**#include <stdarg.h>**

**#include <stdbool.h>**

**#include <stdint.h>**

**#include "TSL2591\_def.h"**

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

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

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

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

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

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

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

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

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

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

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

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

**#include "utils/uartstdio.c"**

**#include "utils/ustdlib.c"**

**#ifdef DEBUG**

**void**

**\_\_error\_\_(char \*pcFilename, uint32\_t ui32Line)**

**{**

**UARTprintf("Error at line %d of %s\n", ui32Line, pcFilename);**

**while(1)**

**{**

**}**

**}**

**#endif**

**void ConfigureUART(void) {**

**//Configure UART1 to port B**

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

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

**GPIOPinConfigure(GPIO\_PB0\_U1RX); //PBO = RX**

**GPIOPinConfigure(GPIO\_PB1\_U1TX); //pb1 = TX**

**GPIOPinTypeUART(GPIO\_PORTB\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);**

**UARTClockSourceSet(UART1\_BASE, UART\_CLOCK\_PIOSC); //Set uart clock**

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

**GPIOPinConfigure(GPIO\_PA0\_U0RX);**

**GPIOPinConfigure(GPIO\_PA1\_U0TX);**

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

**GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);\*/**

**UARTStdioConfig(1, 9600, 16000000); //Buad lowered to meet lua requirement**

**}**

**void I2C0\_Init() {**

**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);**

**// Set the clock of the I2C to ensure proper connection**

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

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

**; // wait while the master SDA is busy**

**}**

**void I2C0\_Write(uint8\_t addr, uint8\_t N, ...) {**

**// Writes data from master to slave**

**// Takes the address of the device, the number of arguments, and a variable**

**// amount of register addresses to write to**

**// Find the device based on the address given**

**I2CMasterSlaveAddrSet(I2C0\_BASE, addr, false);**

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

**va\_list vargs; // variable list to hold the register addresses passed**

**va\_start(vargs,N); // initialize the variable list with the number of arguments**

**// put the first argument in the list in to the I2C bus**

**I2CMasterDataPut(I2C0\_BASE, va\_arg(vargs, uint8\_t));**

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

**if (N == 1) // if only 1 argument is passed, send that register command then stop**

**{**

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

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

**va\_end(vargs);**

**} else**

**{**

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

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

**uint8\_t i;**

**for (i = 1; i < N - 1; i++) {**

**// send the next register address to the bus**

**I2CMasterDataPut(I2C0\_BASE, va\_arg(vargs, uint8\_t));**

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

**// burst send, keeps receiving until the stop signal is received**

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

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

**}**

**I2CMasterDataPut(I2C0\_BASE, va\_arg(vargs, uint8\_t));**

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

**// send the finish signal to stop transmission**

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

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

**va\_end(vargs);**

**}**

**}**

**uint32\_t I2C0\_Read(uint8\_t addr, uint8\_t reg) {**

**// Read data from slave to master**

**// Takes in the address of the device and the register to read from**

**// find the device based on the address given**

**I2CMasterSlaveAddrSet(I2C0\_BASE, addr, false);**

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

**;**

**// send the register to be read on to the I2C bus**

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

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

**;**

**// send the send signal to send the register value**

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

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

**;**

**// set the master to read from the device**

**I2CMasterSlaveAddrSet(I2C0\_BASE, addr, true);**

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

**;**

**// send the receive signal to the device**

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

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

**;**

**// return the data read from the bus**

**return I2CMasterDataGet(I2C0\_BASE);**

**}**

**void TSL2591\_init() {**

**// Initializes the TSL2591 to have a medium gain,**

**uint32\_t x;**

**x = I2C0\_Read(TSL2591\_ADDR,(TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_DEVICE\_ID)); // read the device ID**

**//To verify that the device is connected**

**/\*if (x == 0x50) {**

**// used during debuging to make sure correct ID is received**

**UARTprintf("Found Device! %i\n", x);**

**} else {**

**UARTprintf("Device not Found! %i\n", x);**

**while (1) {**

**}; // loop here if the dev ID is not correct**

**}\*/**

**// configures the TSL2591 to have medium gain adn integration time of 100ms**

**I2C0\_Write(TSL2591\_ADDR, 2, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_CONTROL),0x10);**

**// enables proper interrupts and power to work with TSL2591**

**I2C0\_Write(TSL2591\_ADDR, 2, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_ENABLE),**

**(TSL2591\_ENABLE\_POWERON | TSL2591\_ENABLE\_AEN |**

**TSL2591\_ENABLE\_AIEN | TSL2591\_ENABLE\_NPIEN));**

**}**

**uint32\_t GetLuminosity() {**

**// Function obtained from notes and provided material**

**// This function will read the channels of the TSL and returns the calculated**

**// value to the caller**

**float atime = 100.0f,**

**again =**

**25.0f; // the variables to be used to calculate proper lux value**

**uint16\_t ch0, ch1; // variable to hold the channels of the TSL2591**

**uint32\_t cp1, lux1, lux2, lux;**

**uint32\_t x = 1;**

**x = I2C0\_Read(TSL2591\_ADDR, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_CHAN1\_LOW));**

**x <<= 16;**

**x |= I2C0\_Read(TSL2591\_ADDR, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_CHAN0\_LOW));**

**ch1 = x >> 16;**

**ch0 = x & 0xFFFF;**

**cp1 = (uint32\_t)(atime \* again) / TSL2591\_LUX\_DF;**

**lux1 = (uint32\_t)((float)ch0 - (TSL2591\_LUX\_COEFB \* (float)ch1)) / cp1;**

**lux2 = (uint32\_t)((TSL2591\_LUX\_COEFC \* (float)ch0) -**

**(TSL2591\_LUX\_COEFD \* (float)ch1)) /cp1;**

**lux = (lux1 > lux2) ? lux1 : lux2;**

**return lux;**

**}**

**void main(void) {**

**// set the main clock to run at 40MHz**

**SysCtlClockSet(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_OSC | SYSCTL\_OSC\_MAIN |**

**SYSCTL\_XTAL\_16MHZ);**

**uint32\_t lux = 0, i;**

**uint32\_t luxAvg = 0;**

**ConfigureUART(); // configure the UART of Tiva C**

**I2C0\_Init(); // initialize the I2C0 of Tiva C**

**//UARTprintf("Lux Value: %i" lux);**

**TSL2591\_init(); // initialize the TSL2591**

**while(1){**

**SysCtlDelay(10\*SysCtlClockGet());**

**for (i = 0; i < 20; i++)**

**// finds the average of the lux channel to send through uart**

**{**

**lux = GetLuminosity();**

**luxAvg += lux;**

**}**

**luxAvg = luxAvg / 20;**

**// UARTprintf("\nLux Value: %d\r\n", luxAvg);**

**//SEND TO THINGSPEAK**

**UARTprintf("sk=net.createConnection(net.TCP, 0)\r\n");**

**SysCtlDelay(2000000);**

**UARTprintf("sk:on(\"receive\", function(sck, c) print(c) end )\n\r");**

**SysCtlDelay(2000000);**

**UARTprintf("sk:connect(80,\"api.thingspeak.com\")\n\r");**

**SysCtlDelay(5000000);**

**UARTprintf("sk:send(\"GET /update?api\_key=52D8QM61WCD47BDK&field1=%i HTTP/1.1\\r\\nHost: api.thingspeak.com\\r\\nConnection: keep-alive\\r\\nAccept: \*/\*\\r\\n\\r\\n\")\r\n", luxAvg);**

**// OLD AT COMMAND: GET /update?key=52D8QM61WCD47BDK&field1=9.99&headers=false HTTP/1.1{CrLf}Host: api.thingspeak.com{CrLf}Connection: close{CrLf}Accept: \*/\*{CrLf}{CrLf}**

**SysCtlDelay(2000000);**

**}**

**}**