TivaC Lab 10 – I2C

CPE 403

**Checklist for Lab 10**

* 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:**

**Accelerometer:**

**#define** ACCEL\_W 0x3A // Addresses for the accelerometer

**#define** ACCEL\_R 0x3B

**#define** ACCEL\_ADDR 0x1D

**#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"

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

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

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

**signed** **short** **int** LED\_value = 1;

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

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

**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 Peripheral ports for output

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

**GPIOPinTypeGPIOOutput**(GPIO\_PORTC\_BASE, GPIO\_PIN\_6|GPIO\_PIN\_7); // LED 1 LED 2

**GPIOPinTypeGPIOOutput**(GPIO\_PORTB\_BASE, GPIO\_PIN\_5); // LED 4

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOD); // PORT D

**GPIOPinTypeGPIOOutput**(GPIO\_PORTD\_BASE, GPIO\_PIN\_6); // LED 3

// Setup the I2C

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_2, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD);

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_3, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_OD);

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

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

**while**(1){

LED\_value = LED\_value + Accel\_read();

**if**(LED\_value <= 1){

// Cycle through the LEDs on the Orbit board

**GPIOPinWrite**(GPIO\_PORTC\_BASE, GPIO\_PIN\_6|GPIO\_PIN\_7, 0x40); // LED 1 on LED 2 Off

**GPIOPinWrite**(GPIO\_PORTD\_BASE, GPIO\_PIN\_6, 0x00); // LED 3 off

**GPIOPinWrite**(GPIO\_PORTB\_BASE, GPIO\_PIN\_5, 0x00); // LED 4 off

LED\_value = 1; // reset value to maintain range

}

**else** **if**(LED\_value == 2){

// Cycle through the LEDs on the Orbit board

**GPIOPinWrite**(GPIO\_PORTC\_BASE, GPIO\_PIN\_6|GPIO\_PIN\_7, 0x80); // LED 1 off LED 2 on

**GPIOPinWrite**(GPIO\_PORTD\_BASE, GPIO\_PIN\_6, 0x00); // LED 3 off **GPIOPinWrite**(GPIO\_PORTB\_BASE, GPIO\_PIN\_5, 0x00); // LED 4 on

}

**else** **if**(LED\_value == 3){

// Cycle through the LEDs on the Orbit board

**GPIOPinWrite**(GPIO\_PORTC\_BASE, GPIO\_PIN\_6|GPIO\_PIN\_7, 0x00);// LED 1 off LED 2 off

**GPIOPinWrite**(GPIO\_PORTD\_BASE, GPIO\_PIN\_6, 0x40); // LED 3 on

**GPIOPinWrite**(GPIO\_PORTB\_BASE, GPIO\_PIN\_5, 0x00); // LED 4 0ff

}

**else** **if**(LED\_value >= 4){

// Cycle through the LEDs on the Orbit board

**GPIOPinWrite**(GPIO\_PORTC\_BASE, GPIO\_PIN\_6|GPIO\_PIN\_7, 0x00);// LED 1 off LED 2 Off

**GPIOPinWrite**(GPIO\_PORTD\_BASE, GPIO\_PIN\_6, 0x00); // LED 3 off

**GPIOPinWrite**(GPIO\_PORTB\_BASE, GPIO\_PIN\_5, 0x20); // LED 4 on

LED\_value = 4;

}

}

}

**void** **Accel\_int**(){

**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**() {

**signed** **int** data;

**signed** **short** value = 0;

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

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

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

**I2CMasterDataPut**(I2C0\_BASE, 0x32);

**SysCtlDelay**(20000);

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

**SysCtlDelay**(2000000);

**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, 0x33);

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

**SysCtlDelay**(2000000);

**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);

**if**(value < -250 ){ // testing axis for value

data = -1;

}

**else** **if** (value > 250){

data = 1;

}

**else**{

data = 0;

}

//SysCtlDelay(200);

//SysCtlDelay(20000);

**return** data; // return value

}

**Task 1:**

**Temperature Sensor:**

**#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** "driverlib/uart.h"

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

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

**unsigned** **char** start\_screen[29] = "\n\n\r ATE Lab 8 Temp Sensor \n\n\r";

**unsigned** **char** log[18] = "\n\n\r Temp reading: ";

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

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

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

**unsigned** **char** temp\_data[10] = "00.0 C \n\n\r"; // Temp format to be edited by read

**unsigned** **short** **int** i = 0;

// Setup the I2C see lab 7 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

**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);

**GPIOPadConfigSet**(GPIO\_PORTB\_BASE, GPIO\_PIN\_3, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_OD);

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

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Setup the UART see lab 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**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));

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Print\_header(); // Print Header

**while**(1){

Read\_temp(temp\_data); // Read Data from Temp Sensor

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

**for**(i=0;i<10;i++){ // Loop to print out data string

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

}

}

}

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

**int** i = 0;

**for**(i=0;i<29;i++){ // Print Header at start of program

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

}

}

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

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

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

**SysCtlDelay**(20000);

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

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

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_RECEIVE\_CONT); **SysCtlDelay**(20000);

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

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

data[0] = (temp[0] / 10) + 0x30; // convert 10 place to ASCII

data[1] = (temp[0] - ((temp[0] / 10)\*10)) + 0x30; // Convert 1's place to ASCII

**if**(temp[1] == 0x80){ // Test for .5 accuracy

data[3] = 0x35;

}

**else**{

data[3] = 0x30;

}

}

**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](:%20https:/www.youtube.com/watch?v=Qn-b4FWTSug)