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**;

}