CPE403 – Advanced Embedded Systems

Design Assignment #1

Name: Brian Wolak

Email: wolak@unlv.nevada.edu

GitHub Repository link: https://github.com/brianwolak/advanced_submissions

403 Youtube Playlist link: https://youtube.com/playlist?list=PLl6a3M--0IcYnZIMGyucOLe8c-16wAClN

C Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "inc/hw_ints.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "driverlib/debug.h"
#include "driverlib/adc.h"
#include "utils/uartstdio.h"
#include <string.h>
#include "driverlib/pwm.h"
#include "driverlib/rom_map.h"
#include "inc/hw_gpio.h"
#define PWM_FREQUENCY 85

// global variables
uint32_t ui32Period;
uint32_t ui32ADC0Value[4];
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;
char     buffer[4];


// timer 1 interrupt handler function
void Timer1IntHandler(void)
{
     // clear interrupt
     TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
     if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
```

```
        {
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
}
// start up function to display to terminal
void startup(void){
        // startup message
        UARTprintf("- - Welcome to the TM4C123GXL Launchpad! - -\n");
        UARTprintf("Command list: R, r, B, b, G, g, T, t, \n");
        UARTprintf("P, F or H will operate board functionalities \n");
        UARTprintf("Begin with 'H' to see the HELP MENU \n\n");
}
// temperature function for celsius
void tempC(){
    ADCProcessorTrigger(ADC0_BASE, 2);
    ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
    // temperature average calculation
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = ((1475 - ((2475 * ui32TempAvg)) / 4096)/10);
    // print temp to terminal
    UARTprintf("Current temp in Celcius: %3d\t",ui32TempValueC );
}
// temperature function for fahrenheit
void tempF(){
    ADCProcessorTrigger(ADC0_BASE, 2);
    ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
    // temp average calculations
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
    ui32TempValueF = (((ui32TempValueC * 9) + 160) / 5);
    // print temp to terminal
    UARTprintf("Current Fahrenheit temp: %3d\t",ui32TempValueF);
}
// help function to display to terminal
void help(){
    UARTprintf("\n\n--------------  -HELP MENU-  ----------------\n");
    UARTprintf(" Use the commands below to control your TM4C123GXL\n\n");
    UARTprintf(" 'R' to turn the Red LED ON \n");
    UARTprintf(" 'r' to turn the Red LED OFF \n");
    UARTprintf(" 'G' to turn the Green LED ON \n");
    UARTprintf(" 'g' to turn the Green LED OFF\n");
    UARTprintf(" 'B' to turn the Blue LED ON \n");
    UARTprintf(" 'b' to turn the Blue LED OFF \n");
    UARTprintf(" 'T' for current temp in Celsius \n");
    UARTprintf(" 't' for current temp in Fahrenheit \n");
    UARTprintf(" 'P' to fade/brighten the Red LED \n");
    UARTprintf(" 'F' to flash the Blue LED \n");
    UARTprintf(" 'O' to turn OFF all LEDs \n");
    UARTprintf(" 'H' to display this Help Menu \n\n");
```

```
}

int main(void) {
    // volatile variables
    volatile uint32_t ui32PWMClock;
    volatile uint32_t ui32Load;
    volatile uint8_t ui8Adjust;
    ui8Adjust = 83;

    // system clock configure
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);

    // configure GPIO, UART and Timer1 peripherals
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);

    // ADC configuration
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 32);
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 2);

    // UART configure pins
    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);

    // GPIO portf pins set
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    ADCSequenceEnable(ADC0_BASE, 2);

    startup();

    unsigned char input;
    int switcher;

    while (1){
        input = UARTgetc();
        switch(input){
        case 82 : // "R"
            UARTprintf("turning ON Red LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0xFF);
            break;
        case 114 : // "r"
```

```
        UARTprintf("turning OFF Red LED \n");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
        break;
    case 71 : // "G"
        UARTprintf("turning ON Green LED \n");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0xFF);
        break;
    case 103 : // "g"
        UARTprintf("turning OFF Green LED \n");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0);
        break;
    case 66 : // "B"
        UARTprintf("turning ON Blue LED \n");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xFF);
        break;
    case 98 : // "b"
        UARTprintf("turning OFF Blue LED \n");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
        break;
    case 84 : // "T"
        UARTprintf("getting current temperature.. \n");
        tempC();
        UARTprintf("\n");
        break;
    case 79 : // "O"
        UARTprintf("turning OFF all LEDs \n");
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        break;
    case 116 : // "t"
        UARTprintf("getting current temperature.. \n");
        tempF();
        UARTprintf("\n");
        break;
    case 80 : // "P"
        UARTprintf("fading Red LED \n");
        //PWM and clock configuration
        GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
        GPIOPinConfigure(GPIO_PF1_M1PWM5);
        ui32PWMClock = SysCtlClockGet() / 64;
        ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
        // PWM generate setup
        PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
        PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
        // set pulse width
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 1000);
        // change blue led output state
        PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
        // enable PWM 1 generate 2
        PWMGenEnable(PWM1_BASE, PWM_GEN_2);
        // switching value to reset LED fader
        switcher = 1;
        while(!UARTCharsAvail(UART0_BASE)){ // wait here until keystroke is seen
            if(switcher == 1){
                ui8Adjust = ui8Adjust + 1;
```

```
                    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
                    SysCtlDelay(100000);
                    if(ui8Adjust >= 10000)
                        switcher = 0;
                }
                else {
                    ui8Adjust = ui8Adjust - 1;
                    SysCtlDelay(100000);
                    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
                    if(ui8Adjust <= 0)
                        switcher = 1;
                }
            }
            // return red LED to GPIO here on exit of fade sequence
            GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
            UARTprintf("turning OFF Red LED \n");
            break;
        case 70 : // "F"
            UARTprintf("flashing Blue LED \n");
            // timer 1 setup
            TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
            ui32Period = SysCtlClockGet()/2;   // 2Hz frequency
            TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);
            // enable timer 1 interrupt
            IntEnable(INT_TIMER1A);
            TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
            IntMasterEnable();
            // start timer 1
            TimerEnable(TIMER1_BASE, TIMER_A);
            while(!UARTCharsAvail(UART0_BASE)){} // wait here for next key stroke
            // disable timer 1 interrupt
            TimerIntDisable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
            // turn blue LED off
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x00);
            UARTprintf("turning OFF Blue LED \n");
            break;
        case 72 : // "H"
            UARTprintf("help screen loading.. \n");
            help();
            break;
        default: // default for any keystroke not represented above
            UARTprintf("INVALID COMMAND! - SEE HELP MENU \n");
            break;
        }
    }

}
```

---------- INDIVIDUAL TASK SNIPS ----------

Turn on Red LED, “R”:

```
case 82 : // "R"
            UARTprintf("turning ON Red LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0xFF);
            break;
```

Turn off Red LED, “r” input:

```
case 114 : // "r"
            UARTprintf("turning OFF Red LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
            break;
```

Turn on Blue LED, “B” input:

```
case 66 : // "B"
            UARTprintf("turning ON Blue LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xFF);
            break;
```

Turn off Blue LED, “b” input:

```
case 98 : // "b"
            UARTprintf("turning OFF Blue LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
            break;
```

Turn on Green LED, “G” input:

```
case 71 : // "G"
            UARTprintf("turning ON Green LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0xFF);
            break;
```

Turn off Green LED, “g” input:

```
case 103 : // "g"
            UARTprintf("turning OFF Green LED \n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0);
            break;
```

Fade Red LED, “P” input:

```
case 80 : // "P"
            UARTprintf("fading Red LED \n");
            //PWM and clock configuration
            GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
            GPIOPinConfigure(GPIO_PF1_M1PWM5);
            ui32PWMClock = SysCtlClockGet() / 64;
            ui32Load = (ui32PWMClock / PWM_FREQUENCY) - 1;
            // PWM generate setup
            PWMGenConfigure(PWM1_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN);
            PWMGenPeriodSet(PWM1_BASE, PWM_GEN_2, ui32Load);
            // set pulse width
            PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load / 1000);
            // change blue led output state
            PWMOutputState(PWM1_BASE, PWM_OUT_5_BIT, true);
            // enable PWM 1 generate 2
            PWMGenEnable(PWM1_BASE, PWM_GEN_2);
            // switching value to reset LED fader
            switcher = 1;
            while(!UARTCharsAvail(UART0_BASE)){ // wait here until keystroke is seen
                if(switcher == 1){
                    ui8Adjust = ui8Adjust + 1;
                    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
                    SysCtlDelay(100000);
                    if(ui8Adjust >= 10000)
                        switcher = 0;
                }
                else {
                    ui8Adjust = ui8Adjust - 1;
                    SysCtlDelay(100000);
                    PWMPulseWidthSet(PWM1_BASE, PWM_OUT_5, ui8Adjust * ui32Load /
1000);
                    if(ui8Adjust <= 0)
                        switcher = 1;
                }
            }
            // return red LED to GPIO here on exit of fade sequence
            GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
            UARTprintf("turning OFF Red LED \n");
            break;
```

Flash Blue LED, “F” input:

```
case 70 : // "F"
            UARTprintf("flashing Blue LED \n");
            // timer 1 setup
            TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
            ui32Period = SysCtlClockGet()/2;   // 2Hz frequency
            TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);
            // enable timer 1 interrupt
            IntEnable(INT_TIMER1A);
            TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
```

```
            IntMasterEnable();
            // start timer 1
            TimerEnable(TIMER1_BASE, TIMER_A);
            while(!UARTCharsAvail(UART0_BASE)){} // wait here for next key stroke
            // disable timer 1 interrupt
            TimerIntDisable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
            // turn blue LED off
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x00);
            UARTprintf("turning OFF Blue LED \n");
            break;
```

Display C^0 Temp, "T" input:

```
// temperature function for celsius
void tempC(){
    ADCProcessorTrigger(ADC0_BASE, 2);
    ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
    // temperature average calculation
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = ((1475 - ((2475 * ui32TempAvg)) / 4096)/10);
    // print temp to terminal
    UARTprintf("Current temp in Celcius: %3d\t",ui32TempValueC );
}
--------------------------------
case 84 : // "T"
            UARTprintf("getting current temperature.. \n");
            tempC();
            UARTprintf("\n");
            break;
```

Display F^0 Temp, "t" input:

```
// temperature function for fahrenheit
void tempF(){
    ADCProcessorTrigger(ADC0_BASE, 2);
    ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
    // temp average calculations
    ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] +
ui32ADC0Value[3] + 2)/4;
    ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
    ui32TempValueF = (((ui32TempValueC * 9) + 160) / 5);
    // print temp to terminal
    UARTprintf("Current Fahrenheit temp: %3d\t",ui32TempValueF);
}
--------------------------------
case 116 : // "t"
            UARTprintf("getting current temperature.. \n");
            tempF();
            UARTprintf("\n");
            break;
```

Display help menu, “H” input:

```
// help function to display to terminal
void help(){
    UARTprintf("\n\n--------------  -HELP MENU-  ----------------\n");
    UARTprintf(" Use the commands below to control your TM4C123GXL\n\n");
    UARTprintf(" 'R' to turn the Red LED ON \n");
    UARTprintf(" 'r' to turn the Red LED OFF \n");
    UARTprintf(" 'G' to turn the Green LED ON \n");
    UARTprintf(" 'g' to turn the Green LED OFF\n");
    UARTprintf(" 'B' to turn the Blue LED ON \n");
    UARTprintf(" 'b' to turn the Blue LED OFF \n");
    UARTprintf(" 'T' for current temp in Celsius \n");
    UARTprintf(" 't' for current temp in Fahrenheit \n");
    UARTprintf(" 'P' to fade/brighten the Red LED \n");
    UARTprintf(" 'F' to flash the Blue LED \n");
    UARTprintf(" 'O' to turn OFF all LEDs \n");
    UARTprintf(" 'H' to display this Help Menu \n\n");
}
-------------------------------
case 72 : // "H"
            UARTprintf("help screen loading.. \n");
            help();
            break;
```

Video and GitHub Links:

GitHub: https://github.com/brianwolak/advanced_submissions/tree/main/DA_1

YouTube: https://youtu.be/iN7lp_8VYuE

“This assignment submission is my own, original work”.
Brian Wolak