**Date Submitted: 12/11/19**

**LAB2:**

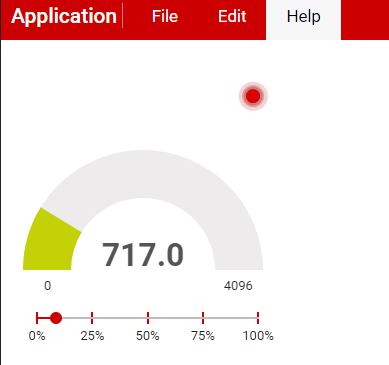
Youtube Link:

**Gui Composer:** [**https://youtu.be/ioe5VlzNEec**](https://youtu.be/ioe5VlzNEec)

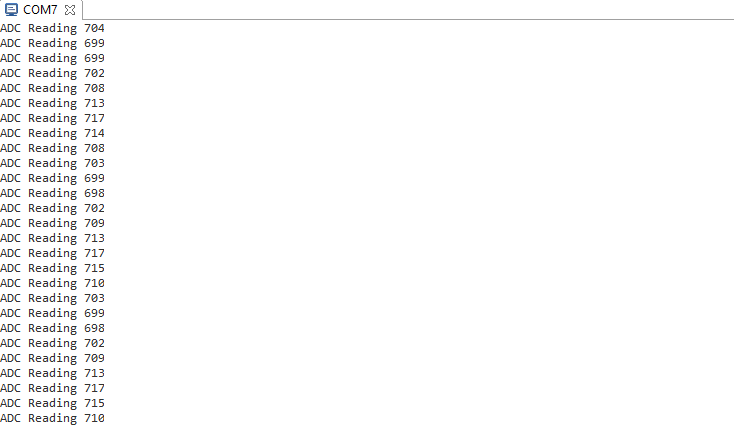
**UART:** [**https://youtu.be/7C1wNGooCO0**](https://youtu.be/7C1wNGooCO0)

**Modified Schematic (if applicable):**

**GUI Composer**

****

**UART shown with terminal**

****

**Modified Code:**

**// Insert code here**

/\*

\* Copyright (c) 2015-2019, Texas Instruments Incorporated

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\*

\* \* Neither the name of Texas Instruments Incorporated nor the names of

\* its contributors may be used to endorse or promote products derived

\* from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

\* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

\* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

\* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;

\* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

\* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

\* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,

\* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*

\* ======== empty.c ========

\*/

/\*Include Semaphores\*/

**#include** <ti/sysbios/knl/Semaphore.h>

**#include** <ti/sysbios/BIOS.h>

// Main loop Semaphore

Semaphore\_Struct semMainLoop;

Semaphore\_Handle hSemMainLoop;

/\*scif header and macro\*/

**#include** "scif.h"

**#define** BV(x) (1 << (x))

/\* For usleep() \*/

**#include** <unistd.h>

**#include** <stdint.h>

**#include** <stddef.h>

/\* Driver Header files \*/

**#include** <ti/drivers/GPIO.h>

// #include <ti/drivers/I2C.h>

// #include <ti/drivers/SPI.h>

// #include <ti/drivers/UART.h>

// #include <ti/drivers/Watchdog.h>

/\* Board Header file \*/

**#include** "Board.h"

/\*SC Task Alert Handling\*/

**void** **processTaskAlert**(**void**) {

// Clear the ALERT interrupt source

scifClearAlertIntSource();

//Do SC Task processing here

// Fetch 'state.high' variable from SC

uint8\_t high = scifTaskData.adcLevelTrigger.state.high;

// Set Red LED state equal to the state.high variable

**GPIO\_write**(Board\_GPIO\_RLED, high);

//Acknowledge the ALERT event

scifAckAlertEvents();

}

/\*SC callback functions\*/

**void** **scCtrlReadyCallback**(**void**) {

} // scCtrlReadyCallback

**void** **scTaskAlertCallback**(**void**) {

//Post to main loop semaphore

Semaphore\_post(hSemMainLoop);

} // scTaskAlertCallback

/\*

\* ======== mainThread ========

\*/

**void** \***tirtosScThread**(**void** \*arg0)

{

// Semaphore initialization

Semaphore\_Params semParams;

Semaphore\_Params\_init(&semParams);

Semaphore\_construct(&semMainLoop, 0, &semParams);

hSemMainLoop = Semaphore\_handle(&semMainLoop);

/\* 1 second delay \*/

//uint32\_t time = 1;

/\* Call driver init functions \*/

**GPIO\_init**();

// I2C\_init();

// SPI\_init();

// UART\_init();

// Watchdog\_init();

/\* Configure the LED pin \*/

**GPIO\_setConfig**(Board\_GPIO\_LED0, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_LOW);

/\* Turn on user LED \*/

**GPIO\_write**(Board\_GPIO\_LED0, Board\_GPIO\_LED\_ON);

/\*SC Driver Initialization\*/

// Initialize the Sensor Controller

scifOsalInit(); //init OSAL of the scif framework

scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback); //init CTRL ready callback

scifOsalRegisterTaskAlertCallback(scTaskAlertCallback); //init Task Alert callback

scifInit(&scifDriverSetup); //init SC task driver

// Set the Sensor Controller task tick interval to 1 second

uint32\_t rtc\_Hz = 1; // 1Hz RTC

scifStartRtcTicksNow(0x00010000 / rtc\_Hz);

//bits 31:16 represent the seconds, bits 15:0 represent 1/65536 of a second

// Configure Sensor Controller tasks

scifTaskData.adcLevelTrigger.cfg.threshold = 600; //set threshold value

//Start Sensor Controller task

scifStartTasksNbl(BV(SCIF\_ADC\_LEVEL\_TRIGGER\_TASK\_ID)); //execute task

**while** (1) {

// Wait on sem indefinitely

Semaphore\_pend(hSemMainLoop, BIOS\_WAIT\_FOREVER);

// Call process function

processTaskAlert();

}

}