Martin Villasenor

CC1350 TI-RTOS/RF LAB4

**11/21/2018**

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

Youtube Task3 Link: <https://youtu.be/XvPzqgXX-No>

Task5 Link: https://youtu.be/jpSgUnMZgLA

**Modified Code:**

**Modifications are highlighted**

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

**#include** "scif.h"

**#include** <ti/devices/DeviceFamily.h>

**#include** DeviceFamily\_constructPath(driverlib/aux\_adc.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) {

int32\_t adcOffset = AUXADCGetAdjustmentOffset(AUXADC\_REF\_FIXED);

int32\_t adcGainError = AUXADCGetAdjustmentGain(AUXADC\_REF\_FIXED);

int32\_t adcValue, adcCorrectedValue, adcValueMicroVolt;

/\* Set threshold based on microVolt \*/

int32\_t adcThresholdLowMicro = 400000; // 0.4 Volt

int32\_t adcThresholdHighMicro = 800000; // 0.8 Volt

int32\_t adcThresholdLow = AUXADCMicrovoltsToValue(AUXADC\_FIXED\_REF\_VOLTAGE\_NORMAL, adcThresholdLowMicro);

int32\_t adcThresholdHigh = AUXADCMicrovoltsToValue(AUXADC\_FIXED\_REF\_VOLTAGE\_NORMAL, adcThresholdHighMicro);

int32\_t adcThresholdLowRaw = AUXADCUnadjustValueForGainAndOffset(adcThresholdLow, adcGainError, adcOffset);

int32\_t adcThresholdHighRaw = AUXADCUnadjustValueForGainAndOffset(adcThresholdHigh, adcGainError, adcOffset);

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));

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

scifTaskData.adcWindowMonitor.cfg.adcWindowHigh = adcThresholdHighRaw;

scifTaskData.adcWindowMonitor.cfg.adcWindowLow = adcThresholdLowRaw;

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();

// Read ADC value

adcValue = scifTaskData.adcWindowMonitor.output.adcValue;

// Correct ADC raw value

adcCorrectedValue = AUXADCAdjustValueForGainAndOffset((int32\_t) adcValue, adcGainError, adcOffset);

// Convert ADC value to Microvolts.

adcValueMicroVolt = AUXADCValueToMicrovolts(AUXADC\_FIXED\_REF\_VOLTAGE\_NORMAL,adcCorrectedValue);

// int32\_t adcValue = (int32\_t) scifTaskData.adcWindowMonitor.output.adcValue;

// 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

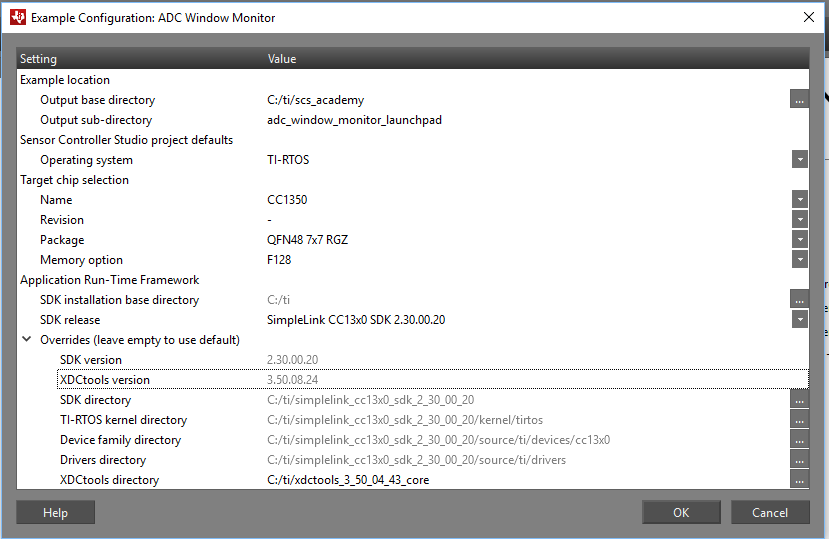
BIOS\_start();

**return** 0;

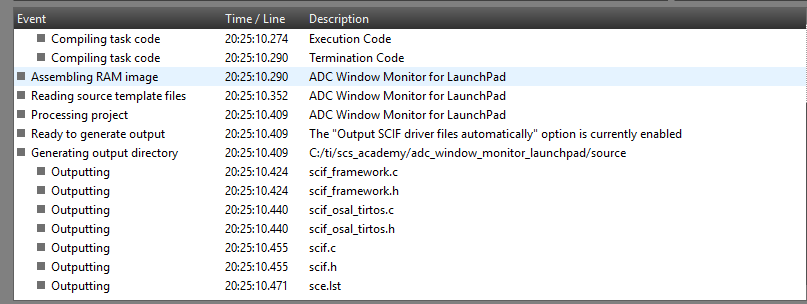
} // main

**TASK1:**

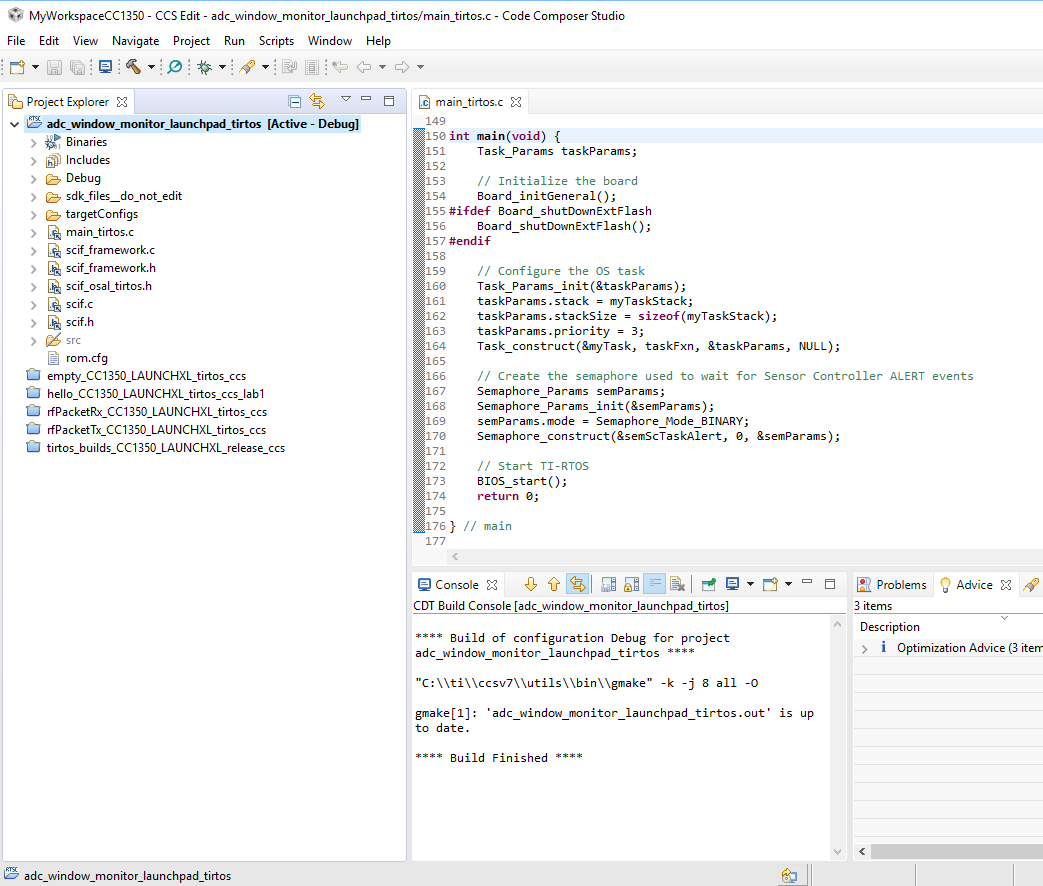
Configuration for Sensor Controller



Generating source code. Output shown on Event screen.



Imported scs project to ccs. Built correctly

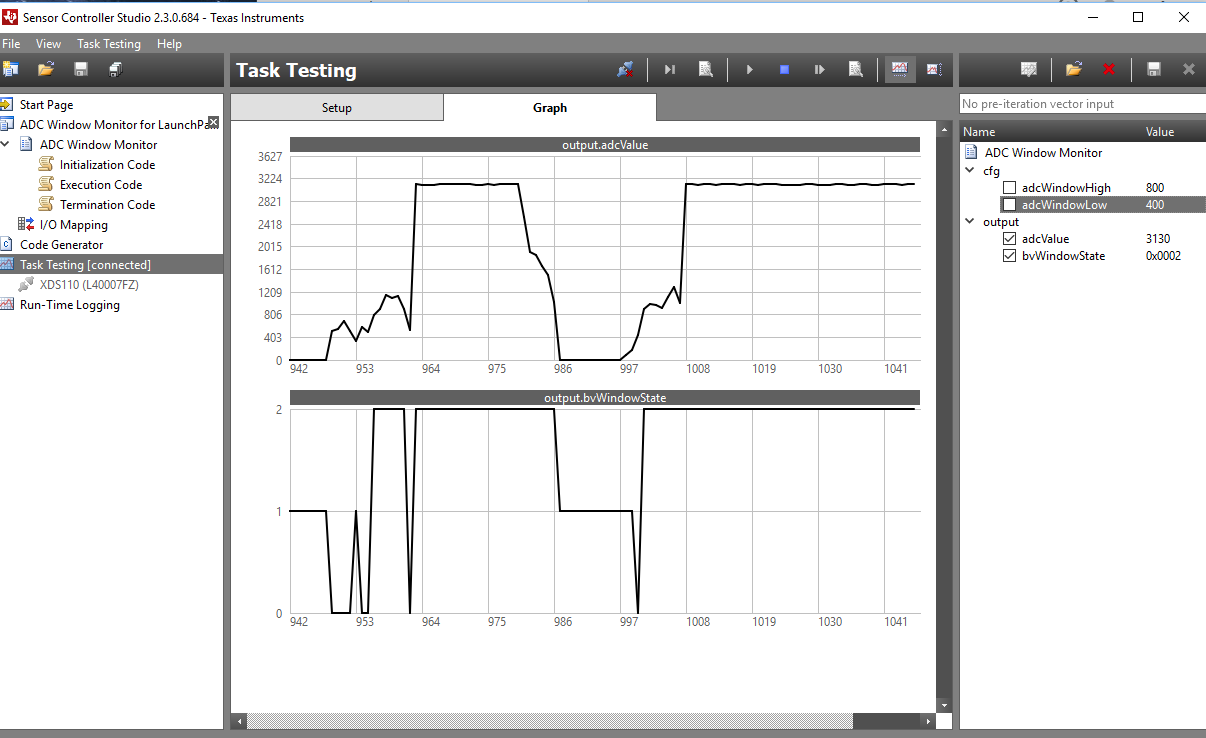


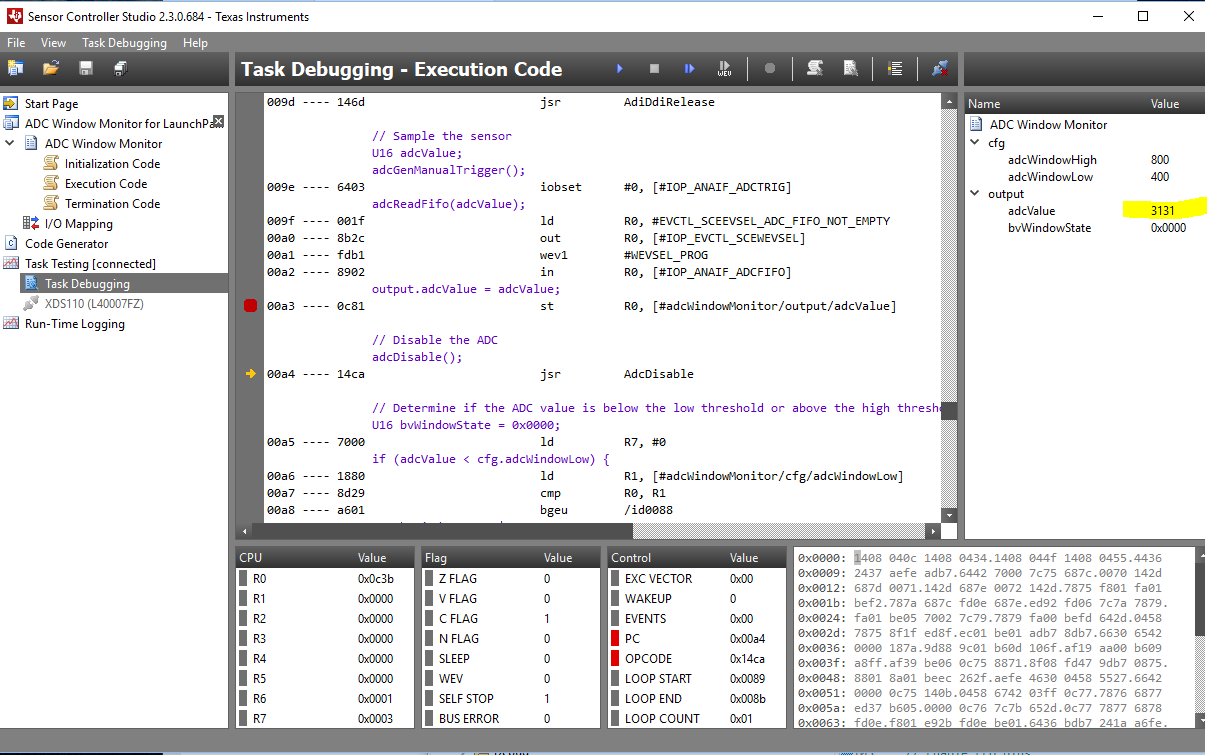
**TASK2:**

Running and debugging project.

**TASK3:**

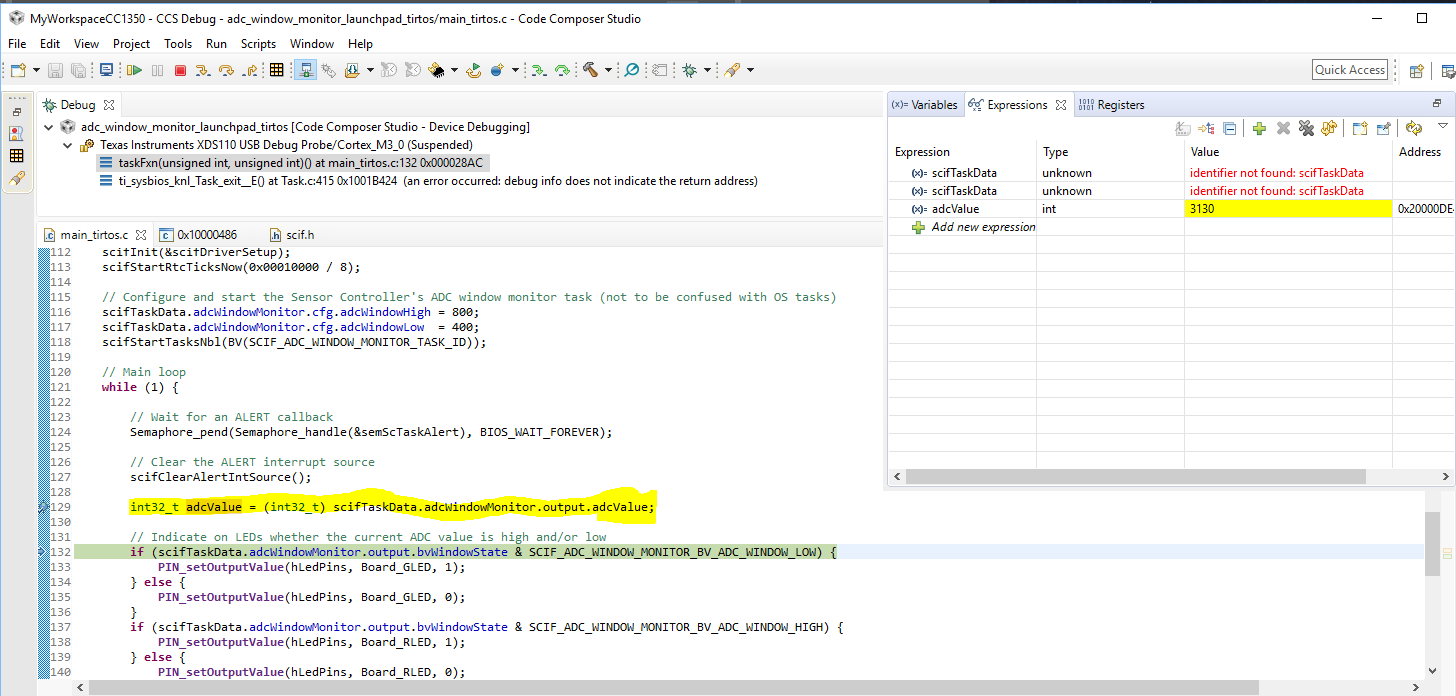
Debugging directly from Sensor Controller Studio.



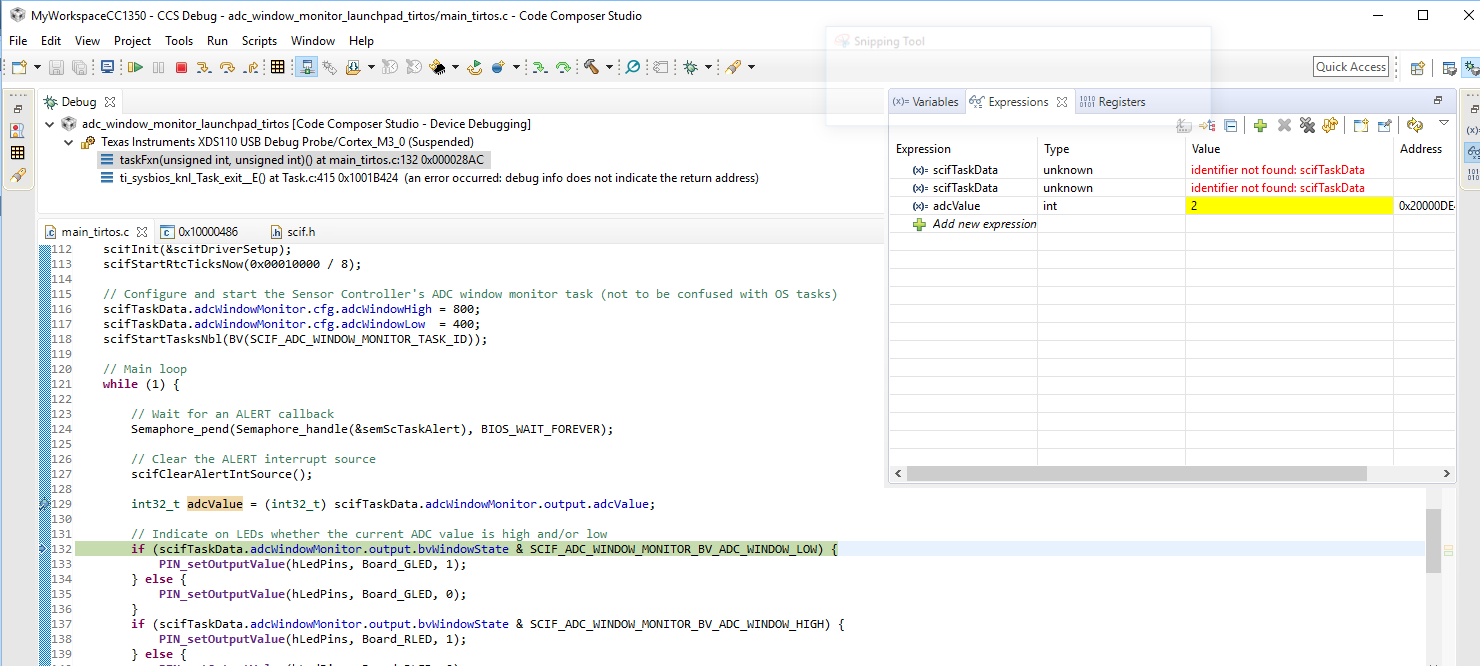


**TASK5: Bonus Task**

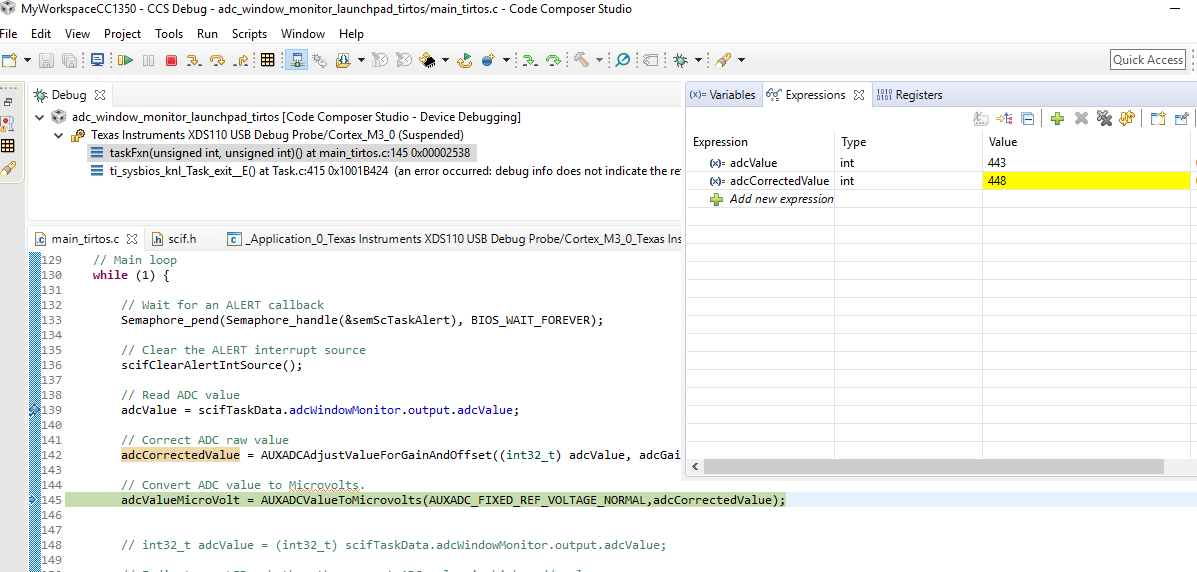
Reading raw value connecting ADC to 3.3 volt pin.



Changing ADC input voltage to 0 (Ground pin) after reading



Correcting raw ADC value and converting to microvolts.



After building and debugging modified code including new variables and aux\_adc apis, observing difference between raw adc value (443) and corrected acd value(448) is 5.

Configuring Microvolt threshold. Modified code shown above.