CPE403 – Advanced Embedded Systems

# Design Assignment 2

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Joe Uesato

Email: uesatj1@unlv.nevada.edu

Github Repository link (root): https://github.com/joeuesato/lab\_submissions

Youtube Playlist: https://www.youtube.com/playlist?list=PLSBOvuRedzOf8JAhpVx0VsSteisJQUKv3

**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.

#include <stdbool.h>

#include <stdint.h>

#include <stdlib.h>

#include <stdio.h>

#include <stdarg.h>

#include <stdbool.h>

#include "inc/hw\_ints.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_sysctl.h"

#include "inc/hw\_types.h"

#include "inc/hw\_i2c.h"

#include "inc/hw\_types.h"

#include "inc/hw\_gpio.h"

#include "driverlib/gpio.h"

#include "driverlib/pin\_map.h"

#include "driverlib/rom.h"

#include "driverlib/rom\_map.h"

#include "driverlib/debug.h"

#include "driverlib/interrupt.h"

#include "driverlib/i2c.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "sensorlib/i2cm\_drv.h"

#include "sensorlib/hw\_tmp006.h"

#include "sensorlib/tmp006.h"

#include "uartstdio.h"

#include "math.h"

#include "IQmath/IQmathLib.h"

#define TMP006\_I2C\_ADDRESS 0x41

// Instance of drivers

tI2CMInstance g\_sI2CInst;

tTMP006 g\_sTMP006Inst;

// Variables

volatile uint\_fast8\_t g\_vui8DataFlag; // Global new data flag to alert main that TMP006 data is ready.

volatile uint\_fast8\_t g\_vui8ErrorFlag; // Global new error flag to store the error condition if encountered.

// Interrupt Handlers

void TMP006AppCallback(void \*pvCallbackData, uint\_fast8\_t ui8Status);

// Other functions

void ConfigureUART(void);

void InitI2C0(void);

int main(void) {

float fAmbient, fObject;

double IQ\_tempFarenheit;

int\_fast32\_t i32IntegerPart;

int\_fast32\_t i32FractionPart;

SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

InitI2C0();

ConfigureUART();

TMP006Init(&g\_sTMP006Inst, &g\_sI2CInst, TMP006\_I2C\_ADDRESS, TMP006AppCallback, &g\_sTMP006Inst);

I2CMInit(&g\_sI2CInst, I2C0\_BASE, INT\_I2C0, 0xff, 0xff, SysCtlClockGet());

UARTprintf("Assignment 2!!\n");

while (1)

{

// Reset the flag

g\_vui8DataFlag = 0;

// Get a local copy of the latest data in float format.

TMP006DataTemperatureGetFloat(&g\_sTMP006Inst, &fAmbient, &fObject);

// Convert the floating point object temperature to an integer part and fraction part for easy printing.

i32IntegerPart = (int32\_t)fObject;

i32FractionPart = (int32\_t)(fObject \* 1000.0f);

i32FractionPart = i32FractionPart - (i32IntegerPart \* 1000);

if(i32FractionPart < 0)

{

i32FractionPart \*= -1;

}

UARTprintf("Object Temperature Celcius: %3d.%03d\n", i32IntegerPart, i32FractionPart);

// Convert to Farenheit

fObject = ((fObject \* 9) + 160) / 5;

// Convert the floating point ambient temperature to an integer part and fraction part for easy printing.

i32IntegerPart = (int32\_t)fObject;

i32FractionPart = (int32\_t)(fObject \* 1000.0f);

i32FractionPart = i32FractionPart - (i32IntegerPart \* 1000);

if(i32FractionPart < 0)

{

i32FractionPart \*= -1;

}

UARTprintf("Object Temperature Farenheit: %3d.%03d\n", i32IntegerPart, i32FractionPart);

SysCtlDelay(10000000);

}

}

void ConfigureUART(void) {

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA); // Enable the GPIO Peripheral used by the UART.

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0); // Enable UART0

GPIOPinConfigure(GPIO\_PA0\_U0RX); // Configure GPIO Pins for UART mode.

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);

UARTStdioConfig(0, 115200, 16000000);

}

void InitI2C0(void) {

SysCtlPeripheralEnable (SYSCTL\_PERIPH\_I2C0); //enables I2C0

SysCtlPeripheralEnable (SYSCTL\_PERIPH\_GPIOB); //enable PORTB as peripheral

GPIOPinTypeI2C (GPIO\_PORTB\_BASE, GPIO\_PIN\_3); //set I2C PB3 as SDA

GPIOPinConfigure (GPIO\_PB3\_I2C0SDA);

GPIOPinTypeI2CSCL (GPIO\_PORTB\_BASE, GPIO\_PIN\_2); //set I2C PB2 as SCLK

GPIOPinConfigure (GPIO\_PB2\_I2C0SCL);

I2CMasterInitExpClk (I2C0\_BASE, SysCtlClockGet(), false); //Set the clock of the I2C to ensure proper connection

while (I2CMasterBusy (I2C0\_BASE)); //wait while the master SDA is busy

}

void TMP006AppCallback(void \*pvCallbackData, uint\_fast8\_t ui8Status)

{

// If the transaction succeeded set the data flag to indicate to

// application that this transaction is complete and data may be ready.

if(ui8Status == I2CM\_STATUS\_SUCCESS)

{

g\_vui8DataFlag = 1;

}

// Store the most recent status in case it was an error condition

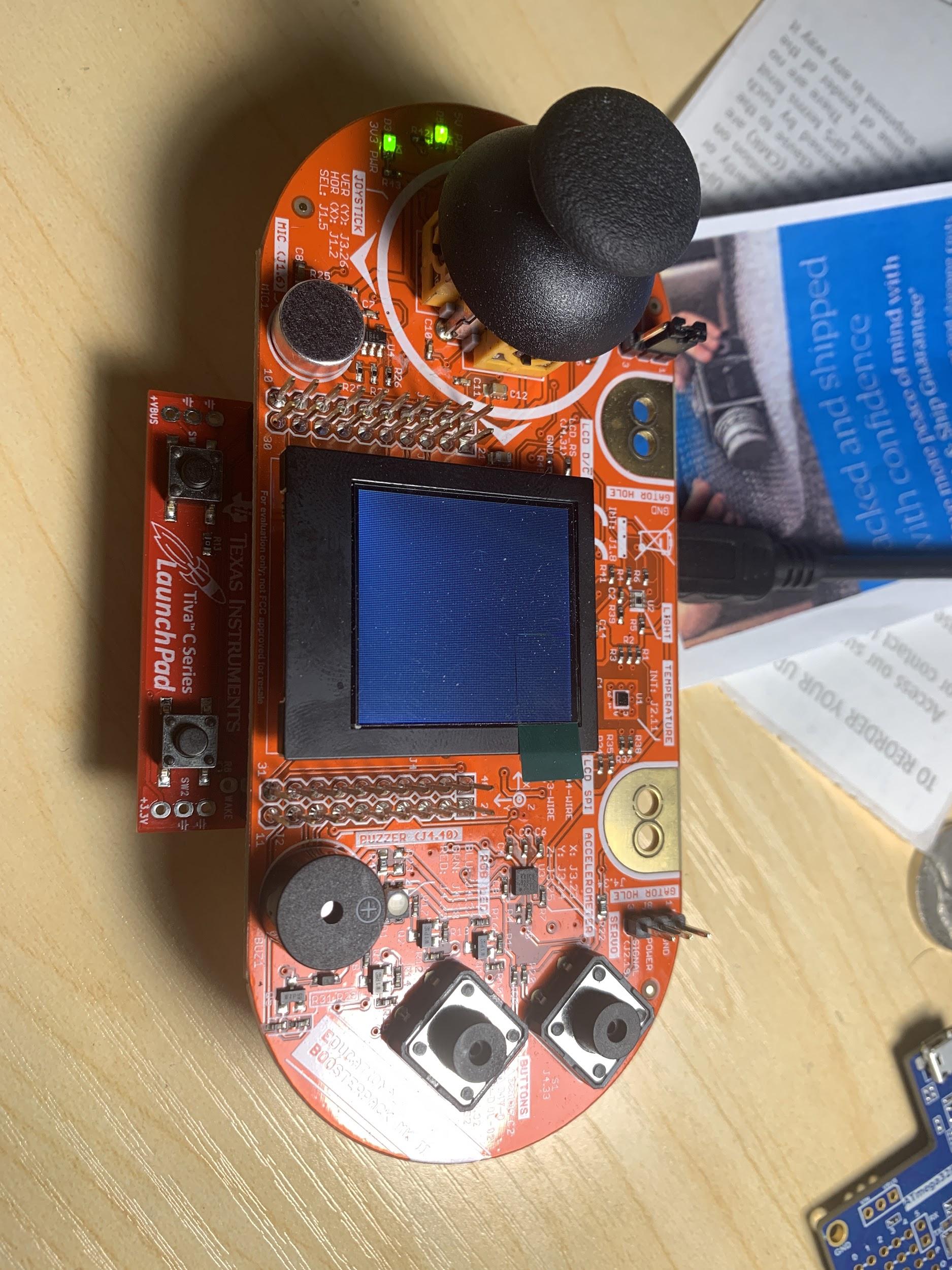
g\_vui8ErrorFlag = ui8Status;

}

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

Educational Boosterpack MKII plugged directly into the TIVAC, no other connections.

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”.

Joe Uesato