CPE403 – Advanced Embedded Systems

Design Assignment 1

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Angelo Nolasco

Email: Nolasco@unlv.nevada.edu

Github Repository link (root):
https://github.com/AngeloNol/Design_Assignments/tree/master/DesignAssignment/TIVAC

Youtube Playlist link (root): Assignment 1

1. Code for Tasks

```
/* ----------------------        Include Files       ---------------------- */
#define TARGET_IS_BLIZZARD_RB1
#include <stdint.h>                    // Library of Standard Integer Types
#include <stdbool.h>                   // Library of Standard Boolean Types
#include "inc/tm4c123gh6pm.h"          // Definitions for interrupt and register assignments on Tiva C
#include "inc/hw_memmap.h"             // Macros defining the memory map of the Tiva C Series device
#include "inc/hw_types.h"              // Defines common types and macros
#include "inc/hw_gpio.h"               // Defines Macros for GPIO hardware
#include "driverlib/sysctl.h"          // Defines and macros for System Control API of DriverLib
#include "driverlib/interrupt.h"       // Defines and macros for NVIC Controller API of DriverLib
#include "driverlib/gpio.h"            // Defines and macros for GPIO API of DriverLib
#include "driverlib/timer.h"           // Defines and macros for Timer API of driverLib
#include "driverlib/adc.h"             // Defines and macros for ADC API of driverLib
#include "driverlib/rom.h"             // Defines and macros for ROM API of driverLib
#include "driverlib/uart.h"
#include "driverlib/pin_map.h"
#include "utils/uartstdio.h"
#include "driverlib/debug.h"
#include "inc/hw_ints.h"


#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

/* ----------------------      Global Variables      ---------------------- */
uint32_t ui32PinStatus = 0x00000000;    // Variable to store the pin status of GPIO PortF
```

```
uint32_t xValue[6];              // Array to store the ADC values of X
volatile uint32_t xValueAvg;          // Variable to store the Average of ADC values of X
uint32_t yValue[6];              // Array to store the ADC values of Y
volatile uint32_t yValueAvg;          // Variable to store the Average of ADC values of Y
uint32_t ui32Period;
char     buffer[4];

/*-------------------------- Timer Interrupt ----------------------*/
void Timer1IntHandler(void)
{
    // Clear the timer interrupt
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    // Clear the ADC Interrupt (if any generated) for Sequencer 0
        ADCIntClear(ADC0_BASE, 0);
        ADCIntClear(ADC0_BASE, 1);
        // Trigger the ADC Sampling for Sequencer 0
        ADCProcessorTrigger(ADC0_BASE, 0);
        ADCProcessorTrigger(ADC0_BASE, 1);

        ADCSequenceDataGet(ADC0_BASE, 0, xValue);
        ADCSequenceDataGet(ADC0_BASE, 1, yValue);
        // Calculate the Average of the Readings
        xValueAvg = (xValue[0] + xValue[1] + xValue[2] + xValue[3]
              + xValue[4] + xValue[5] + 4)/6;

        yValueAvg = (yValue[0] + yValue[1] + yValue[2] + yValue[3]
                    + yValue[4] + yValue[5] + 4)/6;
```

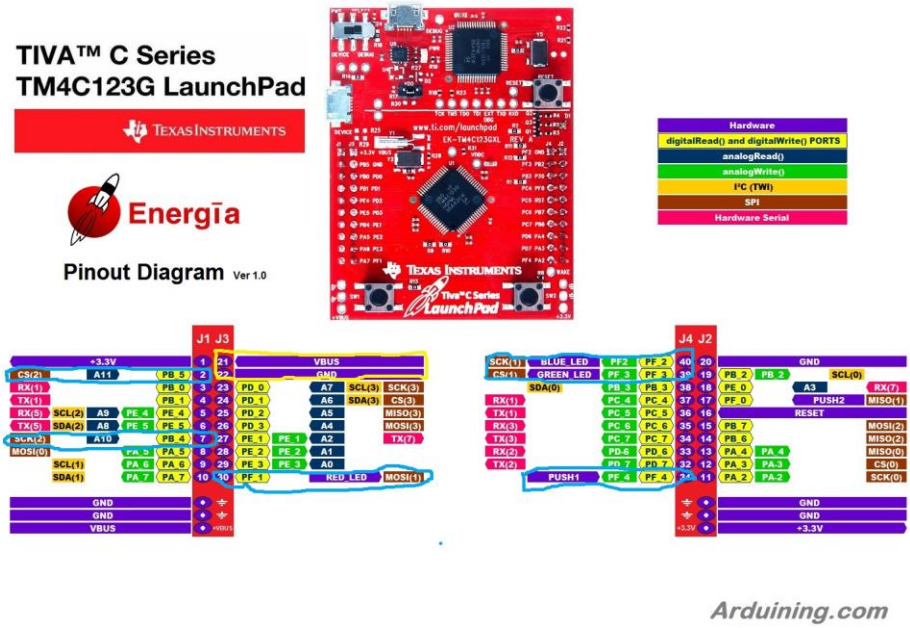
```
    UARTprintf("X: %d\t",xValueAvg );
    UARTprintf("Y: %d",yValueAvg );
    UARTprintf("\n");

}

/* ----------------------        Main Program       ---------------------- */
int main(void){

   // Set the System clock to 80MHz and enable the clock for peripheral PortF,B,A,E,Timer1.
   SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_OSC_MAIN);

// Configure peripherals
   SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);  // Enabling Timer 1

/*------------------------- Task 1 part 1 --------------------*/

   // Configure ADC
   SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
   ADCHardwareOversampleConfigure(ADC0_BASE, 64);
   GPIOPinTypeADC(GPIO_PORTB_BASE, GPIO_PIN_4);//channel 10 for X value
   GPIOPinTypeADC(GPIO_PORTB_BASE, GPIO_PIN_5);//channel 11 for Y value

   ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);// For X value
   ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);// For Y value
```

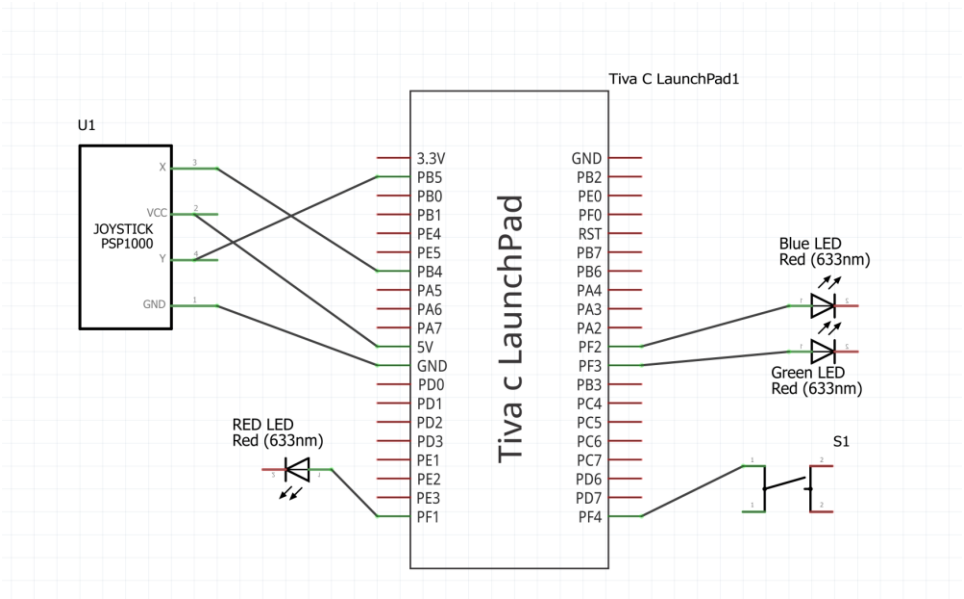
```
    //x value
    ADCSequenceStepConfigure(ADC0_BASE, 0, 0, ADC_CTL_CH10);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 1, ADC_CTL_CH10);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 2, ADC_CTL_CH10);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 3, ADC_CTL_CH10);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 4, ADC_CTL_CH10);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 5, ADC_CTL_CH10|ADC_CTL_IE|ADC_CTL_END);

    //y value
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_CH11);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_CH11);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_CH11);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_CH11);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 4, ADC_CTL_CH11);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 5, ADC_CTL_CH11|ADC_CTL_IE|ADC_CTL_END);

    ADCSequenceEnable(ADC0_BASE, 0);// For X value
    ADCSequenceEnable(ADC0_BASE, 1);// For Y value

// Configure Timer 1 module
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    ui32Period = SysCtlClockGet()/4;   // Period of 1s 2Hz
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

// Configure pins for UART
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
```

```
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);
    UARTStdioConfig(0, 115200, 16000000);

    // Enable interrupts
    IntMasterEnable();
    TimerEnable(TIMER1_BASE, TIMER_A);
    ADCSequenceEnable(ADC0_BASE, 0);
    ADCSequenceEnable(ADC0_BASE, 1);



/*------------------------- Task 1 part 2 ---------------------*/
    // Set the PF1, PF2, PF3 as output and PF4 as input.
    // Connect PF4 to internal Pull-up resistors and set 2 mA as current strength.
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
    GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    // Start an Infinite Loop
    while (true){
       // Read the status of Input
       ui32PinStatus = GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4);

      if (!ui32PinStatus){
          GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
          SysCtlDelay(20000000);


       }else{
          GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
```

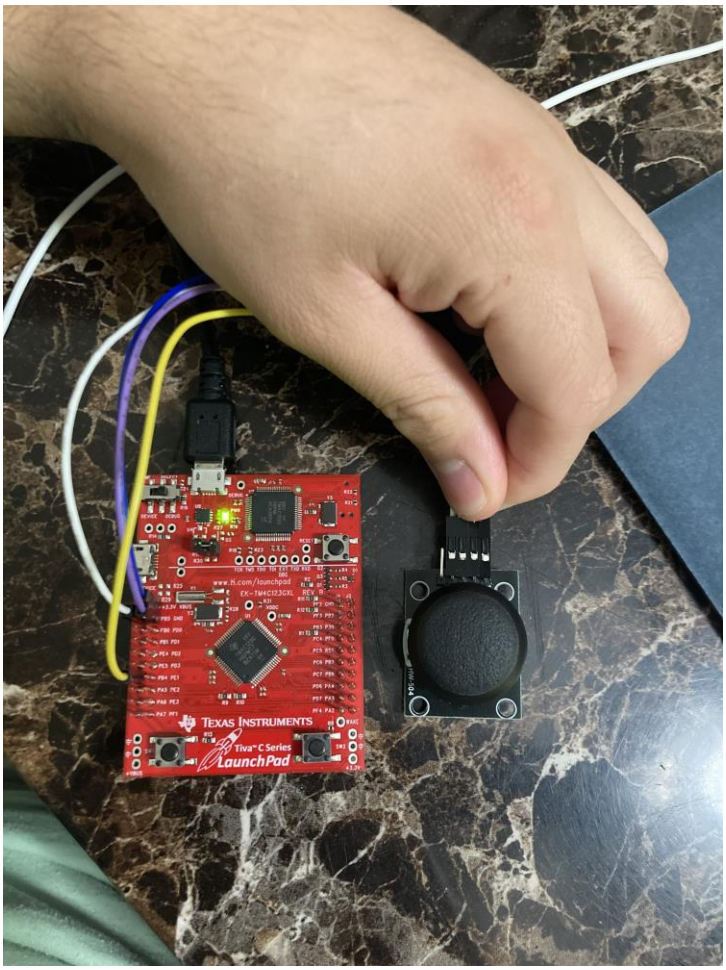
```
        }

      if (!ui32PinStatus){
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 8);
            SysCtlDelay(20000000);

         }else{
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0x0);

         }

      if (!ui32PinStatus){
             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0x0);
             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
             SysCtlDelay(20000000);

          }else{
             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x0);

          }

      if(ui32PinStatus){
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x0);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0x0);
      }

    }
}
```

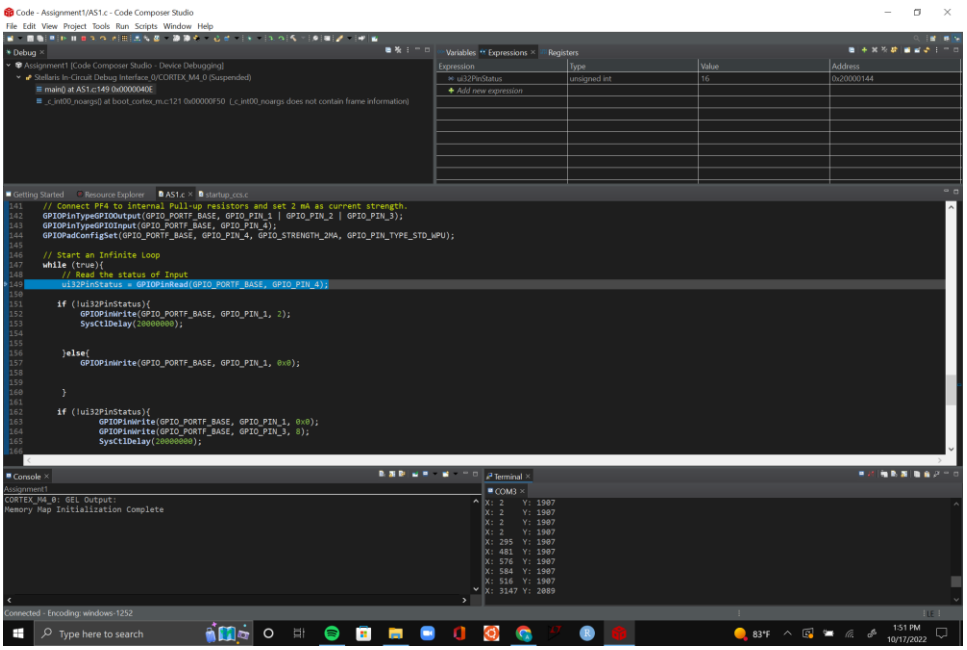
2. Block diagram and/or Schematics showing the components, pins used, and interface.





3. Screenshots of the IDE, physical setup, debugging process





4. Declaration

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".

Angelo Nolasco