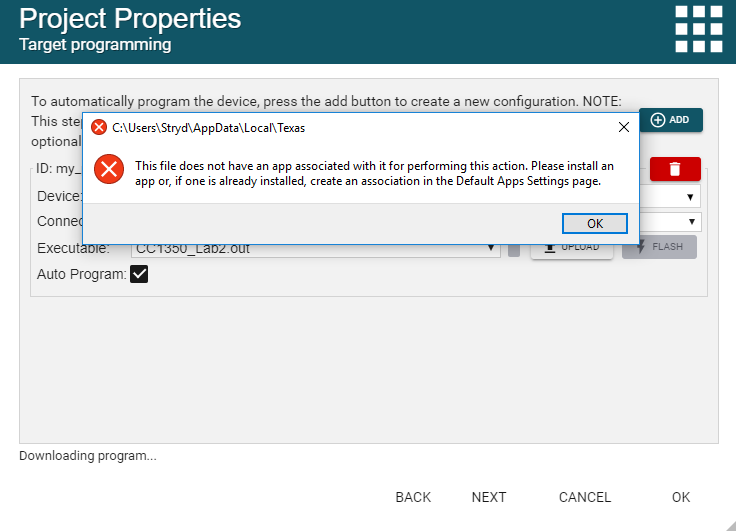
**Date Submitted: 12/04/2018**

**Assignment Youtube Video:https://youtu.be/8nlGG\_RXQPo**

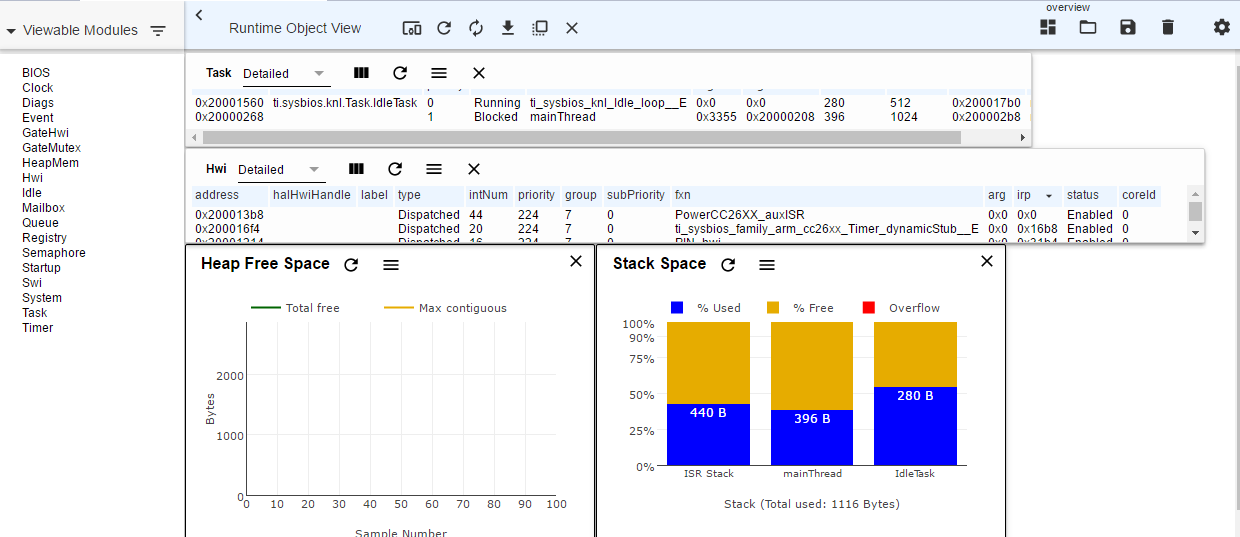
# Part 01: GUI Composer

Unfortunately, I had to skip all elements of GUI composer for this lab, as the interface still refuses to install drivers on my computer similar to the TIVA-C labs.



# Task 01: Finding the Blink LED code example

In Task 1 we find and import the “empty” project. Below are the statistics of the unmodified code.



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

# Task 02: Let's modify the example so LED is ON only if an ADC reading exceeds a threshold

For Task 2 we introduce code for reading the value of our ADC and turning on the LED if the ADC value passes a threshold. Modified code and stack information below.

**ADC\_init**();

ADC\_Handle adc;

ADC\_Params params;

**ADC\_Params\_init**(&params);

adc = **ADC\_open**(Board\_ADC0, &params);

**if** (adc == NULL) {

// Error initializing ADC channel 0

**while** (1);

}

**while** (1) {

int\_fast16\_t res;

res = **ADC\_convert**(adc, &adcValue);

**if** (res == ADC\_STATUS\_SUCCESS) {

//Display\_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);

**if**(adcValue >= threshold){

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

trigger = 1;

} **else**{

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

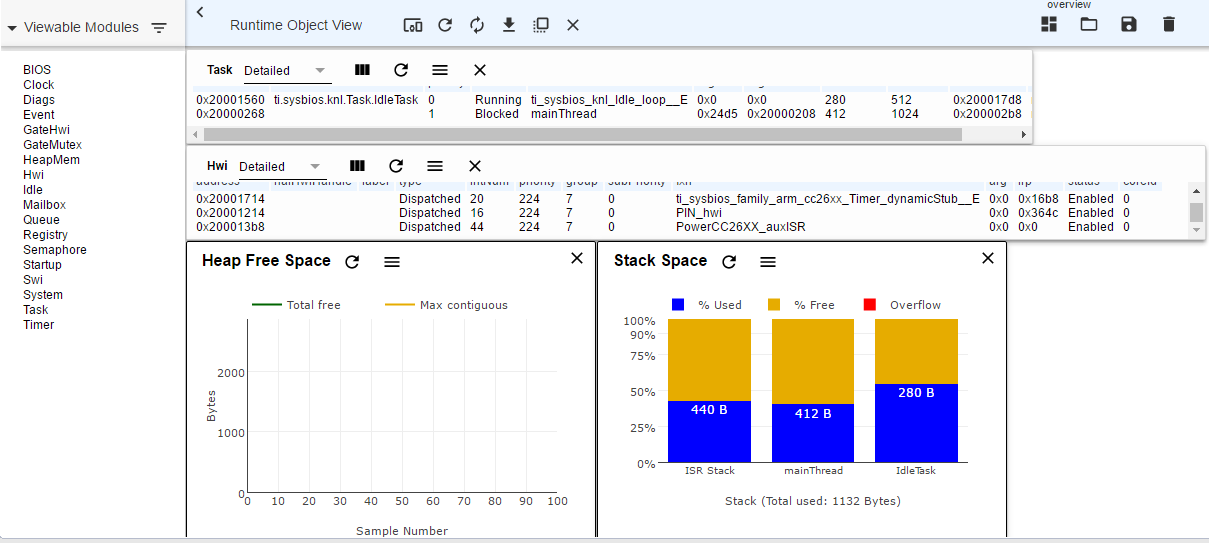
trigger = 0;

}

}

**usleep**(time);

}



# Task 03: Adding a serial UART transmission to report ADC readings

For Task 3 we introduce code for serial UART communication to report our ADC values. Modified code and stack information below.

**#include** <ti/display/Display.h>

Display\_Handle displayHandle;

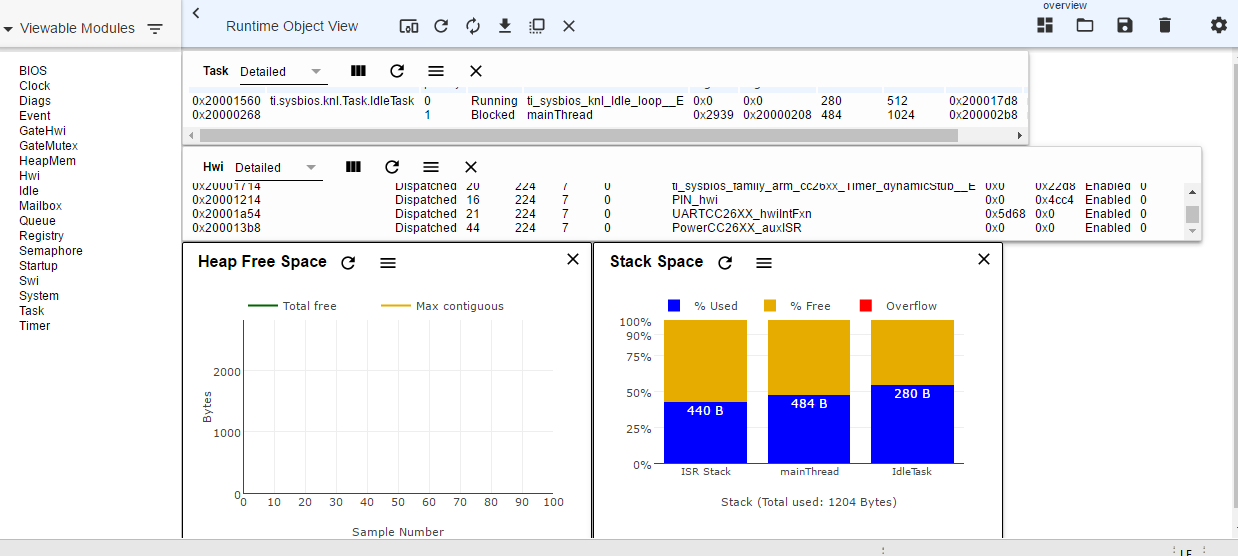
Display\_Params displayParams;

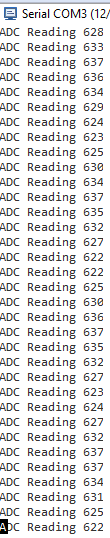
Display\_Params\_init(&displayParams);

displayHandle = Display\_open(Display\_Type\_UART, NULL);

We also add the following code into our main While(1) loop:

Display\_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);





# Task 04: Adding GPIO interrupts to our base example.

For Task 4 we introduce code for serial UART communication to report our ADC values. Modified code and stack information below.

First, we add callback functions for our interrupts. We do not need to initialize our GPIO again, as they are already ready from previous LED setup.

/\*

\* ======== gpioButtonFxn0 ========

\* Callback function for the GPIO interrupt on Board\_GPIO\_BUTTON0.

\*/

**void** **gpioButtonFxn0**(uint\_least8\_t index)

{

/\* Clear the GPIO interrupt and decrement threshold \*/

**if**(threshold <= 0){ // Ensure threshold doesn't go below zero

threshold = 0;

} **else** {

threshold -= 20; // decrement by 20

}

}

/\*

\* ======== gpioButtonFxn1 ========

\* Callback function for the GPIO interrupt on Board\_GPIO\_BUTTON1.

\* This may not be used for all boards.\*/

**void** **gpioButtonFxn1**(uint\_least8\_t index)

{

/\* Clear the GPIO interrupt and increment threshold \*/

**if**(threshold >= 16383){ // Ensure threshold doesn't go above max ADC range

threshold = 16383;

} **else** {

threshold += 20; // increment by 20

}

}

/\*

Next, we insert code to install the callback functions and enable our interrupts.

/\* install Button callback \*/

**GPIO\_setCallback**(Board\_GPIO\_BUTTON0, gpioButtonFxn0);

**GPIO\_setCallback**(Board\_GPIO\_BUTTON1, gpioButtonFxn1);

/\* Enable interrupts \*/

**GPIO\_enableInt**(Board\_GPIO\_BUTTON0);

**GPIO\_enableInt**(Board\_GPIO\_BUTTON1);

One final additional step that was not covered in the lab instructions was to add code to set up our buttons, configuring their pull (up/down) and directional (rising/falling) values.

**GPIO\_setConfig**(Board\_GPIO\_BUTTON0, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING); //Configuring button pins.

**GPIO\_setConfig**(Board\_GPIO\_BUTTON1, GPIO\_CFG\_IN\_PU | GPIO\_CFG\_IN\_INT\_FALLING);

Below is our stack information and console information.

