TivaC Lab 11- SSI

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:**

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

// Define needed for pin\_map.h

**#define** PART\_TM4C123GH6PM

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.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** **display\_counter**();

**void** **OLEDprint\_uChar**(**unsigned** **char**);

/\*

\* main.c

\*/

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

/\* Include File Definitions \*/

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

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <time.h>

**#include** <stdio.h>

**#include** <math.h>

**#include** "LaunchPad.h"

**#include** "OrbitBoosterPackDefs.h"

**#include** "OrbitOled.h"

**#include** "OrbitOledGrph.h"

**#include** "OrbitOledChar.h"

**#include** "FillPat.h"

**#include** "I2CEEPROM.h"

**#include** "delay.h"

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

/\* General Type Definitions \*/

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

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

/\* Local Type Definitions \*/

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

**#define** DEMO\_0 0

**#define** DEMO\_1 2

**#define** DEMO\_2 1

**#define** DEMO\_3 3

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

/\* Global Variables \*/

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

**extern** **int** xchOledMax; // defined in OrbitOled.c

**extern** **int** ychOledMax; // defined in OrbitOled.c

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

/\* Local Variables \*/

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

**char** chSwtCur;

**char** chSwtPrev;

bool fClearOled;

/\*

\* Rocket Definitions

\*/

// Define the top left corner of rocket

**int** xcoRocketStart = 48; //8\*6

**int** ycoRocketStart = 11;

**int** xcoExhstStart = 39;

**int** ycoExhstStart = 11;

**int** cRocketWidth = 24;

**int** cRocketHeight = 16;

**int** cExhstWidth = 9;

**int** cExhstHeight = 16;

**int** fExhstSwt = 0;

**char** rgBMPRocket[] = { 0xFF, 0x11, 0xF1, 0x11, 0xF1, 0x12, 0x14, 0x18, 0x90,

0x10, 0x10, 0x10, 0x10, 0x10, 0x90, 0x10, 0x10, 0xE0, 0xC0, 0x80, 0x80,

0x80, 0x80, 0x80, 0xFF, 0x88, 0x8F, 0x88, 0x8F, 0x48, 0x28, 0x19, 0x0A,

0x09, 0x08, 0x08, 0x08, 0x09, 0x0A, 0x09, 0x08, 0x07, 0x03, 0x01, 0x01,

0x01, 0x01, 0x01 };

**char** rgBMPExhst1[] = { 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF0,

0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x0F };

**char** rgBMPExhst2[] = { 0x00, 0x80, 0x80, 0xC0, 0xE0, 0xE0, 0xF0, 0xF0, 0xF0,

0x00, 0x01, 0x01, 0x03, 0x07, 0x07, 0x0F, 0x0F, 0x0F };

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

/\* Forward Declarations \*/

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

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

**char** **CheckSwitches**();

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

**void** **OrbitDemo0**();

**void** **OrbitDemo1**();

**void** **OrbitDemo2**();

**void** **OrbitDemo3**();

**void** **RocketRight**(**int** xcoUpdate, **int** ycoUpdate);

**void** **RocketLeft**(**int** xcoUpdate, **int** ycoUpdate);

**void** **RocketStop**(**int** xcoUpdate, **int** ycoUpdate, bool fDir);

**char** **I2CGenTransmit**(**char** \* pbData, **int** cSize, bool fRW, **char** bAddr);

bool **I2CGenIsNotIdle**();

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

/\* Procedure Definitions \*/

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

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

/\*\*\* main()

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Main program loop

\*/

**#define** RED\_LED GPIO\_PIN\_1

**#define** BLUE\_LED GPIO\_PIN\_2

**#define** GREEN\_LED GPIO\_PIN\_3

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

**char** bDemoState = 0;

**volatile** uint32\_t ui32Loop;

DeviceInit();

**while** (1) {

bDemoState = CheckSwitches();

**for** (ui32Loop = 0; ui32Loop < 200000; ui32Loop++) {

}

**switch** (bDemoState) {

**case** DEMO\_0:

OrbitDemo0();

**break**;

**case** DEMO\_1:

OrbitDemo1();

**break**;

**case** DEMO\_2:

OrbitDemo2();

**break**;

**case** DEMO\_3:

OrbitDemo3();

**break**;

**default**:

OrbitDemo0();

**break**;

}

}

**return** 0;

}

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

/\*\*\* DeviceInit

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Initialize I2C Communication, and GPIO

\*/

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

/\*

\* 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 | SYSCTL\_PERIPH\_GPIOB | SYSCTL\_PERIPH\_GPIOC |

// SYSCTL\_PERIPH\_GPIOD | SYSCTL\_PERIPH\_GPIOE | SYSCTL\_PERIPH\_GPIOF);

**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();

/\*

\* Reset flags

\*/

chSwtCur = 0;

chSwtPrev = 0;

fClearOled = true;

}

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

/\*\*\* CheckSwitches()

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Return the state of the Switches

\*/

**char** **CheckSwitches**() {

**long** lSwt1;

**long** lSwt2;

chSwtPrev = chSwtCur;

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

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

chSwtCur = (lSwt1 | lSwt2) >> 6;

**if** (chSwtCur != chSwtPrev) {

fClearOled = true;

}

**return** chSwtCur;

}

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

/\*\*\* OrbitDemo0

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Buttons turn on LEDs, and the ADC reading

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

\*\* output to the OLED.

\*/

**void** **OrbitDemo0**() {

uint32\_t ulAIN0;

**long** lBtn1;

**long** lBtn2;

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

**char** cMSB = 0x00;

**char** cMIDB = 0x00;

**char** cLSB = 0x00;

**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' };

/\*

\* If applicable, reset OLED

\*/

**if** (fClearOled == true) {

OrbitOledClear();

OrbitOledMoveTo(0, 0);

OrbitOledSetCursor(0, 0);

fClearOled = false;

}

/\* Display Demo Banner

\*

\*/

OrbitOledSetCursor(0, 0);

OrbitOledPutString(szDemo1);

OrbitOledSetCursor(0, 1);

OrbitOledPutString(szDemo2);

OrbitOledMoveTo(0, 19);

OrbitOledLineTo(127, 19);

OrbitOledSetCursor(0, 4);

OrbitOledPutString(szAnalog);

/\* Check SWT and BTN states and update LEDs

\*

\*/

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

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

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

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

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

} **else** {

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

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

}

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

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

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

} **else** {

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

**GPIOPinWrite**(LED4Port, LED4, 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';

/\*

\* Update the Reading

\*/

OrbitOledSetCursor(8, 4);

OrbitOledPutString(szAIN);

}

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

/\*\*\* OrbitDemo1

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Writes received chars from USBUART to OLED and EEPROM. When

\*\* The ESC character is received, send back the last 25 characters

\*\* received.

\*/

**void** **OrbitDemo1**() {

**char** rgchRecv[25];

**char** chRecv = '-';

**char** chBck = 0x08; //backspace

**char** chEntr = 0x0D; //enter

**int** xCur = 0;

**int** yCur = 0;

**int** i;

**int** cNumRecv = 0;

/\*

\* If applicable, reset OLED

\*/

**if** (fClearOled == true) {

OrbitOledClear();

OrbitOledMoveTo(0, 0);

OrbitOledSetCursor(0, 0);

fClearOled = false;

/\*

\* Initialize UART on JB

\*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART1);

**GPIOPinTypeUART**(U1RXTXPort, UART1TXPin | UART1RXPin);

**GPIOPinConfigure**(UART1TX);

**GPIOPinConfigure**(UART1RX);

**UARTConfigSetExpClk**(UART1\_BASE, **SysCtlClockGet**(), 9600,

UART\_CONFIG\_WLEN\_8 |

UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE);

**UARTFlowControlSet**(UART1\_BASE, UART\_FLOWCONTROL\_NONE);

**UARTEnable**(UART1\_BASE);

/\*

\* Enable I2C Peripheral

\*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0);

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0);

/\*

\* Set I2C GPIO pins

\*/

**GPIOPinTypeI2C**(I2CSDAPort, I2CSDA\_PIN);

**GPIOPinTypeI2CSCL**(I2CSCLPort, I2CSCL\_PIN);

**GPIOPinConfigure**(I2CSCL);

**GPIOPinConfigure**(I2CSDA);

/\*

\* Setup I2C

\*/

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), false);

/\*

\* Initialize EEPROM

\*/

I2CEEPROMInit();

}

**while** (CheckSwitches() == DEMO\_1) {

/\*

\* If a byte has been sent, display it on the OLED

\*/

**if** (**UARTCharsAvail**(UART1\_BASE)) {

chRecv = (**char**) **UARTCharGetNonBlocking**(UART1\_BASE);

**if** (chRecv != chEntr) {

cNumRecv++;

**if** (cNumRecv >= 25) {

cNumRecv = 0;

}

I2CEEPROMWrite(&chRecv, cNumRecv, 1);

OrbitOledGetCursor(&xCur, &yCur);

**if** (xCur == 0 && yCur == 0) {

OrbitOledClear();

}

**if** (chRecv == chBck) {

OrbitOledClear();

OrbitOledSetCursor(0, 0);

cNumRecv = 0;

} **else** {

OrbitOledPutChar(chRecv);

}

} **else** {

cNumRecv++;

I2CEEPROMRead(rgchRecv, 1, cNumRecv);

**for** (i = 0; i < cNumRecv; i++) {

**UARTCharPut**(UART1\_BASE, rgchRecv[i]);

}

**UARTCharPut**(UART1\_BASE, ' ');

cNumRecv = 0;

}

}

}

}

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

/\*\*\* OrbitDemo2

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Reads the temperature and then updates the OLED display

\*\* with the temperature and alerts! if necessary

\*/

**void** **OrbitDemo2**() {

**char** szTempLabel[] = { 'T', 'e', 'm', 'p', ':', ' ', '\0' };

**char** szC[] = { ' ', 'C', '\0' };

**char** rgchReadTemp[] = { 0, 0, 0 };

**char** rgchWriteTemp[] = { 1, 0x20 };

**short** tempReg;

**short** tempWhole;

**short** tempDec;

**int** i;

**char** szTemp[6];

/\*

\* If applicable, reset OLED

\*/

**if** (fClearOled == true) {

OrbitOledClear();

OrbitOledMoveTo(0, 0);

OrbitOledSetCursor(0, 0);

fClearOled = false;

/\*

\* Setup Oled for Temperature

\*/

OrbitOledSetCursor(0, 0);

OrbitOledPutString(szTempLabel);

/\*

\* Enable I2C Peripheral

\*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0);

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0);

/\*

\* Set I2C GPIO pins

\*/

**GPIOPinTypeI2C**(I2CSDAPort, I2CSDA\_PIN);

**GPIOPinTypeI2CSCL**(I2CSCLPort, I2CSCL\_PIN);

**GPIOPinConfigure**(I2CSCL);

**GPIOPinConfigure**(I2CSDA);

/\*

\* Setup I2C

\*/

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), false);

/\*

\* Setup Temperature Sensor

\*/

I2CGenTransmit(rgchWriteTemp, 1, WRITE, TEMPADDR);

}

rgchReadTemp[0] = 0;

I2CGenTransmit(rgchReadTemp, 2, READ, TEMPADDR);

tempReg = (rgchReadTemp[1] << 8) | rgchReadTemp[2];

tempWhole = 0;

tempDec = 0;

**for** (i = 0; i < 7; i++) {

**if** (tempReg & (1 << (8 + i))) {

tempWhole += **pow**(2, i);

}

}

**if** (tempReg & (1 << 7)) {

tempDec += 50;

}

**if** (tempReg & (1 << 6)) {

tempDec += 25;

}

**sprintf**(szTemp, "%d.%d", tempWhole, tempDec);

**if** (tempDec == 0) {

szTemp[4] = ' ';

}

szTemp[5] = '\0';

OrbitOledSetCursor(6, 0);

OrbitOledPutString(szTemp);

OrbitOledSetCursor(11, 0);

OrbitOledPutString(szC);

}

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

/\*\*\* OrbitDemo3

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Prints a rocket ship to the OLED display and uses

\*\* Accelerometer to control it.

\*\*

\*/

**void** **OrbitDemo3**() {

**short** dataX;

**char** chPwrCtlReg = 0x2D;

**char** chX0Addr = 0x32;

**char** rgchReadAccl[] = { 0, 0, 0 };

**char** rgchWriteAccl[] = { 0, 0 };

**int** xcoRocketCur = xcoRocketStart;

**int** ycoRocketCur = ycoRocketStart;

**int** xcoExhstCur = xcoExhstStart;

**int** ycoExhstCur = ycoExhstStart;

**int** xDirThreshPos = 50;

**int** xDirThreshNeg = -50;

bool fDir = true;

/\*

\* If applicable, reset OLED

\*/

**if** (fClearOled == true) {

OrbitOledClear();

OrbitOledMoveTo(0, 0);

OrbitOledSetCursor(0, 0);

fClearOled = false;

/\*

\* Enable I2C Peripheral

\*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_I2C0);

**SysCtlPeripheralReset**(SYSCTL\_PERIPH\_I2C0);

/\*

\* Set I2C GPIO pins

\*/

**GPIOPinTypeI2C**(I2CSDAPort, I2CSDA\_PIN);

**GPIOPinTypeI2CSCL**(I2CSCLPort, I2CSCL\_PIN);

**GPIOPinConfigure**(I2CSCL);

**GPIOPinConfigure**(I2CSDA);

/\*

\* Setup I2C

\*/

**I2CMasterInitExpClk**(I2C0\_BASE, **SysCtlClockGet**(), false);

/\* Initialize the Accelerometer

\*

\*/

**GPIOPinTypeGPIOInput**(ACCL\_INT2Port, ACCL\_INT2);

rgchWriteAccl[0] = chPwrCtlReg;

rgchWriteAccl[1] = 1 << 3; // sets Accl in measurement mode

I2CGenTransmit(rgchWriteAccl, 1, WRITE, ACCLADDR);

}

/\*

\* Draw the starting Rocket

\*/

OrbitOledMoveTo(xcoRocketStart, ycoRocketStart);

OrbitOledPutBmp(cRocketWidth, cRocketHeight, rgBMPRocket);

OrbitOledUpdate();

/\*

\* Loop and check for movement until switches

\* change

\*/

**while** (CheckSwitches() == DEMO\_3) {

/\*

\* Read the X data register

\*/

rgchReadAccl[0] = chX0Addr;

I2CGenTransmit(rgchReadAccl, 2, READ, ACCLADDR);

dataX = (rgchReadAccl[2] << 8) | rgchReadAccl[1];

/\*

\* Check and see if Accel is positive or negative

\* and set fDir accordingly

\*/

**if** (dataX < 0 && dataX < xDirThreshNeg) {

fDir = true;

**if** (xcoRocketCur >= (ccolOledMax - 32)) {

xcoRocketCur = 0;

/\*

\* Clear the Oled

\*/

OrbitOledClear();

}

**else** {

xcoRocketCur++;

}

RocketRight(xcoRocketCur, ycoRocketCur);

}

**else** **if** (dataX > 0 && dataX > xDirThreshPos) {

fDir = false;

**if** (xcoRocketCur <= 0) {

xcoRocketCur = ccolOledMax - 32;

/\*

\* Clear the Oled

\*/

OrbitOledClear();

}

**else** {

xcoRocketCur--;

}

RocketLeft(xcoRocketCur, ycoRocketCur);

}

**else** {

RocketStop(xcoRocketCur, ycoRocketCur, fDir);

}

}

}

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

/\*\*\* RocketRight

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Moves the rocket to the right on the OLED display

\*\*

\*/

**void** **RocketRight**(**int** xcoUpdate, **int** ycoUpdate) {

OrbitOledMoveTo(xcoUpdate, ycoUpdate);

OrbitOledPutBmp(cRocketWidth, cRocketHeight, rgBMPRocket);

/\*

\* If Rocket moves right

\*/

OrbitOledMoveTo(xcoUpdate - cExhstWidth, ycoUpdate);

**if** (fExhstSwt == 0) {

OrbitOledPutBmp(cExhstWidth, cExhstHeight, rgBMPExhst1);

fExhstSwt++;

} **else** {

OrbitOledPutBmp(cExhstWidth, cExhstHeight, rgBMPExhst2);

fExhstSwt--;

}

OrbitOledUpdate();

}

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

/\*\*\* RocketLeft

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Moves the rocket to the left on the OLED display

\*\*

\*/

**void** **RocketLeft**(**int** xcoUpdate, **int** ycoUpdate) {

OrbitOledMoveTo(xcoUpdate, ycoUpdate);

OrbitOledPutBmpFlipped(cRocketWidth, cRocketHeight, rgBMPRocket);

/\*

\* If Rocket moves left

\*/

OrbitOledMoveTo(xcoUpdate + cRocketWidth, ycoUpdate);

**if** (fExhstSwt == 0) {

OrbitOledPutBmpFlipped(cExhstWidth, cExhstHeight, rgBMPExhst1);

fExhstSwt++;

} **else** {

OrbitOledPutBmpFlipped(cExhstWidth, cExhstHeight, rgBMPExhst2);

fExhstSwt--;

}

OrbitOledUpdate();

}

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

/\*\*\* RocketStop

\*\*

\*\* Parameters:

\*\* none

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Keeps the Rocket in one place on the OLED display

\*\*

\*/

**void** **RocketStop**(**int** xcoUpdate, **int** ycoUpdate, bool fDir) {

**if** (fDir) {

OrbitOledMoveTo(xcoUpdate - cExhstWidth, ycoUpdate);

OrbitOledSetFillPattern(OrbitOledGetStdPattern(0));

OrbitOledFillRect(xcoUpdate - 1, ycoUpdate + cExhstHeight);

} **else** {

OrbitOledMoveTo(xcoUpdate + cRocketWidth, ycoUpdate);

OrbitOledSetFillPattern(OrbitOledGetStdPattern(0));

OrbitOledFillRect(xcoUpdate + cRocketWidth + cExhstWidth,

ycoUpdate + cExhstHeight);

}

OrbitOledUpdate();

}

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

/\*\*\* I2CGenTransmit

\*\*

\*\* Parameters:

\*\* pbData - Pointer to transmit buffer (read or write)

\*\* cSize - Number of byte transactions to take place

\*\*

\*\* Return Value:

\*\* none

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Transmits data to a device via the I2C bus. Differs from

\*\* I2C EEPROM Transmit in that the registers in the device it

\*\* is addressing are addressed with a single byte. Lame, but..

\*\* it works.

\*\*

\*/

**char** **I2CGenTransmit**(**char** \* pbData, **int** cSize, bool fRW, **char** bAddr) {

**int** i;

**char** \* pbTemp;

pbTemp = pbData;

/\*Start\*/

/\*Send Address High Byte\*/

/\* Send Write Block Cmd

\*/

**I2CMasterSlaveAddrSet**(I2C0\_BASE, bAddr, WRITE);

**I2CMasterDataPut**(I2C0\_BASE, \*pbTemp);

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_START);

DelayMs(1);

/\* Idle wait

\*/

**while** (I2CGenIsNotIdle())

;

/\* Increment data pointer

\*/

pbTemp++;

/\*Execute Read or Write\*/

**if** (fRW == READ) {

/\* Resend Start condition

\*\* Then send new control byte

\*\* then begin reading

\*/

**I2CMasterSlaveAddrSet**(I2C0\_BASE, bAddr, READ);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

/\* Begin Reading

\*/

**for** (i = 0; i < cSize; i++) {

**if** (cSize == i + 1 && cSize == 1) {

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_RECEIVE);

DelayMs(1);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

} **else** **if** (cSize == i + 1 && cSize > 1) {

**I2CMasterControl**(I2C0\_BASE,

I2C\_MASTER\_CMD\_BURST\_RECEIVE\_FINISH);

DelayMs(1);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

} **else** **if** (i == 0) {

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

DelayMs(1);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

/\* Idle wait

\*/

**while** (I2CGenIsNotIdle())

;

} **else** {

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

DelayMs(1);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

/\* Idle wait

\*/

**while** (I2CGenIsNotIdle())

;

}

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

/\* Read Data

\*/

\*pbTemp = (**char**) **I2CMasterDataGet**(I2C0\_BASE);

pbTemp++;

}

} **else** **if** (fRW == WRITE) {

/\*Loop data bytes

\*/

**for** (i = 0; i < cSize; i++) {

/\* Send Data

\*/

**I2CMasterDataPut**(I2C0\_BASE, \*pbTemp);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

**if** (i == cSize - 1) {

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_FINISH);

DelayMs(1);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

} **else** {

**I2CMasterControl**(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_CONT);

DelayMs(1);

**while** (**I2CMasterBusy**(I2C0\_BASE))

;

/\* Idle wait

\*/

**while** (I2CGenIsNotIdle())

;

}

pbTemp++;

}

}

/\*Stop\*/

**return** 0x00;

}

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

/\*\*\* I2CGenIsNotIdle()

\*\*

\*\* Parameters:

\*\* pbData - Pointer to transmit buffer (read or write)

\*\* cSize - Number of byte transactions to take place

\*\*

\*\* Return Value:

\*\* TRUE is bus is not idle, FALSE if bus is idle

\*\*

\*\* Errors:

\*\* none

\*\*

\*\* Description:

\*\* Returns TRUE if the bus is not idle

\*\*

\*/

bool **I2CGenIsNotIdle**() {

**return** !**I2CMasterBusBusy**(I2C0\_BASE);

}

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

DeviceInit();

**while** (1) {

display\_counter();

}

}

/\*

\* DeviceInit

\*/

**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

\* Set message on on OLED

\*/

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

**char** \*name = "Clinton Bess";

**char** \*label = "CpE403:Lab 11";

**char** \*temp\_label = "Count:";

OrbitOledSetCursor(0, 0);

OrbitOledPutString(name);

OrbitOledSetCursor(0, 1);

OrbitOledPutString(label);

OrbitOledMoveTo(0, 19);

OrbitOledLineTo(127, 19);

OrbitOledSetCursor(0, 4);

OrbitOledPutString(temp\_label);

}

/\*

\* OrbitDemo

\*/

**void** **display\_counter**() {

**static** **unsigned** **char** counter = 0;

OrbitOledSetCursor(11, 4);

OLEDprint\_uChar(counter);

**if** (counter == 100)

**SysCtlDelay**(9000000); // Delay

counter++;

**if** (counter > 100) {

OrbitOledSetCursor(11, 4);

OrbitOledPutString(" ");

counter = 0;

}

**SysCtlDelay**(900000); // Delay

}

**void** **OLEDprint\_uChar**(**unsigned** **char** value) {

**char** buffer[10];

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

**int** temp = value;

**if** (value == 0) {

OrbitOledPutString("0");

**return**;

}

// Convert to string

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

{

i++;

temp /= 10;

}

buffer[i] = '\0';

i--;

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

{

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

value /= 10;

}

OrbitOledPutString(buffer);

}

**Task 2:**

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

**#define** PART\_TM4C123GH6PM

**#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"

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

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

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

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

**void** **OrbitDemo2**();

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

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

**char**\* **ftos**(**float**, **char**); // convert float to string (char\*)

**void** **init\_i2c**();

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

DeviceInit();

init\_i2c(); // Initiate i2c

**while** (1) {

OrbitDemo2();

}

}

/\*

/\*\*\* DeviceInit

\*/

**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

\* Set message on on OLED

\*/

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

**char** \*name = "Clinton Bess";

**char** \*label = "CpE403:Lab 11";

**char** \*temp\_label = "Temp:";

OrbitOledSetCursor(0, 0);

OrbitOledPutString(name);

OrbitOledSetCursor(0, 1);

OrbitOledPutString(label);

OrbitOledMoveTo(0, 19);

OrbitOledLineTo(127, 19);

OrbitOledSetCursor(0, 4);

OrbitOledPutString(temp\_label);

}

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

/\*\*\* OrbitDemo

\*/

**void** **OrbitDemo2**() {

**float** temp;

**char** temp\_str[5];

/\*

\* Read temperature and display.

\*/

Read\_temp(temp\_str, 'C');

OrbitOledSetCursor(8, 4);

OrbitOledPutString(temp\_str);

**SysCtlDelay**(20000000); // Delay

Read\_temp(temp\_str, 'F');

OrbitOledSetCursor(8, 4);

OrbitOledPutString(temp\_str);

**SysCtlDelay**(20000000); // Delay

}

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

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

**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

**if** (t == 'F')

temp[0] = (**unsigned** **char**) (temp[0] \* (9.0 / 5) + 32);

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

data[2] = t;

data[3] = '\0';

}

**float** **read\_float\_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;

}

**void** **init\_i2c**() {

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

}

**char**\* **ftos**(**float** fVal, **char** t)

// convert float to char\*. t must be 'F' or 'C'

{

**char** result[10];

**int** dVal, dec, i;

**if** (t == 'F') // if type is Farenheit, convert.

fVal = fVal \* (9 / 5) + 32;

fVal += 0.005; // round to nearest hundedth.

dVal = fVal;

dec = (**int**) (fVal \* 100) % 100;

result[0] = (dec % 10) + '0';

result[1] = (dec / 10) + '0';

result[2] = '.';

**while** (dVal > 0)

**for** (i = 3; i <= 4; i++) {

result[i] = (dVal % 10) + '0';

dVal /= 10;

}

result[6] = t;

result[7] = '\0';

**return** result;

}

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

// Define needed for pin\_map.h

**#define** PART\_TM4C123GH6PM

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.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** **display\_counter**();

**void** **OLEDprint\_uChar**(**unsigned** **char**);

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

DeviceInit();

**while** (1) {

display\_counter();

}

}

/\*

\* DeviceInit

\*/

**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

\* Set message on on OLED

\*/

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

**char** \*name = "Clinton Bess";

**char** \*label = "CpE403:Lab 11";

**char** \*temp\_label = "Count:";

OrbitOledSetCursor(0, 0);

OrbitOledPutString(name);

OrbitOledSetCursor(0, 1);

OrbitOledPutString(label);

OrbitOledMoveTo(0, 19);

OrbitOledLineTo(127, 19);

OrbitOledSetCursor(0, 4);

OrbitOledPutString(temp\_label);

}

/\*

\* OrbitDemo

\*/

**void** **display\_counter**() {

**static** **unsigned** **char** counter = 0;

OrbitOledSetCursor(11, 4);

OLEDprint\_uChar(counter);

**if** (counter == 100)

**SysCtlDelay**(9000000); // Delay

counter++;

**if** (counter > 100) {

OrbitOledSetCursor(11, 4);

OrbitOledPutString(" ");

counter = 0;

}

**SysCtlDelay**(900000); // Delay

}

**void** **OLEDprint\_uChar**(**unsigned** **char** value) {

**char** buffer[10];

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

**int** temp = value;

**if** (value == 0) {

OrbitOledPutString("0");

**return**;

}

// Convert to string

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

{

i++;

temp /= 10;

}

buffer[i] = '\0';

i--;

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

{

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

value /= 10;

}

OrbitOledPutString(buffer);

}

**Video Link to Demo**

Task 1: <https://www.youtube.com/watch?v=sRQxKIHKe2g>

Task 2: <https://www.youtube.com/watch?v=jNDbB6XXR-o>

Task 3: <https://www.youtube.com/watch?v=LDc6TpWwBRM>