CPE403 – Advanced Embedded Systems

Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Rishawn Peppers Johnson

Email: peppersj@unlv.nevada.edu

Github Repository link (root): https://github.com/PeppersJ/v4e0nk_i3

Youtube Playlist link (root): https://drive.google.com/drive/u/2/folders/1fJ029-AAWjTnN-QrRqNLd0iLwKGm6A08

Follow the submission guideline to be awarded points for this Assignment.

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE/403, Lastname, Firstname). Place all labs under the root folder TIVAC, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your youtube videos as playlist under the name “cpe403”. The playlist should have the video sequence arranged as submission or due dates.
7. Only submit pdf documents. Do not forget to upload this document in the github repository and in the canvas submission portal.

1. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

```
/*  Modified By: Rishawn Peppers Johnson
 *  Date Created: 13 November 2020
 *  Device: CC1352R1 & MKII
 *  CpE 403 Assignment 04
 *
 *  Purpose: Interface the TivaC123GH6PM with the Educational BoosterPack MKII to
 *      read the MKII's vertical joystick value using TivaC's ADC, display the ADC
 *      value to terminal through UART, constatly update the PWM duty cycle routed
 *      to an LED.
 *
 *  Inputs:     MKII horizontal joystick
 *  Outputs:    UART to terminal ADC value
 *              PWM value to LED
 *  */
/*
 * Copyright (c) 2015-2019, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
/*
 *  ======== hello.c ========
 */
#include <stdio.h>      // Common C functions
```

```c
/* XDC Module Headers */
#include <xdc/std.h>
#include <xdc/cfg/global.h>
#include <xdc/runtime/System.h>

/* BIOS Module Headers */
#include <ti/sysbios/BIOS.h>
#include <ti/sysbios/knl/Task.h>
#include <ti/sysbios/knl/Semaphore.h>
#include <ti/sysbios/knl/Clock.h>

/* Driver Header files */
#include <ti/drivers/ADC.h>
#include <ti/drivers/UART.h>
#include <ti/drivers/GPIO.h>
#include <ti/drivers/PWM.h>

/* Board Header file */
#include <ti/drivers/Board.h>

/* Driver configuration */
#include "ti_drivers_config.h"

// Variables
uint16_t adcValue;
uint16_t count = 0;          // 1ms timer interrupt count

// Prototypes
void timer0Fxn();
void idleFxn();
void readADCFxn();
void displayUARTFxn();
void ledPWMFxn();
void hearbeatFxn();

/*
 *  ======== main ========
 */
int main() {
    /* Call driver init functions */
    Board_init();
    GPIO_init();
    UART_init();
    PWM_init();
    ADC_init();

    /* Configure the LED pin */
    GPIO_setConfig(CONFIG_GPIO_LED_0, GPIO_CFG_OUT_STD | GPIO_CFG_OUT_LOW);
    /* Turn on user LED */
    GPIO_write(CONFIG_GPIO_LED_0, CONFIG_GPIO_LED_ON);

    System_printf("Assignment 4\nUse the MKII's horizontal joy stick to adjust green
LEDs PWM value");

    // Kernal Start
```

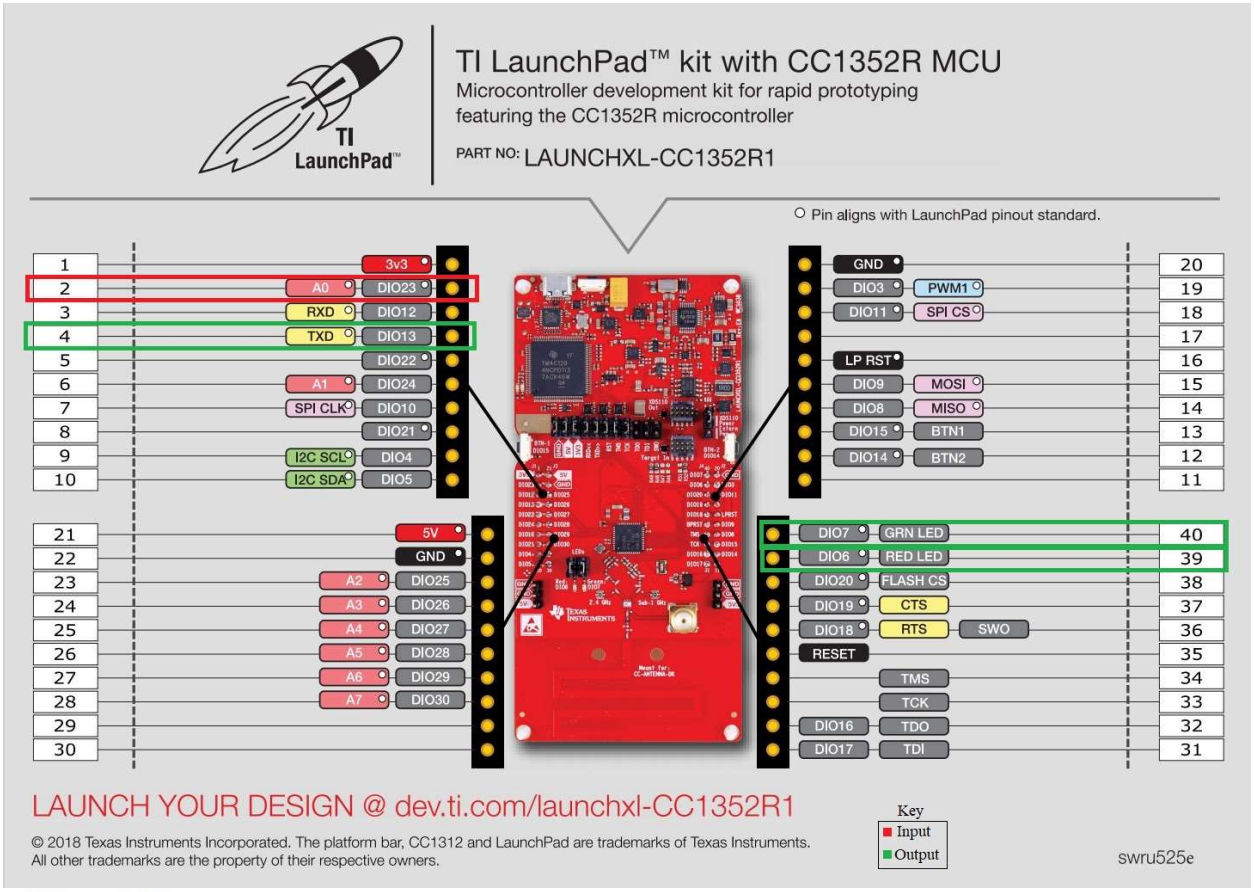
```
    BIOS_start();

    return(0);
}

void timer0Fxn() {
// Triggered every 1ms
    count++;
    if (count == 5)         // Run ADC
        Semaphore_post(adc_sem);
    else if (count == 10)   // Display ADC value on UART
        Semaphore_post(uart_sem);
    else if (count == 15) { // Update PWM value
        Semaphore_post(pwm_sem);
        count = 0;
    }
}

void workLoopFxn() {
// Main loop
    Timer_start(timer0);    // Start counting Timer
    while (1) {}
}

void readADCFxn() {
// Read in AD0's (horizontal joy stick) value
    // Create and initialize joy stick peripheral
    ADC_Handle adc;
    ADC_Params params;

    ADC_Params_init(&params);
    adc = ADC_open(CONFIG_ADC_0, &params);

    while(1) { // Dosen't check if conversion was successful
       ADC_convert(adc, &adcValue);
       Semaphore_pend(adc_sem, BIOS_WAIT_FOREVER);
    }
}

void displayUARTFxn() {
// Displays ADC value to terminal
    const char  prompt[] = "Console Entry:\r\n";
    UART_Handle uart;
    UART_Params uartParams;

    /* Create a UART with data processing off. */
    UART_Params_init(&uartParams);
    uartParams.writeDataMode = UART_DATA_BINARY;
    uartParams.readDataMode = UART_DATA_BINARY;
    uartParams.readReturnMode = UART_RETURN_FULL;
    uartParams.baudRate = 115200;

    uart = UART_open(CONFIG_UART_0, &uartParams);

    if (uart == NULL) {
```

```
        /* UART_open() failed */
        while (1);
    }

    UART_write(uart, prompt, sizeof(prompt));

    char adcValueStr[6];        // ADC value as C string
    uint32_t clearCount = 0;    // Count till clearing console

    while (1) { // Infinitely display ADC value
        sprintf(adcValueStr, "%d\r", adcValue);  // Convert int to string
        UART_write(uart, adcValueStr, sizeof(adcValueStr));
        if(clearCount == 60) {
            UART_write(uart, "     \r", sizeof(adcValueStr));  // Clear old value
            clearCount = 0;
        }
        clearCount++;
        Semaphore_pend(uart_sem, BIOS_WAIT_FOREVER);
    }
}
```

```
void ledPWMFxn() {
// Check if switch is pressed and update PWM duty cycle to LED
    PWM_Handle pwm;
    PWM_Params params;
    uint16_t   pwmPeriod = 3100;      // Period and duty in microseconds
    uint16_t   duty = 0;

    // Create and initialize PWM
    PWM_Params_init(&params);
    params.periodUnits = PWM_PERIOD_US;
    params.periodValue = pwmPeriod;
    params.dutyUnits = PWM_DUTY_US;
    pwm = PWM_open(CONFIG_PWM_0, &params);
    PWM_start(pwm);

    while (1) {
        duty = adcValue;    // 32bit to 16bit
        PWM_setDuty(pwm, duty);
        Semaphore_pend(pwm_sem, BIOS_WAIT_FOREVER);
    }
}
```

```
void hearbeatFxn() {
// Toggle Red LED every 1s
    uint32_t time = 1000000/Clock_tickPeriod;   // 1 second

    while (1) {
        Task_sleep(time);
        GPIO_toggle(CONFIG_GPIO_LED_0);
    }
}
```

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



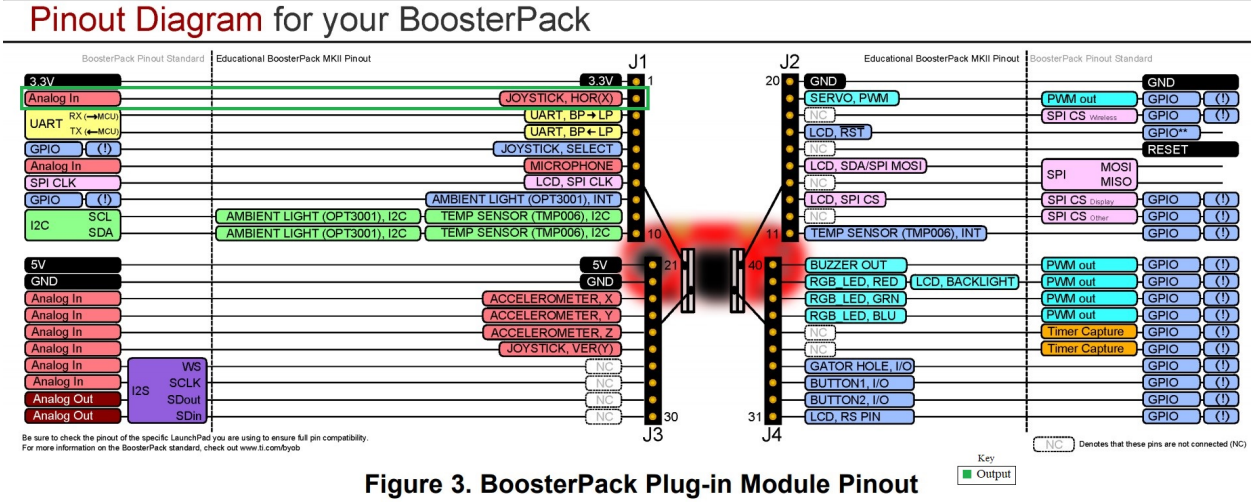
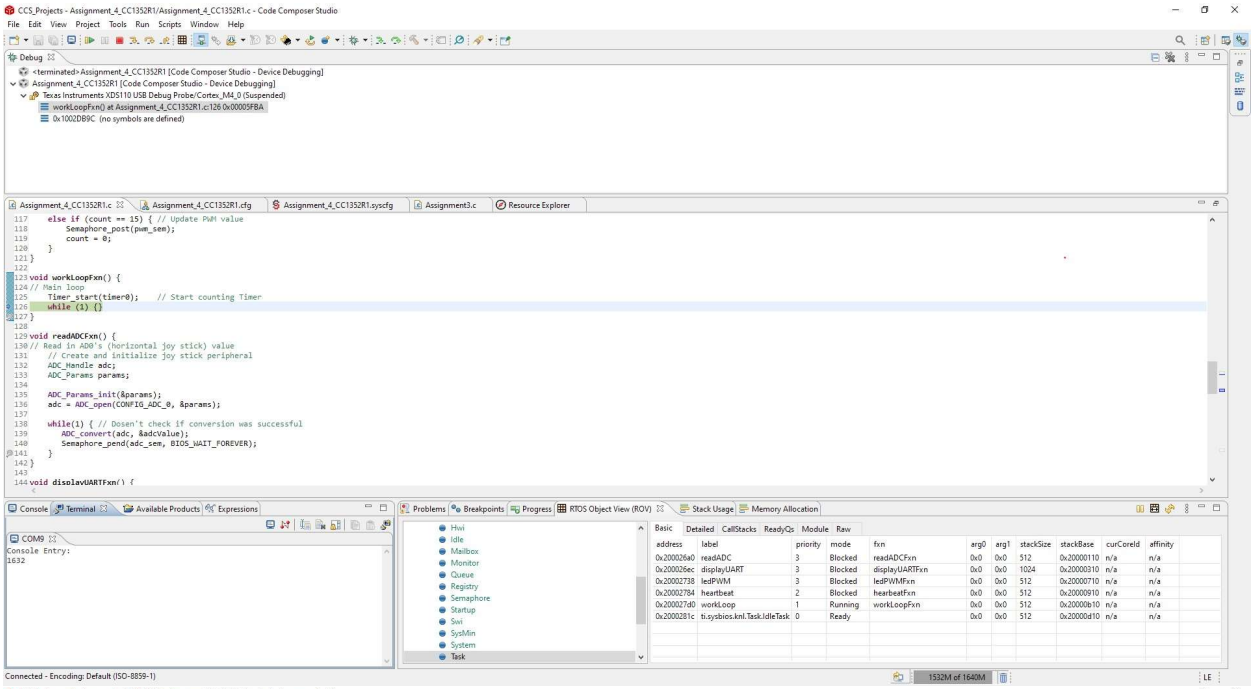
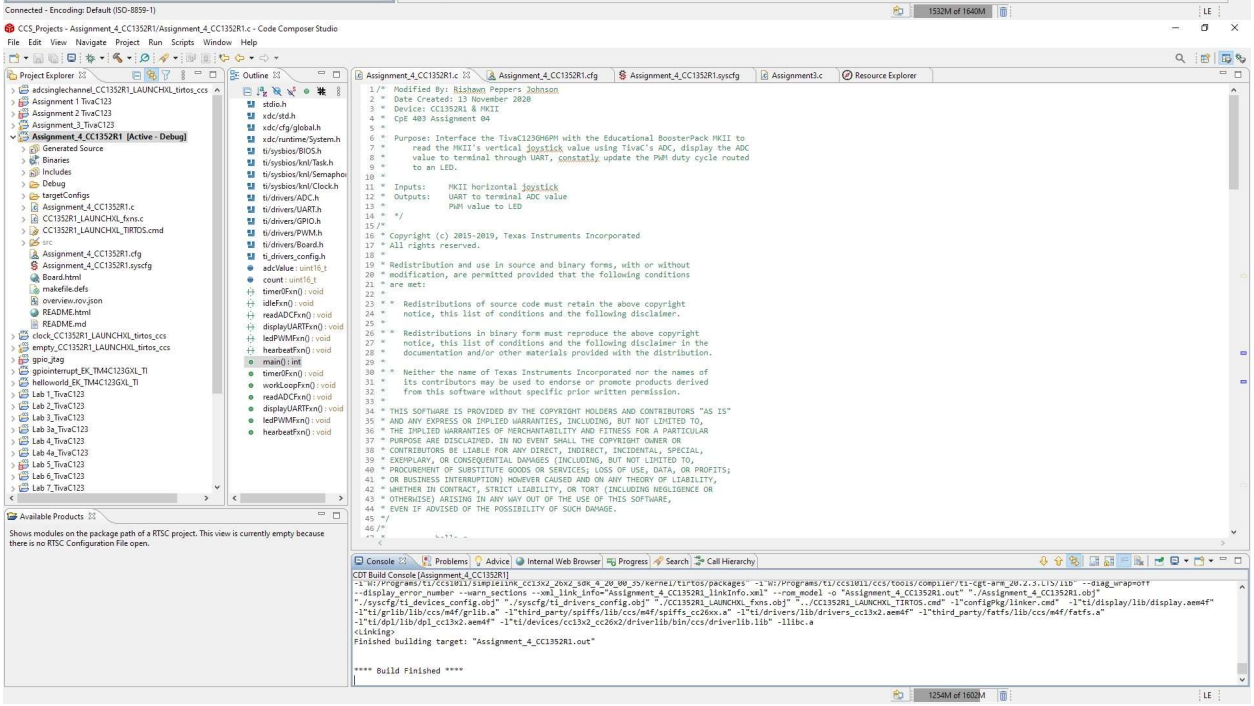
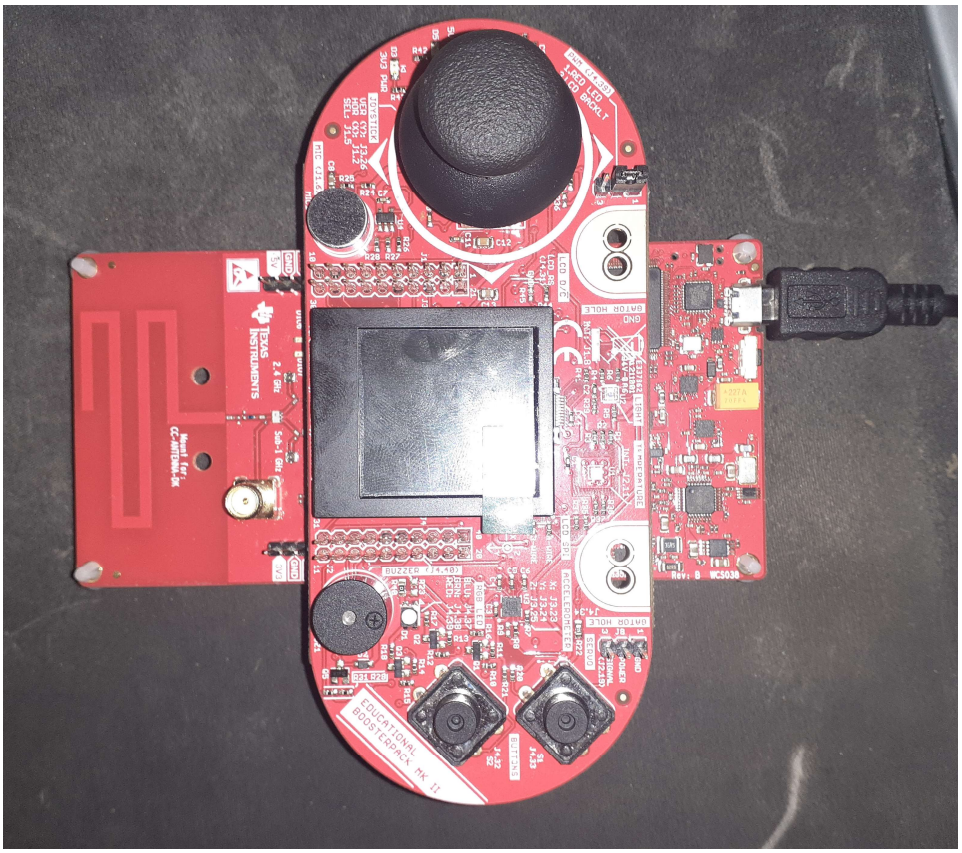


Figure 3. BoosterPack Plug-in Module Pinout

3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.







4. Declaration

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

"This assignment submission is my own, original work".
Rishawn Peppers Johnson