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.
8. 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.

//

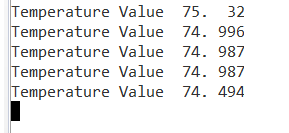
**UARTStdioConfig**(0, 115200, 16000000);

}

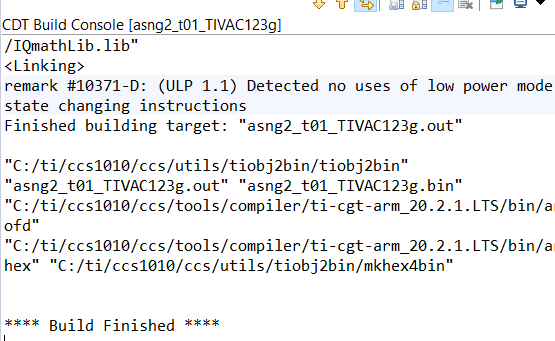
1. Block diagram and/or Schematics showing the components, pins used, and interface.

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

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







1. 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