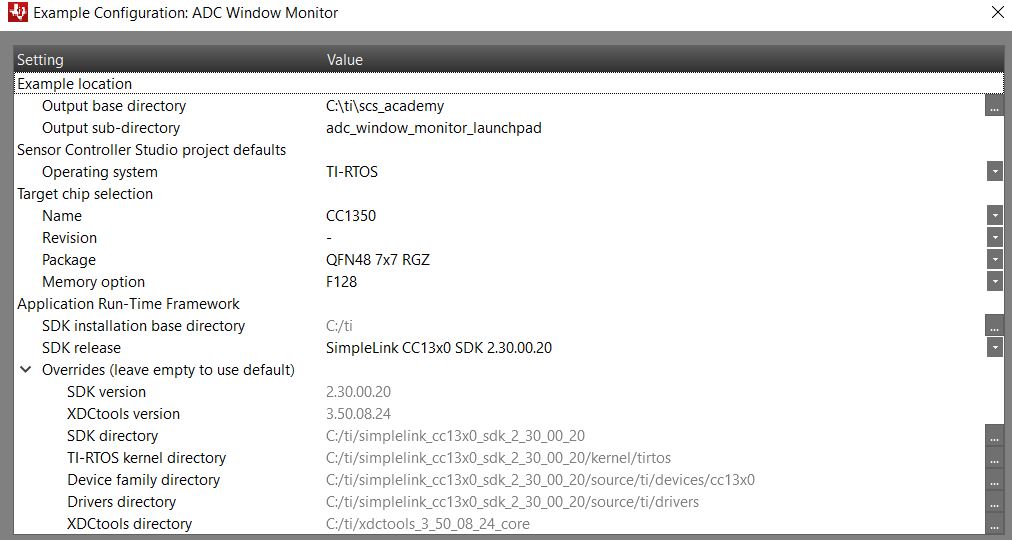
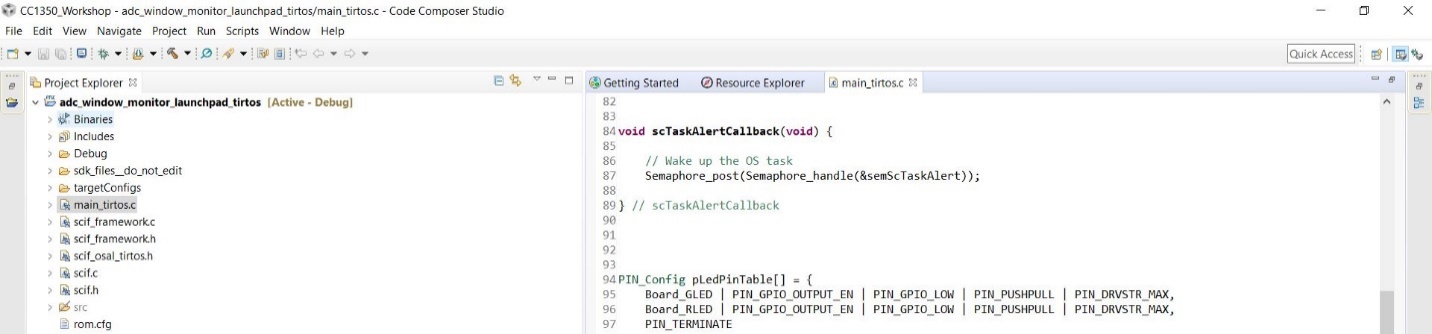
**Date Submitted: 11/21/2018**

**Task 01: Set up project in SCS**

****

****

Youtube Link: No video required for this task since we were only required to set up the project in SCS.

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**// Insert code here**

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

{

.

.

.

**SysCtlClockSet**(SYSCTL\_SYSDIV\_11|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN);

.

.

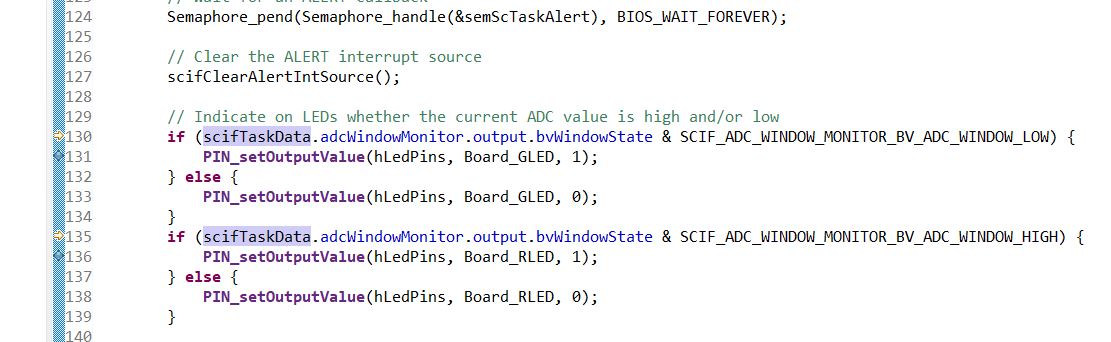
.

}

**------------------------------------------------------------------------------------**

**Task 02: Download and Debug with CCS**

For this task, I had to set breakpoints on lines 131 and 136 of the main\_tirtos.c file in CCS. This would allow me to alternate the ADC input between high and low using a jumper wire connected to either 3.3V or GND.

****

The image below on the left shows the red LED on because the ADC input is 3.3V, while the image on the right shows the green LED on because the ADC input is GND which is below the threshold.

** **

Youtube Link: <https://youtu.be/fyr3B33I2IQ>

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

**#include** "ex\_include\_tirtos.h"

**#include** "scif.h"

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

// Display error message if the SCIF driver has been generated with incorrect operating system setting

**#if** !(defined(SCIF\_OSAL\_TIRTOS\_H) || defined(SCIF\_OSAL\_TIDPL\_H))

**#error** "SCIF driver has incorrect operating system configuration for this example. Please change to 'TI-RTOS' or 'TI Driver Porting Layer' in the Sensor Controller Studio project panel and re-generate the driver."

**#endif**

// Display error message if the SCIF driver has been generated with incorrect target chip package

**#ifndef** SCIF\_TARGET\_CHIP\_PACKAGE\_QFN48\_7X7\_RGZ

**#error** "SCIF driver has incorrect target chip package configuration for this example. Please change to 'QFN48 7x7 RGZ' in the Sensor Controller Studio project panel and re-generate the driver."

**#endif**

// Task data

Task\_Struct myTask;

Char myTaskStack[1024];

// Semaphore used to wait for Sensor Controller task ALERT event

**static** Semaphore\_Struct semScTaskAlert;

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

} // scCtrlReadyCallback

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

// Wake up the OS task

Semaphore\_post(Semaphore\_handle(&semScTaskAlert));

} // scTaskAlertCallback

PIN\_Config pLedPinTable[] = {

Board\_GLED | PIN\_GPIO\_OUTPUT\_EN | PIN\_GPIO\_LOW | PIN\_PUSHPULL | PIN\_DRVSTR\_MAX,

Board\_RLED | PIN\_GPIO\_OUTPUT\_EN | PIN\_GPIO\_LOW | PIN\_PUSHPULL | PIN\_DRVSTR\_MAX,

PIN\_TERMINATE

};

PIN\_State ledPinState;

**void** **taskFxn**(UArg a0, UArg a1) {

PIN\_Handle hLedPins;

// Enable LED pins

hLedPins = **PIN\_open**(&ledPinState, pLedPinTable);

// Initialize the Sensor Controller

scifOsalInit();

scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback);

scifOsalRegisterTaskAlertCallback(scTaskAlertCallback);

scifInit(&scifDriverSetup);

scifStartRtcTicksNow(0x00010000 / 8);

// Configure and start the Sensor Controller's ADC window monitor task (not to be confused with OS tasks)

scifTaskData.adcWindowMonitor.cfg.adcWindowHigh = 800;

scifTaskData.adcWindowMonitor.cfg.adcWindowLow = 400;

scifStartTasksNbl(BV(SCIF\_ADC\_WINDOW\_MONITOR\_TASK\_ID));

// Main loop

**while** (1) {

// Wait for an ALERT callback

Semaphore\_pend(Semaphore\_handle(&semScTaskAlert), BIOS\_WAIT\_FOREVER);

// Clear the ALERT interrupt source

scifClearAlertIntSource();

// Indicate on LEDs whether the current ADC value is high and/or low

**if** (scifTaskData.adcWindowMonitor.output.bvWindowState & SCIF\_ADC\_WINDOW\_MONITOR\_BV\_ADC\_WINDOW\_LOW) {

**PIN\_setOutputValue**(hLedPins, Board\_GLED, 1);

} **else** {

**PIN\_setOutputValue**(hLedPins, Board\_GLED, 0);

}

**if** (scifTaskData.adcWindowMonitor.output.bvWindowState & SCIF\_ADC\_WINDOW\_MONITOR\_BV\_ADC\_WINDOW\_HIGH) {

**PIN\_setOutputValue**(hLedPins, Board\_RLED, 1);

} **else** {

**PIN\_setOutputValue**(hLedPins, Board\_RLED, 0);

}

// Acknowledge the alert event

scifAckAlertEvents();

}

} // taskFxn

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

Task\_Params taskParams;

// Initialize the board

Board\_initGeneral();

**#ifdef** Board\_shutDownExtFlash

Board\_shutDownExtFlash();

**#endif**

// Configure the OS task

Task\_Params\_init(&taskParams);

taskParams.stack = myTaskStack;

taskParams.stackSize = **sizeof**(myTaskStack);

taskParams.priority = 3;

Task\_construct(&myTask, taskFxn, &taskParams, NULL);

// Create the semaphore used to wait for Sensor Controller ALERT events

Semaphore\_Params semParams;

Semaphore\_Params\_init(&semParams);

semParams.mode = Semaphore\_Mode\_BINARY;

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

// Start TI-RTOS

BIOS\_start();

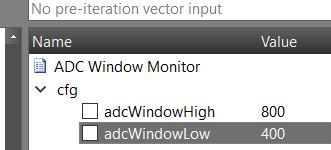
**return** 0;

} // main

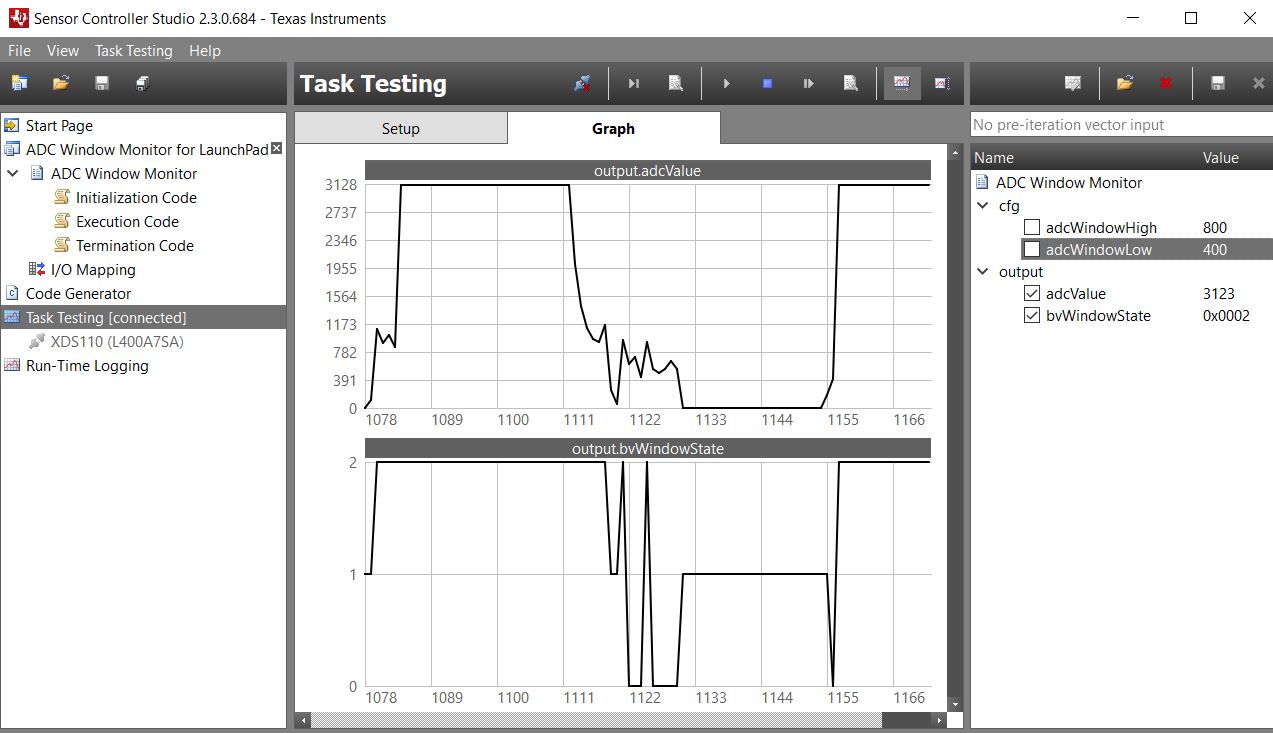
**------------------------------------------------------------------------------------**

**Task 03: Download and Debug with SCS**

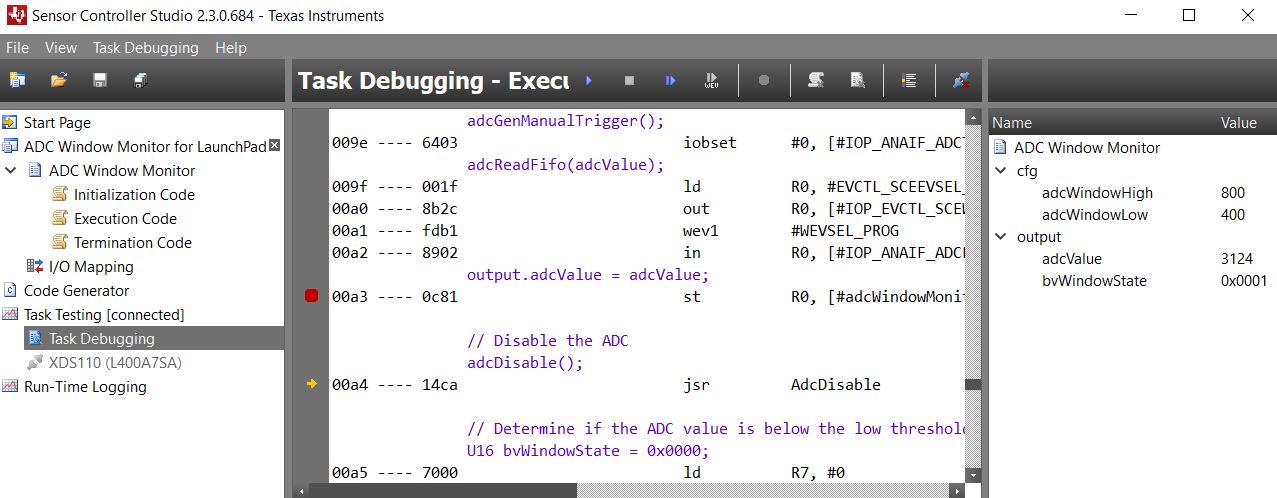
The image below shows the settings I used for the cfg in Task Testing.

****

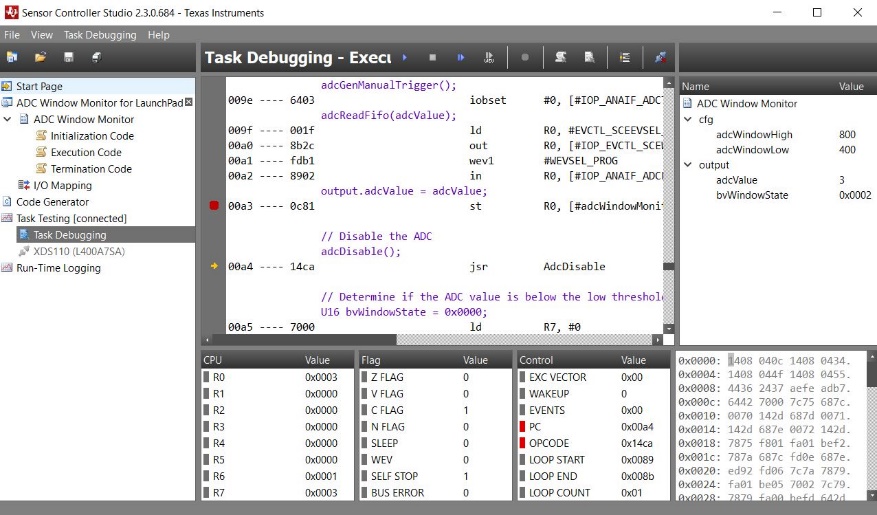
The image below shows the graph in task testing as I changed the ADC input from GND to 3.3V and vice versa. The spikes in the middle of the graph are caused by the time the jumper wire was not connected to either ground or 3.3V.

****

The image below shows the initial ADC input value of 3124.

****

After changing the ADC input (from 3.3V to GND), the new value loaded into the variable adcValue changed to 3.

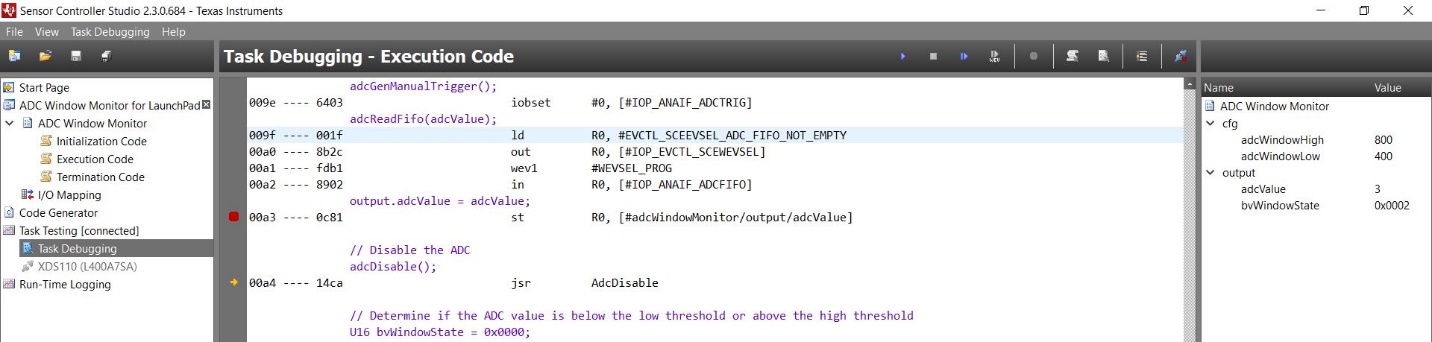
****

Youtube Link: <https://youtu.be/YeaXrwqoaDA>

**Modified Schematic (if applicable):**

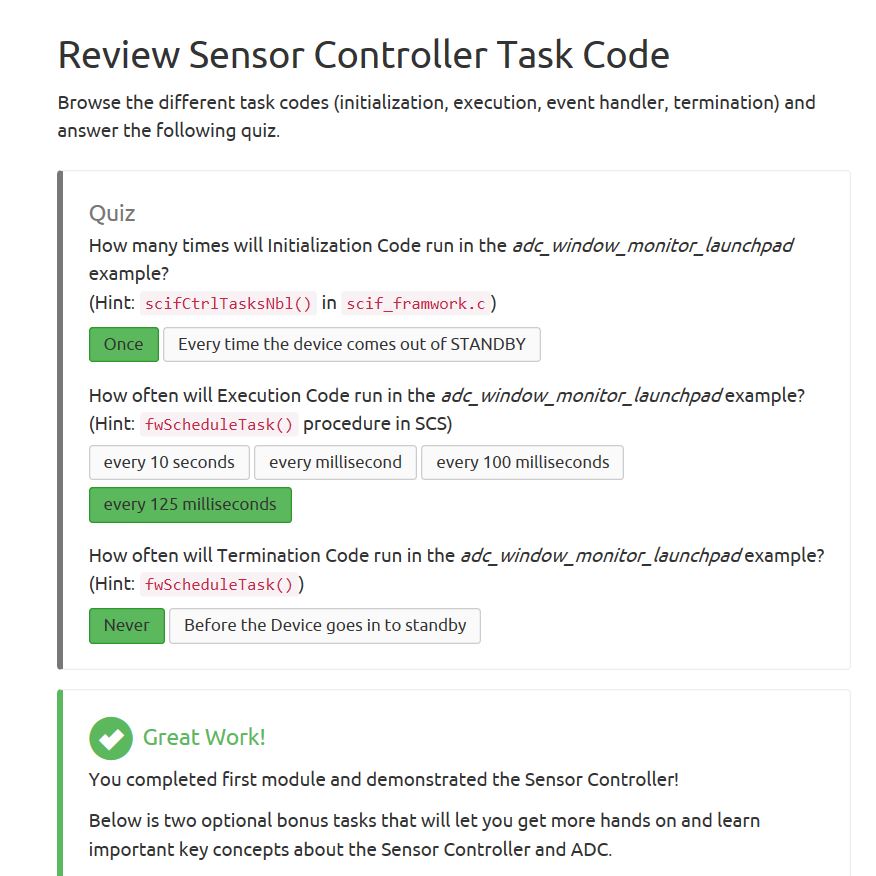
**Modified Code: No modifications to the code were made in CCS since we only used SCS for this task.**

**For this task all we had to add to the execution code was a breakpoint, so we can clearly see the adcValue change.**

****

**------------------------------------------------------------------------------------**

**Task 04: Understand**

****

Youtube Link: Video is not required for this task because the goal of this task is for me to understand what is happening when the code is executed.

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**// Insert code here**

**#include** "ex\_include\_tirtos.h"

**#include** "scif.h"

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

// Display error message if the SCIF driver has been generated with incorrect operating system setting

**#if** !(defined(SCIF\_OSAL\_TIRTOS\_H) || defined(SCIF\_OSAL\_TIDPL\_H))

**#error** "SCIF driver has incorrect operating system configuration for this example. Please change to 'TI-RTOS' or 'TI Driver Porting Layer' in the Sensor Controller Studio project panel and re-generate the driver."

**#endif**

// Display error message if the SCIF driver has been generated with incorrect target chip package

**#ifndef** SCIF\_TARGET\_CHIP\_PACKAGE\_QFN48\_7X7\_RGZ

**#error** "SCIF driver has incorrect target chip package configuration for this example. Please change to 'QFN48 7x7 RGZ' in the Sensor Controller Studio project panel and re-generate the driver."

**#endif**

// Task data

Task\_Struct myTask;

Char myTaskStack[1024];

// Semaphore used to wait for Sensor Controller task ALERT event

**static** Semaphore\_Struct semScTaskAlert;

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

} // scCtrlReadyCallback

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

// Wake up the OS task

Semaphore\_post(Semaphore\_handle(&semScTaskAlert));

} // scTaskAlertCallback

PIN\_Config pLedPinTable[] = {

Board\_GLED | PIN\_GPIO\_OUTPUT\_EN | PIN\_GPIO\_LOW | PIN\_PUSHPULL | PIN\_DRVSTR\_MAX,

Board\_RLED | PIN\_GPIO\_OUTPUT\_EN | PIN\_GPIO\_LOW | PIN\_PUSHPULL | PIN\_DRVSTR\_MAX,

PIN\_TERMINATE

};

PIN\_State ledPinState;

**void** **taskFxn**(UArg a0, UArg a1) {

PIN\_Handle hLedPins;

// Enable LED pins

hLedPins = **PIN\_open**(&ledPinState, pLedPinTable);

// Initialize the Sensor Controller

scifOsalInit();

scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback);

scifOsalRegisterTaskAlertCallback(scTaskAlertCallback);

scifInit(&scifDriverSetup);

scifStartRtcTicksNow(0x00010000 / 8);

// Configure and start the Sensor Controller's ADC window monitor task (not to be confused with OS tasks)

scifTaskData.adcWindowMonitor.cfg.adcWindowHigh = 800;

scifTaskData.adcWindowMonitor.cfg.adcWindowLow = 400;

scifStartTasksNbl(BV(SCIF\_ADC\_WINDOW\_MONITOR\_TASK\_ID));

// Main loop

**while** (1) {

// Wait for an ALERT callback

Semaphore\_pend(Semaphore\_handle(&semScTaskAlert), BIOS\_WAIT\_FOREVER);

// Clear the ALERT interrupt source

scifClearAlertIntSource();

// Indicate on LEDs whether the current ADC value is high and/or low

**if** (scifTaskData.adcWindowMonitor.output.bvWindowState & SCIF\_ADC\_WINDOW\_MONITOR\_BV\_ADC\_WINDOW\_LOW) {

**PIN\_setOutputValue**(hLedPins, Board\_GLED, 1);

} **else** {

**PIN\_setOutputValue**(hLedPins, Board\_GLED, 0);

}

**if** (scifTaskData.adcWindowMonitor.output.bvWindowState & SCIF\_ADC\_WINDOW\_MONITOR\_BV\_ADC\_WINDOW\_HIGH) {

**PIN\_setOutputValue**(hLedPins, Board\_RLED, 1);

} **else** {

**PIN\_setOutputValue**(hLedPins, Board\_RLED, 0);

}

// Acknowledge the alert event

scifAckAlertEvents();

}

} // taskFxn

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

Task\_Params taskParams;

// Initialize the board

Board\_initGeneral();

**#ifdef** Board\_shutDownExtFlash

Board\_shutDownExtFlash();

**#endif**

// Configure the OS task

Task\_Params\_init(&taskParams);

taskParams.stack = myTaskStack;

taskParams.stackSize = **sizeof**(myTaskStack);

taskParams.priority = 3;

Task\_construct(&myTask, taskFxn, &taskParams, NULL);

// Create the semaphore used to wait for Sensor Controller ALERT events

Semaphore\_Params semParams;

Semaphore\_Params\_init(&semParams);

semParams.mode = Semaphore\_Mode\_BINARY;

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

// Start TI-RTOS

BIOS\_start();

**return** 0;

} // main

**------------------------------------------------------------------------------------**