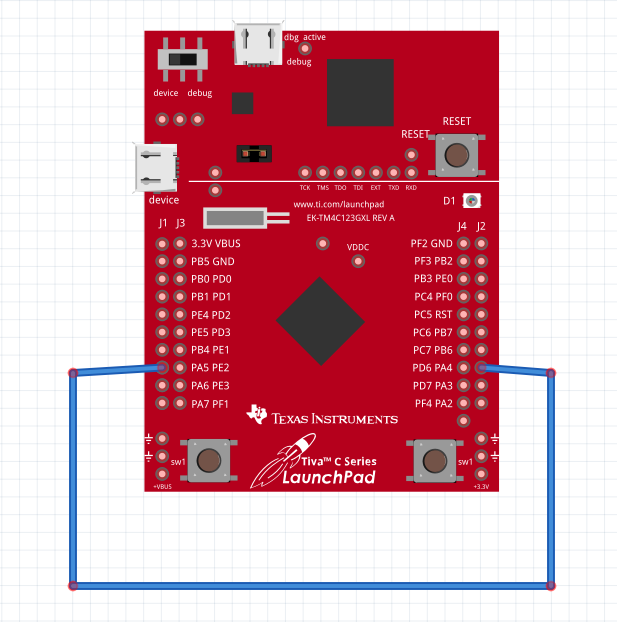
**Date Submitted: 10/19/19**

**------------------------------------------------------------------------------------**

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=tr27sTP2TSI>

**Modified Schematic (if applicable):**



**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

**#include <stdlib.h>**

**#include "inc/hw\_memmap.h"**

**#include "inc/hw\_ssi.h"**

**#include "inc/hw\_types.h"**

**#include "driverlib/ssi.h"**

**#include "driverlib/gpio.h"**

**#include "driverlib/pin\_map.h"**

**#include "driverlib/sysctl.h"**

**#include "utils/uartstdio.h"**

**#include "utils/uartstdio.c"**

**#include "driverlib/adc.h"**

**#include "inc/hw\_ints.h"**

**#include "driverlib/interrupt.h"**

**#include "driverlib/debug.h"**

**#define NUM\_SSI\_DATA 2**

**void initADC(void)**

**{**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0); // enable ADC0**

**ADCHardwareOversampleConfigure(ADC0\_BASE, 32);**

**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);**

**}**

**int main(void)**

**{**

**SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ); //40mhz clock**

**initADC();**

**//I had the following UART initialization lines in the init Console function**

**//but my program kept going to the FaultISR so I moved them to main and then it began working**

**// Enable GPIO port A which is used for UART0 pins.**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);**

**// Configure the pin muxing for UART0 functions on port A0 and A1.**

**GPIOPinConfigure(GPIO\_PA0\_U0RX);**

**GPIOPinConfigure(GPIO\_PA1\_U0TX);**

**// Enable UART0 so that we can configure the clock.**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);**

**// Use the internal 16MHz oscillator as the UART clock source.**

**UARTClockSourceSet(UART0\_BASE, UART\_CLOCK\_PIOSC);**

**// Select the alternate (UART) function for these pins.**

**GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);**

**// Initialize the UART for console I/O.**

**UARTStdioConfig(0, 115200, 16000000);**

**// The SSI0 peripheral must be enabled for use.**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_SSI0);**

**// Configure the pin muxing for SSI0 functions on port A2, A3, A4, and A5.**

**// This step is not necessary if your part does not support pin muxing.**

**GPIOPinConfigure(GPIO\_PA2\_SSI0CLK);**

**GPIOPinConfigure(GPIO\_PA3\_SSI0FSS);**

**GPIOPinConfigure(GPIO\_PA4\_SSI0RX);**

**GPIOPinConfigure(GPIO\_PA5\_SSI0TX);**

**// Configure the GPIO settings for the SSI pins. This function also gives**

**// control of these pins to the SSI hardware. Consult the data sheet to**

**// see which functions are allocated per pin.**

**// The pins are assigned as follows:**

**// PA5 - SSI0Tx**

**// PA4 - SSI0Rx**

**// PA3 - SSI0Fss**

**// PA2 - SSI0CLK**

**GPIOPinTypeSSI(GPIO\_PORTA\_BASE, GPIO\_PIN\_5 | GPIO\_PIN\_4 | GPIO\_PIN\_3 |**

**GPIO\_PIN\_2);**

**GPIOPinWrite(GPIO\_PORTA\_BASE, GPIO\_PIN\_4, GPIO\_PIN\_4);**

**// Configure and enable the SSI port for SPI master mode. Use SSI0,**

**// system clock supply, idle clock level low and active low clock in**

**// freescale SPI mode, master mode, 1MHz SSI frequency, and 8-bit data.**

**SSIClockSourceSet(SSI0\_BASE, SSI\_CLOCK\_SYSTEM);**

**SSIConfigSetExpClk(SSI0\_BASE, SysCtlClockGet(), SSI\_FRF\_MOTO\_MODE\_0,**

**SSI\_MODE\_MASTER, 1000000, 8);**

**// Enable the SSI0 module.**

**SSIEnable(SSI0\_BASE);**

**// Display the setup on the console.**

**UARTprintf("SSI ->\r\n");**

**UARTprintf(" Mode: SPI\r\n");**

**UARTprintf(" Data: 8-bit\r\n\r\n");**

**//SSI variables**

**//uint32\_t pui32DataTx[NUM\_SSI\_DATA];**

**uint32\_t pui32DataRx[NUM\_SSI\_DATA];**

**uint32\_t ui32Index;**

**//ADC variables**

**uint32\_t ui32ADC0Value[4];**

**uint32\_t ui32TempAvg;**

**uint32\_t ui32TempValueC;**

**uint32\_t ui32TempValueF;**

**char temperatureTx[10];**

**while(1)**

**{**

**while(SSIDataGetNonBlocking(SSI0\_BASE, &pui32DataRx[0]))**

**{**

**}**

**// turn on ADC**

**ADCIntClear(ADC0\_BASE, 1);**

**ADCSequenceEnable(ADC0\_BASE, 1);**

**ADCProcessorTrigger(ADC0\_BASE, 1);**

**while(!ADCIntStatus(ADC0\_BASE, 1, false))**

**{**

**//wait until ADC finishes**

**}**

**//Get ADC values from SS1**

**ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);**

**//Average and Calculate Temperature**

**ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;**

**ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;**

**ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;**

**//Convert Temperature to Character string**

**ltoa(ui32TempValueF, temperatureTx);**

**ADCSequenceDisable(ADC0\_BASE, 0);**

**// Display indication that the SSI is transmitting data.**

**UARTprintf("\r\nSent:\r\n '");**

**// Send 2 bytes of data.**

**for(ui32Index = 0; ui32Index < NUM\_SSI\_DATA; ui32Index++)**

**{**

**// Display the data that SSI is transferring.**

**UARTprintf("%c", temperatureTx[ui32Index]);**

**// Send the data using the "blocking" put function. This function**

**// will wait until there is room in the send FIFO before returning.**

**SSIDataPut(SSI0\_BASE, temperatureTx[ui32Index]);**

**}**

**UARTprintf("'F");**

**// Wait until SSI0 is done transferring all the data in the transmit FIFO**

**while(SSIBusy(SSI0\_BASE))**

**{**

**}**

**UARTprintf("\r\nReceived:\r\n '");**

**// Receive 2 bytes of data.**

**for(ui32Index = 0; ui32Index < NUM\_SSI\_DATA; ui32Index++)**

**{**

**// Receive the data using the "blocking" Get function. This function**

**// will wait until there is data in the receive FIFO before returning.**

**SSIDataGet(SSI0\_BASE, &pui32DataRx[ui32Index]);**

**// Since we are using 8-bit data, mask off the MSB.**

**pui32DataRx[ui32Index] &= 0x00FF;**

**// Display the data that SSI0 received.**

**UARTprintf("%c", pui32DataRx[ui32Index]);**

**}**

**UARTprintf("'F");**

**SysCtlDelay(5000000);**

**}**

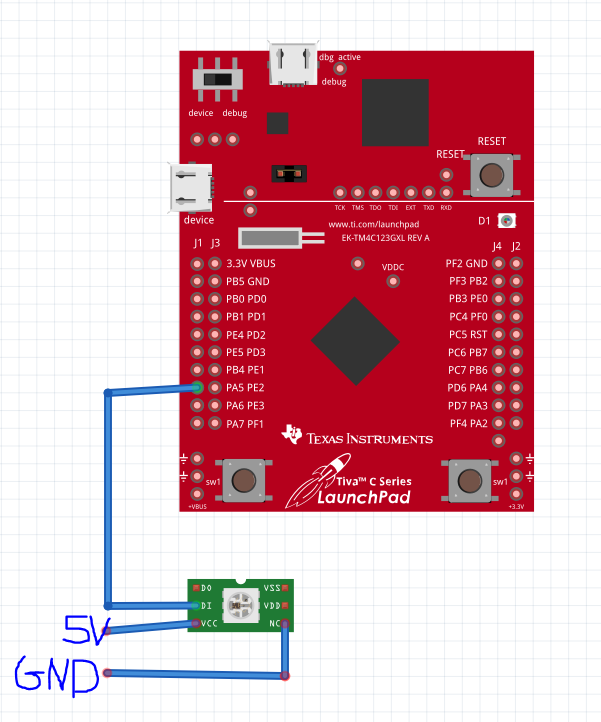
**}**

**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=f2ms46w2_4c>

**Modified Schematic (if applicable):**



**Used single LED since fritzing doesn’t have the WS2818 model.**

**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

**#include <stdlib.h>**

**#include "inc/hw\_memmap.h"**

**#include "inc/hw\_ssi.h"**

**#include "inc/hw\_types.h"**

**#include "driverlib/ssi.h"**

**#include "driverlib/gpio.h"**

**#include "driverlib/fpu.h"**

**#include "driverlib/rom.h"**

**#include "driverlib/pin\_map.h"**

**#include "driverlib/sysctl.h"**

**#include "utils/uartstdio.h"**

**#include "utils/uartstdio.c"**

**#include "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);**

**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)**

**{**

**fill\_frame\_buffer(255, 0, 0, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**fill\_frame\_buffer(0, 255, 0, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**fill\_frame\_buffer(0, 0, 255, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**fill\_frame\_buffer(255, 255, 0, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**fill\_frame\_buffer(255, 0, 255, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**fill\_frame\_buffer(0, 255, 255, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**fill\_frame\_buffer(255, 255, 255, NUM\_LEDS);**

**send\_data(frame\_buffer, NUM\_LEDS);**

**}**

**return 0;**

**}**

**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(5000000); //delay**

**}**

**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;**

**}**

**}**

**------------------------------------------------------------------------------------**