW0_SYNC  |   // Set as synchronous mode
                      EUSCI_B_CTLW0_CKPL  |   // Set clock polarity high
                      EUSCI_B_CTLW0_MSB;      // MSB first

    EUSCI_B0->CTLW0 |= EUSCI_B_CTLW0_SSEL__SMCLK; // SMCLK
    EUSCI_B0->BRW = 0x01;                         // divide by 16, clock =
fBRCLK/(UCBRx)
    EUSCI_B0->CTLW0 &= ~EUSCI_B_CTLW0_SWRST;      // Initialize USCI state machine,
SPI
                                                  // now waiting for something to
                                                  // be placed in TXBUF

    EUSCI_B0->IFG |= EUSCI_B_IFG_TXIFG;  // Clear TXIFG flag
}

void Drive_DAC(unsigned int level){
  unsigned int DAC_Word = 0;
  int i;

  DAC_Word = (0x1000) | (level & 0x0FFF);   // 0x1000 sets DAC for Write
                                            // to DAC, Gain = 2, /SHDN = 1
```

```
                                          // and put 12-bit level value
                                          // in low 12 bits.

  P4->OUT &= ~BIT1;                                 // Clear P4.1 (drive /CS low on
DAC)
                                                    // Using a port output to do
this for now

  EUSCI_B0->TXBUF = (unsigned char) (DAC_Word >> 8);  // Shift upper byte of DAC_Word
                                                      // 8-bits to right

  while (!(EUSCI_B0->IFG & EUSCI_B_IFG_TXIFG));             // USCI_A0 TX buffer
ready?

  EUSCI_B0->TXBUF = (unsigned char) (DAC_Word & 0x00FF);     // Transmit lower byte
to DAC

  while (!(EUSCI_B0->IFG & EUSCI_B_IFG_TXIFG));      // Poll the TX flag to wait for
completion

  for(i = 200; i > 0; i--);                         // Delay 200 16 MHz SMCLK
periods
                                                    //to ensure TX is complete by
SIMO

  P4->OUT |= BIT1;                                  // Set P4.1   (drive /CS high on
DAC)

  return;
}
```