CPE403 – Advanced Embedded Systems

Design Assignment #02

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Cameron Kirk

Email: kirkc1@unlv.nevada.edu

Github Repository link (root): https://github.com/kirkster96/DqF514-not-embedded

Youtube Playlist link (root):

https://www.youtube.com/playlist?list=PLiqmqQ7XKuf7ArV7lO6b20D1ES5SUp0Yk

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.

Task 1

```
#include <stdarg.h>
#include <stdbool.h>
#include <stdint.h>
#include "inc/hw_i2c.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "driverlib/i2c.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/uart.h"
#include "driverlib/interrupt.h"

#include "hw_tmp006.h"
#include "tmp006.h"

#include "driverlib/debug.h"
#include "utils/uartstdio.h"
#include <string.h>
#include <math.h>
//#include"IQmath/IQmathLib.h"



void ConfigureUART(void);
void init_I2C1(void);
void write16_I2C1(uint8_t slave_addr, uint8_t pointer_reg, uint16_t TxData);
uint16_t read16_I2C1(uint8_t slave_addr, uint8_t pointer_reg);
void init_tmp006(void);
double GetTemp(void);

volatile long double Tobj;


int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
                           SYSCTL_OSC_MAIN);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    double i, temp=0, tempAve=0;
    uint16_t convertA, convertB;
    ConfigureUART();
    init_I2C1();
    UARTprintf("Starting TMP006 Initialization ..... \n");
```

```
    init_tmp006();
    while(1){
        for (i=0;i<20;i++)
        {
            temp = GetTemp();
            tempAve += temp;
        }
        tempAve = tempAve/20;
        convertA = tempAve;
        tempAve = tempAve * 1000;
        convertB = tempAve - (convertA * 1000);

        UARTprintf("Temperature Value %3d", convertA);
        UARTprintf(". %3d\n", convertB);


        SysCtlDelay(5000000);
        tempAve = 0;

    }
}



void init_I2C1()
{
    SysCtlPeripheralEnable (SYSCTL_PERIPH_I2C1);    //enables I2C1
    SysCtlDelay(3);

    SysCtlPeripheralEnable (SYSCTL_PERIPH_GPIOA);   //enables PORTA as peripheral
    SysCtlDelay(3);
    GPIOPinConfigure (GPIO_PA7_I2C1SDA);
    GPIOPinConfigure(GPIO_PA6_I2C1SCL);

    GPIOPinTypeI2C(GPIO_PORTA_BASE, GPIO_PIN_7);    //set I2C PA7 as SDA
    GPIOPinTypeI2CSCL(GPIO_PORTA_BASE, GPIO_PIN_6);    //set I2C PA6 as SCLK

    I2CMasterInitExpClk (I2C1_BASE, SysCtlClockGet(), true);   //set  the clock of
the I2C to ensure proper connection


    HWREG(I2C1_BASE + I2C_O_FIFOCTL) = 80008000;    //clear I2C FIFOs
}

void write16_I2C1(uint8_t slave_addr, uint8_t pointer_reg, uint16_t TxData)
{
    uint8_t data;
    I2CMasterSlaveAddrSet (I2C1_BASE, slave_addr, true); //Find the device based on
the address given
    I2CMasterDataPut (I2C1_BASE, pointer_reg);  //put the first argument in the list
in to the I2C bus
    I2CMasterControl(I2C1_BASE, I2C_MASTER_CMD_BURST_SEND_START);
    while (I2CMasterBusy (I2C1_BASE));
    //MSB First
    data = (uint8_t)((TxData >> 8) & 0x00FF);
```

```
    I2CMasterDataPut(I2C1_BASE, data);
    while (I2CMasterBusy (I2C1 BASE));
    //LSB Second
    data = (uint8_t)(TxData & 0x00FF);
    I2CMasterDataPut(I2C1_BASE, data);
    I2CMasterControl(I2C1_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
    while (I2CMasterBusy (I2C1 BASE));

}


uint16_t read16_I2C1(uint8_t slave_addr, uint8_t pointer_reg)
{
    uint8_t data;
    uint16_t RxData;

    I2CMasterSlaveAddrSet(I2C1_BASE, slave_addr, false);   //set the master to read
from the device
    I2CMasterDataPut(I2C1_BASE, pointer_reg);
    I2CMasterControl(I2C1_BASE, I2C_MASTER_CMD_SINGLE_SEND);

    while(I2CMasterBusy(I2C1 BASE));
    // Set read mode
    I2CMasterSlaveAddrSet(I2C1_BASE, slave_addr, true);
    // Get first byte from slave and ackfor more
    I2CMasterControl(I2C1_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);
    while(I2CMasterBusy(I2C1 BASE));
    data = I2CMasterDataGet(I2C1 BASE);
    RxData = (uint16_t) (data<<8);
    // Get second byte from slave and nackfor complete
    I2CMasterControl(I2C1_BASE,  I2C_MASTER_CMD_BURST_RECEIVE_CONT);
    while(I2CMasterBusy(I2C1 BASE));
    data = I2CMasterDataGet(I2C1 BASE);
    RxData |= (uint16_t) data;
    return RxData;
}
void init_tmp006()
{
    uint16_t x;

    x = read16_I2C1(TMP006_SLAVE_ADDRESS, TMP006_O_DEV_ID);

    /* Specify slave address for TMP006 */
    UARTprintf("device id = %3d\n",x);
    if(x!=0x67)
    {
        UARTprintf("TMP006_O_DEV_ID = 0x67. Invalid device ID.");
        while(1)
        {

        }

    }

    /* Reset TMP006 */
```

```
    write16_I2C1 (TMP006_O_CONFIG, 1, (TMP006_CONFIG_RESET_M|TMP006_CONFIG_MODE_M));

    volatile int i;
    for (i=10000; i>0;i--);

    /* Power-up and re-enable device */
    write16_I2C1(TMP006_SLAVE_ADDRESS, 1, TMP006_CONFIG_MODE_CONT |
TMP006_CONFIG_CR_2);
}

double GetTemp()
{
    int16_t Vobj = 0;
    int16_t Tdie = 0;
    static long double S0 = 0;


    /* Read the object voltage */
    Vobj = read16_I2C1(TMP006_SLAVE_ADDRESS, TMP006_O_VOBJECT);

    /* Read the ambient temperature */
    Tdie = read16_I2C1(TMP006_SLAVE_ADDRESS, TMP006_O_TAMBIENT);
    Tdie = Tdie >> 2;

    /* Calculate TMP006. This needs to be reviewed and calibrated */

    long double Vobj2 = (double)Vobj*.00000015625;

    long double Tdie2 = (double)Tdie*.03525 + 273.15;

    /* Initialize constants */
    S0 = 6 * pow(10, -14);


    long double a1 = 1.75*pow(10, -3);


    long double a2 = -1.678*pow(10, -5);


    long double b0 = -2.94*pow(10, -5);

    long double b1 = -5.7*pow(10, -7);


    long double b2 = 4.63*pow(10, -9);


    long double c2 = 13.4;


    long double Tref = 298.15;

    /* Calculate values */
    long double S = S0*(1+a1*(Tdie2 - Tref)+a2*pow((Tdie2 - Tref),2));
```

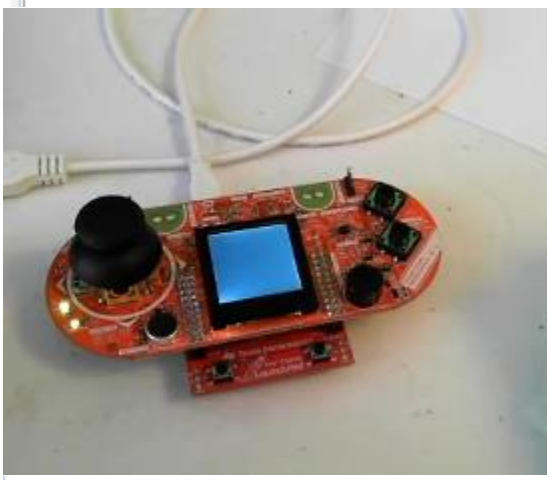
```
    long double Vos = b0 + b1*(Tdie2 - Tref) + b2*pow((Tdie2 - Tref),2);

    volatile long double fObj = (Vobj2 - Vos) + c2*pow((Vobj2 - Vos),2);

    Tobj = pow(pow(Tdie2,4) + (fObj/S),.25);
    Tobj = (9.0/5.0)*(Tobj - 273.15) + 32;
    //Tobj = (Tobj - 273.15);


    /* Return temperature of object */
    return (Tobj);
}


//*****************************************************************************
//
// Configure the UART and its pins.  This must be called before UARTprintf().
//
//*****************************************************************************
void
ConfigureUART(void)
{
    //
    // Enable the GPIO Peripheral used by the UART.
    //
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    //
    // Enable UART0
    //
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

    //
    // Configure GPIO Pins for UART mode.
    //
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    //
    // Use the internal 16MHz oscillator as the UART clock source.
    //
    UARTClockSourceSet(UART0_BASE, UART_CLOCK_PIOSC);

    //
    // Initialize the UART for console I/O.
    //
    UARTStdioConfig(0, 115200, 16000000);
}
```

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

Educational BoosterPack MKII Launchpad configuration with the Tiva C 123. I2C1 is used on pin J1.9 and J1.10

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

```
Temperature Value  75.  32
Temperature Value  74. 996
Temperature Value  74. 987
Temperature Value  74. 987
Temperature Value  74. 494
```



CDT Build Console [asng2_t01_TIVAC123g]

```
/IQmathLib.lib"
<Linking>
remark #10371-D: (ULP 1.1) Detected no uses of low power mode
state changing instructions
Finished building target: "asng2_t01_TIVAC123g.out"

"C:/ti/ccs1010/ccs/utils/tiobj2bin/tiobj2bin"
"asng2_t01_TIVAC123g.out" "asng2_t01_TIVAC123g.bin"
"C:/ti/ccs1010/ccs/tools/compiler/ti-cgt-arm_20.2.1.LTS/bin/a
ofd"
"C:/ti/ccs1010/ccs/tools/compiler/ti-cgt-arm_20.2.1.LTS/bin/a
hex" "C:/ti/ccs1010/ccs/utils/tiobj2bin/mkhex4bin"


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

4. Declaration

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

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