TivaC Lab 11

CPE 403

**Checklist for Lab 11**

* A text/word document of the initial code with comments
* In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also include the comments.
* Provide a permanent link to all main and dependent source code files only (name them as LabXX-TYY, XX-Lab# and YY-task#)Screenshots of debugging process along with pictures of actual circuit
* Video link of demonstration.

**Code for Experiment**

**Task 1:**

**Temperature Sensor:**

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "LaunchPad.h"

**#include** "OrbitBoosterPackDefs.h"

**#include** "OrbitOled.h"

**#include** "OrbitOledGrph.h"

**#include** "OrbitOledChar.h"

**#include** "FillPat.h"

**#include** "delay.h"

**void** **DeviceInit**();

**void** **OrbitSetOled**();

**void** **OrbitDemo**();

**void** **Read\_temp**(**unsigned** **char**\*); // Read Temperature sensor

/\* ------------------------------------------------------------ \*/

/\*\*\* main()

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Main program loop

\*/

**int** **main**() {

DeviceInit();

**while**(1) {

OrbitDemo();

}

//return 0;

}

/\* ------------------------------------------------------------ \*/

/\*\*\* DeviceInit

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Initialize I2C Communication, and GPIO

\*/

**void** **DeviceInit**(**void**) {

/\*

\* First, Set Up the Clock.

\* Main OSC -> SYSCTL\_OSC\_MAIN

\* Runs off 16MHz clock -> SYSCTL\_XTAL\_16MHZ

\* Use PLL -> SYSCTL\_USE\_PLL

\* Divide by 4 -> SYSCTL\_SYSDIV\_4

\*/

**SysCtlClockSet**(SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ | SYSCTL\_USE\_PLL | SYSCTL\_SYSDIV\_4);

/\*

\* Enable and Power On All GPIO Ports

\*/

**SysCtlPeripheralEnable**( SYSCTL\_PERIPH\_GPIOA);

**SysCtlPeripheralEnable**( SYSCTL\_PERIPH\_GPIOB);

**SysCtlPeripheralEnable**( SYSCTL\_PERIPH\_GPIOC);

**SysCtlPeripheralEnable**( SYSCTL\_PERIPH\_GPIOD);

**SysCtlPeripheralEnable**( SYSCTL\_PERIPH\_GPIOE);

**SysCtlPeripheralEnable**( SYSCTL\_PERIPH\_GPIOF);

/\*

\* Pad Configure.. Setting as per the Button Pullups on

\* the Launch pad (active low).. changing to pulldowns for Orbit

\*/

**GPIOPadConfigSet**(SWTPort, SWT1 | SWT2, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPD);

**GPIOPadConfigSet**(BTN1Port, BTN1, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPD);

**GPIOPadConfigSet**(BTN2Port, BTN2, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPD);

**GPIOPadConfigSet**(LED1Port, LED1, GPIO\_STRENGTH\_8MA\_SC, GPIO\_PIN\_TYPE\_STD);

**GPIOPadConfigSet**(LED2Port, LED2, GPIO\_STRENGTH\_8MA\_SC, GPIO\_PIN\_TYPE\_STD);

**GPIOPadConfigSet**(LED3Port, LED3, GPIO\_STRENGTH\_8MA\_SC, GPIO\_PIN\_TYPE\_STD);

**GPIOPadConfigSet**(LED4Port, LED4, GPIO\_STRENGTH\_8MA\_SC, GPIO\_PIN\_TYPE\_STD);

/\*

\* Initialize Switches as Input

\*/

**GPIOPinTypeGPIOInput**(SWTPort, SWT1 | SWT2);

/\*

\* Initialize Buttons as Input

\*/

**GPIOPinTypeGPIOInput**(BTN1Port, BTN1);

**GPIOPinTypeGPIOInput**(BTN2Port, BTN2);

/\*

\* Initialize LEDs as Output

\*/

**GPIOPinTypeGPIOOutput**(LED1Port, LED1);

**GPIOPinTypeGPIOOutput**(LED2Port, LED2);

**GPIOPinTypeGPIOOutput**(LED3Port, LED3);

**GPIOPinTypeGPIOOutput**(LED4Port, LED4);

/\*

\* Enable ADC Periph

\*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

**GPIOPinTypeADC**(AINPort, AIN);

/\*

\* Enable ADC with this Sequence

\* 1. ADCSequenceConfigure()

\* 2. ADCSequenceStepConfigure()

\* 3. ADCSequenceEnable()

\* 4. ADCProcessorTrigger();

\* 5. Wait for sample sequence ADCIntStatus();

\* 6. Read From ADC

\*/

**ADCSequenceConfigure**(ADC0\_BASE, 0, ADC\_TRIGGER\_PROCESSOR, 0);

**ADCSequenceStepConfigure**(ADC0\_BASE, 0, 0, ADC\_CTL\_IE | ADC\_CTL\_END | ADC\_CTL\_CH0);

**ADCSequenceEnable**(ADC0\_BASE, 0);

/\*

\* Initialize the OLED

\*/

OrbitOledInit();

OrbitSetOled();

}

/\* ------------------------------------------------------------ \*/

/\*\*\* OrbitSetOled

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Set the OLED for Analog Demo

\*\*

\*/

**void** **OrbitSetOled**() {

**char** szAnalog[] = {'A', 'n', 'a', 'l', 'o', 'g', ':', ' ', '\0'};

**char** szDemo1[] = {'O', 'r', 'b', 'i', 't', ' ', 'D', 'e', 'm', 'o', '!', '\0'};

**char** szDemo2[] = {'B', 'y', ' ', 'D', 'i', 'g', 'i', 'l', 'e', 'n', 't', '\0'};

OrbitOledSetCursor(0, 0);

OrbitOledPutString(szDemo1);

OrbitOledSetCursor(0, 1);

OrbitOledPutString(szDemo2);

OrbitOledMoveTo(0,19);

OrbitOledLineTo(127, 19);

OrbitOledSetCursor(0, 4);

OrbitOledPutString(szAnalog);

}

/\* ------------------------------------------------------------ \*/

/\*\*\* OrbitDemo

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Switches and buttons turn on LEDs, and the ADC reading

\*\* (altered with the potentiometer, VR1) is continuously

\*\* output to the OLED.

\*/

**void** **OrbitDemo**() {

uint32\_t ulAIN0;

**long** lSwt1;

**long** lSwt2;

**long** lBtn1;

**long** lBtn2;

**char** szAIN[6] = {0};

**char** cMSB = 0x00;

**char** cMIDB = 0x00;

**char** cLSB = 0x00;

/\* Check SWT and BTN states and update LEDs

\*

\*/

lSwt1 = **GPIOPinRead**(SWT1Port, SWT1);

lSwt2 = **GPIOPinRead**(SWT2Port, SWT2);

lBtn1 = **GPIOPinRead**(BTN1Port, BTN1);

lBtn2 = **GPIOPinRead**(BTN2Port, BTN2);

**if**(lSwt1 == SWT1) {

**GPIOPinWrite**(LED3Port, LED3, LED3);

}

**else** {

**GPIOPinWrite**(LED3Port, LED3, LOW);

}

**if**(lSwt2 == SWT2) {

**GPIOPinWrite**(LED4Port, LED4, LED4);

}

**else** {

**GPIOPinWrite**(LED4Port, LED4, LOW);

}

**if**(lBtn1 == BTN1) {

**GPIOPinWrite**(LED1Port, LED1, LED1);

}

**else** {

**GPIOPinWrite**(LED1Port, LED1, LOW);

}

**if**(lBtn2 == BTN2) {

**GPIOPinWrite**(LED2Port, LED2, LED2);

}

**else** {

**GPIOPinWrite**(LED2Port, LED2, LOW);

}

/\*

\* Initiate ADC Conversion and update the OLED

\*/

**ADCProcessorTrigger**(ADC0\_BASE, 0);

**while**(!**ADCIntStatus**(ADC0\_BASE, 0, false));

**ADCSequenceDataGet**(ADC0\_BASE, 0, &ulAIN0);

/\*

\* Process data

\*/

cMSB = (0xF00 & ulAIN0) >> 8;

cMIDB = (0x0F0 & ulAIN0) >> 4;

cLSB = (0x00F & ulAIN0);

szAIN[0] = '0';

szAIN[1] = 'x';

szAIN[2] = (cMSB > 9) ? 'A' + (cMSB - 10) : '0' + cMSB;

szAIN[3] = (cMIDB > 9) ? 'A' + (cMIDB - 10) : '0' + cMIDB;

szAIN[4] = (cLSB > 9) ? 'A' + (cLSB - 10) : '0' + cLSB;

szAIN[5] = '\0';

OrbitOledSetCursor(8, 4);

OrbitOledPutString(szAIN);

}

**Task 2:**

// Addresses for the accelerometer

**#define** ACCEL\_W 0x3A // Write

**#define** ACCEL\_R 0x3B // read

**#define** ACCEL\_X 0x32 // LSB x-axis reg

**#define** ACCEL\_Y 0x34 // LSB y-axis reg

**#define** ACCEL\_Z 0x36 // LSB z-axis reg

**#define** ACCEL\_ADDR 0x1D

// Define needed for pin\_map.h

**#ifndef** PART\_TM4C123GH6PM

**#define** PART\_TM4C123GH6PM

**#endif**

**#include** <stdbool.h>

**#include** <stdint.h>

**#include** "inc/tm4c123gh6pm.h"

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "inc/hw\_i2c.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/i2c.h"

**#include** "driverlib/uart.h" // UART APIs

**void** **Accel\_int**(); // Function prototype to initialize the Accelerometer

**signed** **int** **Accel\_read**(**unsigned** **char**); // Function prototype to read the Accelerometer

**void** **Print\_header**(); // Print Header at start of program

**void** **print\_shortInt**(**signed** **short** **int**); // Print short int

**void** **print\_axis\_header**(**char**); // Print label for axis being printed

**void** **main**(**void**) {

**signed** **short** **int** value;

**SysCtlClockSet**(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_OSC | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ); //setup clock

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0); // Enable I2C hardware

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOB); // Enable Pin hardware

**GPIOPinConfigure**(GPIO\_PB3\_I2C0SDA); // Configure GPIO pin for I2C Data line

**GPIOPinConfigure**(GPIO\_PB2\_I2C0SCL); // Configure GPIO Pin for I2C clock line

**GPIOPinTypeI2C**(GPIO\_PORTB\_BASE, GPIO\_PIN\_2 | GPIO\_PIN\_3); // Set Pin Type

// Enable UART

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0); // Enable UART hardware

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA); // Enable Pin hardware

**GPIOPinConfigure**(GPIO\_PA0\_U0RX); // Configure GPIO pin for UART RX line

**GPIOPinConfigure**(GPIO\_PA1\_U0TX); // Configure GPIO Pin for UART TX line

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1); // Set Pins for UART

**UARTConfigSetExpClk**(UART0\_BASE, **SysCtlClockGet**(), 115200, // Configure UART to 8N1 at 115200bps

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

// Setup the I2C

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_2, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD);// SDA MUST BE STD

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_3, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_OD);// SCL MUST BE OPEN DRAIN

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), false); // The False sets the controller to 100kHz communication

Accel\_int(); // Function to initialize the Accelerometer

Print\_header();

**while**(1){

value = Accel\_read(ACCEL\_X);

print\_axis\_header('x');

print\_shortInt(value);

value = Accel\_read(ACCEL\_Y);

print\_axis\_header('y');

print\_shortInt(value);

value = Accel\_read(ACCEL\_Z);

print\_axis\_header('z');

print\_shortInt(value);

}

}

**void** **print\_axis\_header**(**char** axis) // Print header for axis

{

**unsigned** **char** \*label = "-axis = ";

**int** i = 0; // general counter

**UARTCharPut**(UART0\_BASE, '\r');

**UARTCharPut**(UART0\_BASE, axis);

**while**(label[i] != '\0'){ // Print Header at start of program

**UARTCharPut**(UART0\_BASE, label[i]);

i++;

}

}

**void** **print\_shortInt**(**signed** **short** **int** value){ // Print Header at start of program

**char** buffer[10];

**char** sign = '\0';

**int** i = 0; // iterator

**int** temp = value;

**if** (value == 0)

{

**UARTCharPut**(UART0\_BASE, '0');

**UARTCharPut**(UART0\_BASE, '\n');**UARTCharPut**(UART0\_BASE, '\r');

**return**;

}

**if** (value < 0)

{

sign = '-';

value \*= -1;

}

// Convert to string

**while**(temp != 0) // count the number of digits

{

i++;

temp /= 10;

}

buffer[i] = '\0';

i--;

**for**( i; i >= 0; i--) // convert digits to chars, and store in buffer

{

buffer[i] = value % 10 + '0';

value /= 10;

}

**UARTCharPut**(UART0\_BASE, sign);

**for**(i = 0; i < **sizeof**(buffer); i++) // Loop to print out data string

{

**if** (buffer[i] == '\0') **break**;

**UARTCharPut**(UART0\_BASE, buffer[i]);

}

**UARTCharPut**(UART0\_BASE, '\n');**UARTCharPut**(UART0\_BASE, '\r');

}

**void** **Print\_header**(){ // Print Header at start of program

**unsigned** **char** \*start\_screen = "\n\n\rLab 10 Accelerometer Sensor Read\n\r";

**int** i = 0; // general counter

**while**(start\_screen[i] != '\0'){ // Print Header at start of program

**UARTCharPut**(UART0\_BASE, start\_screen[i]);

i++;

}

}

**void** **Accel\_int**(){ // Function to initialize the Accelerometer

**I2CMasterSlaveAddrSet**(I2C0\_BASE, ACCEL\_ADDR, false); // false means transmit

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_START); // Send Start condition

**I2CMasterDataPut**(I2C0\_BASE, 0x2D); // Writing to the Accel control reg

**SysCtlDelay**(20000); // Delay for first transmission

**I2CMasterDataPut**(I2C0\_BASE, 0x08); // Send Value to control Register

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_FINISH); // Send Stop condition

**while**(**I2CMasterBusBusy**(I2C0\_BASE)){}; // Wait for I2C controller to finish operations

}

**signed** **int** **Accel\_read**(**unsigned** **char** axis\_addr) { // Function to read the Accelerometer

//signed int data;

**signed** **short** value = 0; // value of x

**unsigned** **char** MSB;

**unsigned** **char** LSB;

**I2CMasterSlaveAddrSet**(I2C0\_BASE, ACCEL\_ADDR, false); // false means transmit

**I2CMasterDataPut**(I2C0\_BASE, axis\_addr);

**SysCtlDelay**(20000);

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_SEND); // Request LSB of X Axis

**SysCtlDelay**(2000000); // Delay for first transmission

**I2CMasterSlaveAddrSet**(I2C0\_BASE, ACCEL\_ADDR, true); // false means transmit

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_RECEIVE); //Request LSB of X Axis

**SysCtlDelay**(20000);

LSB = **I2CMasterDataGet**(I2C0\_BASE);

**SysCtlDelay**(20000);

**I2CMasterSlaveAddrSet**(I2C0\_BASE, ACCEL\_ADDR, false); // false means transmit

**I2CMasterDataPut**(I2C0\_BASE, axis\_addr + 1);

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_SEND); // Request LSB of X Axis

**SysCtlDelay**(2000000); // Delay for first transmission

**I2CMasterSlaveAddrSet**(I2C0\_BASE, ACCEL\_ADDR, true); // false means transmit

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_RECEIVE); //Request LSB of X Axis

**SysCtlDelay**(20000);

MSB = **I2CMasterDataGet**(I2C0\_BASE);

value = (MSB << 8 | LSB);

**SysCtlDelay**(2000);

**return** value;

}

**Task 3:**

**#define** TEMP\_ADDR 0x4F // Address for Temp Sensor

// Define needed for pin\_map.h

**#define** PART\_TM4C123GH6PM

**#include** <stdbool.h>

**#include** <stdint.h>

**#include** "inc/tm4c123gh6pm.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** "inc/hw\_i2c.h"

**#include** "driverlib/i2c.h"

**void** **Print\_header**(); // Prints Header

**float** **Read\_temp**(); // Read Temperature sensor

**void** **main**(**void**) {

**float** value;

**SysCtlClockSet**(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_OSC | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ); //setup clock

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0); // Enable I2C hardware

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOB); // Enable Pin hardware

**GPIOPinConfigure**(GPIO\_PB3\_I2C0SDA); // Configure GPIO pin for I2C Data line

**GPIOPinConfigure**(GPIO\_PB2\_I2C0SCL); // Configure GPIO Pin for I2C clock line

**GPIOPinTypeI2C**(GPIO\_PORTB\_BASE, GPIO\_PIN\_2 | GPIO\_PIN\_3); // Set Pin Type

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_2, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD);// SDA MUST BE STD

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_3, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_OD);// SCL MUST BE OPEN DRAIN

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), false); // The False sets the controller to 100kHz communication

**I2CMasterSlaveAddrSet**(I2C0\_BASE, TEMP\_ADDR, true); // false means transmit

// Set up GPIO output for LEDs

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF); // PORTF

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_3); // red and green LEDs

**while**(1){

value = Read\_temp(); // Read Data from Temp Sensor

// Chose 27 C since cur readings of room temp were 27.

**if** (value > 27) // If temp > room temp, light red.

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_3, 2);

**else** // else, light green.

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_3, 8);

**SysCtlDelay**(6000000); // Delay

}

}

**float** **Read\_temp**(){ // Read Temperature sensor

**unsigned** **char** temp[2]; // storage for data

**float** value;

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_RECEIVE\_START); // Start condition

**SysCtlDelay**(20000); // Delay

temp[0] = **I2CMasterDataGet**(I2C0\_BASE); // Read first char

**SysCtlDelay**(20000); // Delay

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_RECEIVE\_CONT); // Push second Char

**SysCtlDelay**(20000); // Delay

temp[1] = **I2CMasterDataGet**(I2C0\_BASE); // Read second char

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_RECEIVE\_FINISH); // Stop Condition

value = temp[0];

**if** (temp[1] != 128)

value += 0.5;

**return** value;

}

**Video Link to Demo**

Task 1: <https://www.youtube.com/watch?v=0NcLEllnfHY>

Task 2: <https://www.youtube.com/watch?v=rwwS_c4vFi4>

Task 3[: https://www.youtube.com/watch?v=Qn-b4FWTSug](file:///C:\Users\Clinton\Documents\CPE403\Lab10\:%20https:\www.youtube.com\watch%3fv=Qn-b4FWTSug)