Steven Lee
Github root directory: https://github.com/lees106/STAT411

Date Submitted: 10/26/2019


Lab8_Video_Links.txt

Task 01:

Youtube Link:
https://youtu.be/jz0GHeqmh00

Modified Code:

```
#include <stdbool.h>
#include <stdint.h>
#include "\ti\tivaware_c_series_2_1_4_178\inc\hw_memmap.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\gpio.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\pin_map.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\ssi.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\sysctl.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\uart.h"
#include "uartstdio.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\adc.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\debug.h"

#define NUM_SSI_DATA            3

void
InitConsole(void)
{

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);

    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTStdioConfig(0, 115200, 16000000);
}
```

```
int
main(void)
{
#if defined(TARGET_IS_TM4C129_RA0) ||
\
    defined(TARGET_IS_TM4C129_RA1) ||
\
    defined(TARGET_IS_TM4C129_RA2)
    uint32_t ui32SysClock;
#endif

#if defined(TARGET_IS_TM4C129_RA0) ||
\
    defined(TARGET_IS_TM4C129_RA1) ||
\
    defined(TARGET_IS_TM4C129_RA2)
    ui32SysClock = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
                                       SYSCTL_OSC_MAIN |
                                       SYSCTL_USE_OSC), 25000000);
#else
    SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
                   SYSCTL_XTAL_16MHZ);
#endif

    InitConsole();

    UARTprintf("SSI ->\n");
    UARTprintf("  Mode: SPI\n");
    UARTprintf("  Data: 8-bit\n\n");

    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    GPIOPinConfigure(GPIO_PA2_SSI0CLK);
    GPIOPinConfigure(GPIO_PA3_SSI0FSS);
    GPIOPinConfigure(GPIO_PA4_SSI0RX);
    GPIOPinConfigure(GPIO_PA5_SSI0TX);

    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_5 | GPIO_PIN_4 | GPIO_PIN_3 |
                   GPIO_PIN_2);

#if defined(TARGET_IS_TM4C129_RA0) ||
\
    defined(TARGET_IS_TM4C129_RA1) ||
\
    defined(TARGET_IS_TM4C129_RA2)
    SSIConfigSetExpClk(SSI0_BASE, ui32SysClock, SSI_FRF_MOTO_MODE_0,
                       SSI_MODE_MASTER, 1000000, 8);
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
#else
    SSIConfigSetExpClk(SSI0_BASE, SysCtlClockGet(), SSI_FRF_MOTO_MODE_0,
                       SSI_MODE_MASTER, 1000000, 8);
#endif
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;


SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16M
HZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);

ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 1);

    SSIEnable(SSI0_BASE);
    while(1){


    ADCIntClear(ADC0_BASE, 1);
    ADCProcessorTrigger(ADC0_BASE, 1);

    while(!ADCIntStatus(ADC0_BASE, 1, false))
    {
    }


    ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
      ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
      ui32ADC0Value[3] + 2)/4;
      ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
      ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

    while(SSIDataGetNonBlocking(SSI0_BASE, &pui32DataRx[0]))
    {
    }

    pui32DataTx[0] = ui32TempValueF;
    pui32DataTx[1] = ui32TempValueF;
    pui32DataTx[2] = ui32TempValueF;
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
    UARTprintf("\nSent:\n  ");

    for(ui32Index = 0; ui32Index < 1; ui32Index++)
    {
        UARTprintf("'%u' ", pui32DataTx[ui32Index]);
        SSIDataPut(SSI0_BASE, pui32DataTx[ui32Index]);
    }

    SysCtlDelay(10000000);

    while(SSIBusy(SSI0_BASE))
    {
    }

    UARTprintf("\nReceived:\n  ");

    for(ui32Index = 0; ui32Index < 1; ui32Index++)
    {
        SSIDataGet(SSI0_BASE, &pui32DataRx[ui32Index]);

        pui32DataRx[ui32Index] &= 0x00FF;

        UARTprintf("'%u' ", pui32DataRx[ui32Index]);
     }
    }
    SysCtlDelay(10000000);

    return(0);
}
```

Task 02:

Youtube Link:
https://youtu.be/r6PYxtap78E

Modified Code:

```
#include <stdbool.h>
#include <stdint.h>
#include "\ti\tivaware_c_series_2_1_4_178\inc\hw_memmap.h"
#include "\ti\tivaware_c_series_2_1_4_178\inc\hw_types.h"

#include "\ti\tivaware_c_series_2_1_4_178\driverlib\gpio.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\pin_map.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\ssi.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\sysctl.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\uart.h"
#include "\ti\tivaware_c_series_2_1_4_178\utils\uartstdio.h"
```

```
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\adc.h"
#include "\ti\tivaware_c_series_2_1_4_178\driverlib\debug.h"

#define NUM_LEDS 8
uint8_t frame_buffer[NUM_LEDS*3];
void send_data(uint8_t* data, uint8_t num_leds);
void fill_frame_buffer(uint8_t r, uint8_t g, uint8_t b, uint32_t num_leds);
static volatile uint32_t ssi_lut[] = {
    0b100100100,
    0b110100100,
    0b100110100,
    0b110110100,
    0b100100110,
    0b110100110,
    0b100110110,
    0b110110110
};

int main(void) {

    FPULazyStackingEnable();

    // 80MHz
    SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
                   SYSCTL_OSC_MAIN);

    //initialize SPI
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlDelay(50000);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);
    SysCtlDelay(50000);

    GPIOPinConfigure(GPIO_PA5_SSI0TX);
    GPIOPinConfigure(GPIO_PA2_SSI0CLK);
    GPIOPinConfigure(GPIO_PA4_SSI0RX);
    GPIOPinConfigure(GPIO_PA3_SSI0FSS);

    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_5);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_2);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_4);
    GPIOPinTypeSSI(GPIO_PORTA_BASE, GPIO_PIN_3);

    //20 MHz data rate
    SSIConfigSetExpClk(SSI0_BASE, 80000000, SSI_FRF_MOTO_MODE_0, SSI_MODE_MASTER,
2400000, 9);
    SSIEnable(SSI0_BASE);

    while(1)
    {
        // RED
        fill_frame_buffer(255, 0, 0, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
        // GREEN
        fill_frame_buffer(0, 255, 0, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);

        // BLUE
        fill_frame_buffer(0, 0, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);

        // YELLOW
        fill_frame_buffer(255, 255, 0, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);

        // PURPLE
        fill_frame_buffer(255, 0, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);

        // LIGHT BLUE
        fill_frame_buffer(0, 255, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);

        // WHITE
        fill_frame_buffer(255, 255, 255, NUM_LEDS);
        send_data(frame_buffer, NUM_LEDS);
        SysCtlDelay(SysCtlClockGet()/5);
    }


}

void send_data(uint8_t* data, uint8_t num_leds)
{
    uint32_t i, j, curr_lut_index, curr_rgb;
    for(i = 0; i < (num_leds*3); i = i + 3) {
        curr_rgb = (((uint32_t)data[i + 2]) << 16) | (((uint32_t)data[i + 1]) << 8) |
data[i];
        for(j = 0; j < 24; j = j + 3) {
            curr_lut_index = ((curr_rgb>>j) & 0b111);
            SSIDataPut(SSI0_BASE, ssi_lut[curr_lut_index]);
        }
    }

    SysCtlDelay(50000); // delay more then 50us
}

void fill_frame_buffer(uint8_t r, uint8_t g, uint8_t b, uint32_t num_leds)
{
    uint32_t i;
    uint8_t* frame_buffer_index = frame_buffer;
    for(i = 0; i < num_leds; i++) {
```

```
        *(frame_buffer_index++) = g;
        *(frame_buffer_index++) = r;
        *(frame_buffer_index++) = b;
    }
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.