CPE403 – Advanced Embedded Systems

Design Assignment 04

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Do Le

Email: led2@unlv.nevada.edu

Github Repository link (root): https://github.com/DoVietLe/AES

Youtube Playlist link (root):
https://www.youtube.com/playlist?list=PLFfzhLPj7fvOz1lm2Vd9DevkHetoyvRQ6

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.

```
#include <stddef.h>
#include <stdint.h>

/* XDC Module Headers */
#include <xdc/std.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/GPIO.h>
#include <ti/drivers/UART.h>
#include <ti/drivers/Board.h>
#include <ti/drivers/Timer.h>
#include <ti/drivers/ADC.h>
#include <ti/drivers/PWM.h>

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

#define TASKSTACKSIZE   512

const char INTRO[] = "UART CHANNEL OPENED\n\r";
```

```
/* Stacks and Structs */
Task_Struct task0Struct, task1Struct, task2Struct, task3Struct;
Char task0Stack[TASKSTACKSIZE*2], task1Stack[TASKSTACKSIZE],
task2Stack[TASKSTACKSIZE], task3Stack[TASKSTACKSIZE];
Semaphore_Struct semStruct0, semStruct1, semStruct2;
Semaphore_Handle semUARTPost, semADCCon, semPWMUpdate;

/* Variables */
uint8_t timerCount = 0;
uint16_t adcVal;

/* Callbacks */
void timerCallback(Timer_Handle myHandle, int_fast16_t status);

/* Tasks */
void taskADCPublish(void);
void taskADCConvert(void);
void taskHeartbeat(void);
void taskPWM(void);

/*
 *  ======== main ========
 */
int main()
{
    /* Variables and handlers */
    Timer_Handle timer0;

    /* Construct BIOS Objects */
    Timer_Params params;
    Task_Params taskParams;
    Semaphore_Params semParams;

    /* Call driver init functions */
    //GPIO_init();
    UART_init();
    Timer_init();
    ADC_init();
    PWM_init();
    Board_init();

    System_printf("Initializing Program\n\r");

    // Sets up the LED.
    GPIO_setConfig(CONFIG_GPIO_LED_0, GPIO_CFG_OUT_STD | GPIO_CFG_OUT_LOW);
    GPIO_write(CONFIG_GPIO_LED_0, CONFIG_GPIO_LED_ON);

    // Timer Handler
    Timer_Params_init(&params);
    params.period = 1000;
    params.periodUnits = Timer_PERIOD_US;
    params.timerMode = Timer_CONTINUOUS_CALLBACK;
    params.timerCallback = timerCallback;
    timer0 = Timer_open(CONFIG_TIMER, &params);
    Timer_start(timer0);

    // UART Post Task
    Task_Params_init(&taskParams);
    taskParams.stackSize = TASKSTACKSIZE*2;
    taskParams.stack = &task0Stack;
    taskParams.priority = 1;
    Task_construct(&task0Struct, (Task_FuncPtr)taskADCPublish, &taskParams, NULL);
```

```
    // ADC Convert Task
    Task_Params_init(&taskParams);
    taskParams.stackSize = TASKSTACKSIZE;
    taskParams.stack = &task1Stack;
    taskParams.priority = 1;
    Task_construct(&task1Struct, (Task_FuncPtr)taskADCConvert, &taskParams, NULL);

    // LED Heartbeat Task
    Task_Params_init(&taskParams);
    taskParams.stackSize = TASKSTACKSIZE;
    taskParams.stack = &task2Stack;
    taskParams.priority = 1;
    Task_construct(&task2Struct, (Task_FuncPtr)taskHeartbeat, &taskParams, NULL);

    // PWM Task
    Task_Params_init(&taskParams);
    taskParams.stackSize = TASKSTACKSIZE;
    taskParams.stack = &task3Stack;
    taskParams.priority = 1;
    Task_construct(&task3Struct, (Task_FuncPtr)taskPWM, &taskParams, NULL);

    // UART Post Semaphore
    Semaphore_Params_init(&semParams);
    Semaphore_construct(&semStruct0, 1, &semParams);
    semUARTPost = Semaphore_handle(&semStruct0);

    // ADC Convert Semaphore
    Semaphore_Params_init(&semParams);
    Semaphore_construct(&semStruct1, 1, &semParams);
    semADCCon = Semaphore_handle(&semStruct1);

    // PWM Update Semaphore
    Semaphore_Params_init(&semParams);
    Semaphore_construct(&semStruct2, 1, &semParams);
    semPWMUpdate = Semaphore_handle(&semStruct2);

    BIOS_start();
    return(0);
}

/*
 *  ======== taskADCPublish ========
 */
void taskADCPublish() {
    /* Variables and handlers */
    UART_Handle uart;
    char buffer[25] = "Publish ADC\n\r";

    /* Construct BIOS Objects */
    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);

    UART_write(uart, INTRO, sizeof(INTRO));

//*
```

```
    while (1) {
        Semaphore_pend(semUARTPost, BIOS_WAIT_FOREVER);
        System_sprintf(buffer, "ADC Value: %d\n\r", adcVal);
        UART_write(uart, buffer, sizeof(buffer));
    }
//*/
}

/*
 *  ======== taskADCConvert ========
 */
void taskADCConvert() {
    /* Variables and handlers */
    ADC_Handle   adc;

    /* Construct BIOS Objects */
    ADC_Params   params;

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

//*
    while (1) {
        Semaphore_pend(semADCCon, BIOS_WAIT_FOREVER);
        ADC_convert(adc, &adcVal);
    }
//*/

}

/*
 *  ======== taskHeartbeat ========
 */
void taskHeartbeat() {
    uint32_t sleepTickCount = 1000000 / Clock_tickPeriod;

    while (1) {
        GPIO_toggle(CONFIG_GPIO_LED_0);
        Task_sleep(sleepTickCount);
    }
}

/*
 *  ======== taskPWM ========
 */
void taskPWM() {
    /* Variables and handlers */
    PWM_Handle pwm0;
    uint16_t period = 3000;
    uint16_t duty = 0;
    uint32_t sleepTickCount = 1000000 / Clock_tickPeriod;

    /* Construct BIOS Objects */
    PWM_Params params;

    PWM_Params_init(&params);
    params.dutyUnits = PWM_DUTY_US;
    params.dutyValue = duty;
    params.periodUnits = PWM_PERIOD_US;
    params.periodValue = period;
    pwm0 = PWM_open(CONFIG_PWM_0, &params);
    PWM_start(pwm0);
    PWM_setDuty(pwm0, duty);
```

```
    while (1) {
        Semaphore_pend(semPWMUpdate, BIOS_WAIT_FOREVER);
        duty = (adcVal*3000)/3070;
        PWM_setDuty(pwm0, duty);
    }
}


/*
 *  ======== interruptTimerHandler ========
 */
void timerCallback(Timer_Handle myHandle, int_fast16_t status) {
    if (timerCount == 5) {
        Semaphore_post(semADCCon);
    } else if (timerCount == 10) {
        Semaphore_post(semUARTPost);
    } else if (timerCount == 15) {
        Semaphore_post(semPWMUpdate);
        timerCount = 0;
    }
    timerCount++;
    //GPIO_toggle(CONFIG_GPIO_LED_0);
}
```

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

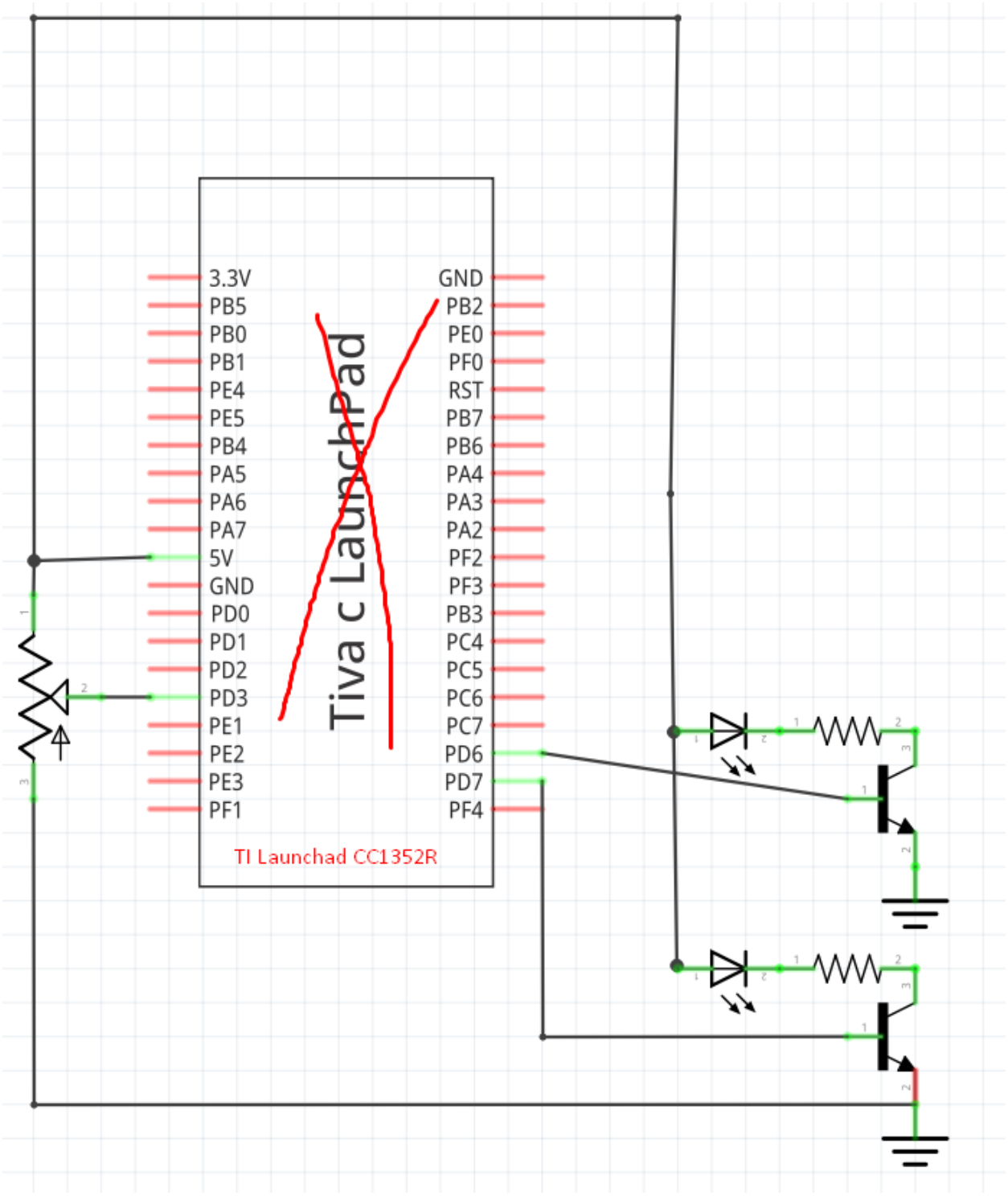


Figure 1: Schematic View

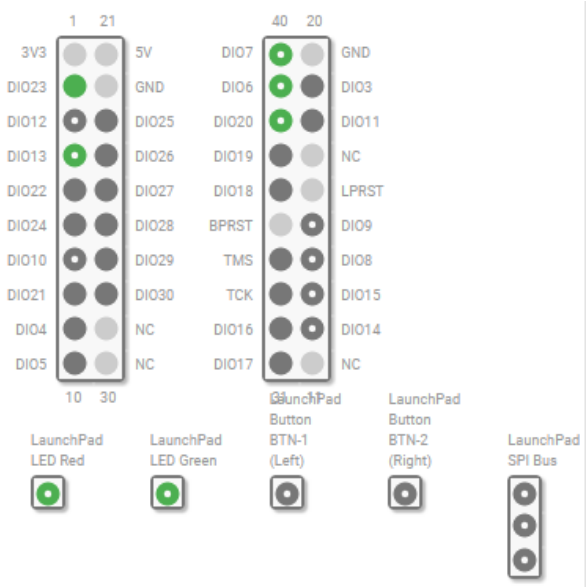
3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.

Figure 4: Physical Setup

```
Finished building target: "Assignment04.out"


**** Build Finished ****
```

Figure 3: Assignment Compilation



Pin Types	Available	Used
Free Pin	●	●
Attached To Software	◉	◉

Figure 2: Pinout

4. Declaration

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

"This assignment submission is my own, original work".
Do V. Le