LABORATORY MANUAL

Cx1106
Computer Organization and Architecture

Lab Experiment #4

Microprocessor
Integrated Development Environment

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

1. OBJECTIVES

- Using features and tools in an Integrated Development Environment (IDE)
- Use and Analyse Interrupt-based Data Transfer Mechanism
- Explore different Computer Memory sub-system
- Explore Signal Chain sub-system

2. EQUIPMENT

- A Windows-based computer (PC) with a Universal Serial Bus (USB) port.
- Texas Instruments MSP432 Processor Launchpad and EDUMKII daughter board.
- USB micro cable.
- Oscilloscope.

3. LITERATURE READING

- Week8a, 8b and 9a lectures on
  - Computer Organisation Overview
  - Signal Chain Sub-System
  - Interrupts and Polling
  - Semiconductors Memories
- MSP432 Launchpad and EDUMKII Development board. Some online resources
  - https://www.ti.com/tool/MSP-EXP432P401R
  - https://www.ti.com/tool/BOOSTXL-EDUMKII

4. INTRODUCTION

The hardware you will be using in this lab is the MSP432 Launchpad board with the EDUMKII daughter board. The MSP432 is a processor with an ARM Cortex M4F CPU core. The EDUMKII is a daughter card that has a number of input/output devices as illustrated in Fig. 1 below. The labs are adapted from the examples in the Simplelink MSP432 software package. https://www.ti.com/tool/SIMPLELINK-MSP432-SDK

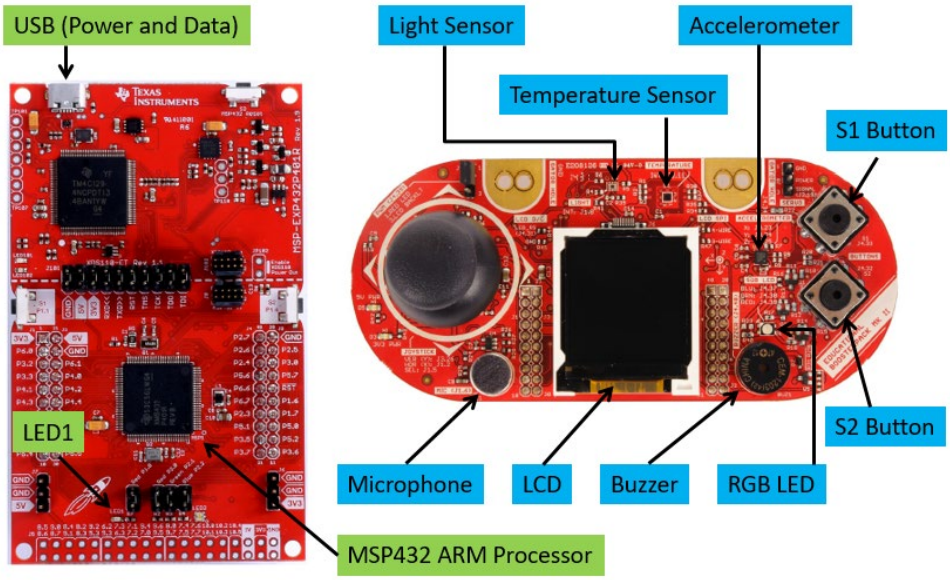

Fig1: MSP432 Launchpad and EDUMKII daughter board.

5. Integrated Development Environment

In the industry, an application generally known as an Integrated Development Environment (IDE) is used to develop code on a microprocessor system. In this lab, we will be using the IDE provide by Texas Instruments known as the Code Composer Studio (CCS). CCS controls the MSP432 processor on the target board directly via a debug module that is integrated onto the MSP432 Launchpad. See Fig. 2 below. Using the debug module, CCS is able to start/halt/stop the MSP432 processor, peek into its memory/registers and perform many other operations. No worries, you need not know the detail connections between the debug module and the MSP432.

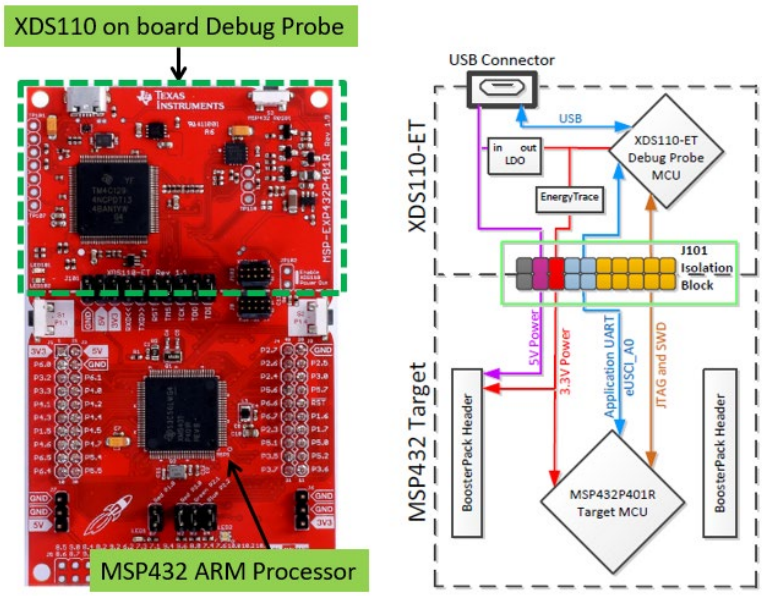

Fig. 2: XDS110 Debug Module

5.1 Connecting the MSP432 Launchpad to the PC

Connect the USB cable to the MSP432 Launchpad and the PC. The board is powered via the USB cable so there is no need to provide additional external power source. On the PC, open the START menu and search for ***Device Manager***. Check that the board has been enumerated successfully and assigned with COM Ports. Fig. 3 illustrates an example that assigns COM3 and COM4 to the MSP432 Launchpad. The actual COM port number may differ on your PC. Unlike Arduino Sketch, CCS will automatically detect the presence of the XDS110 device and connect to it when you run the application.

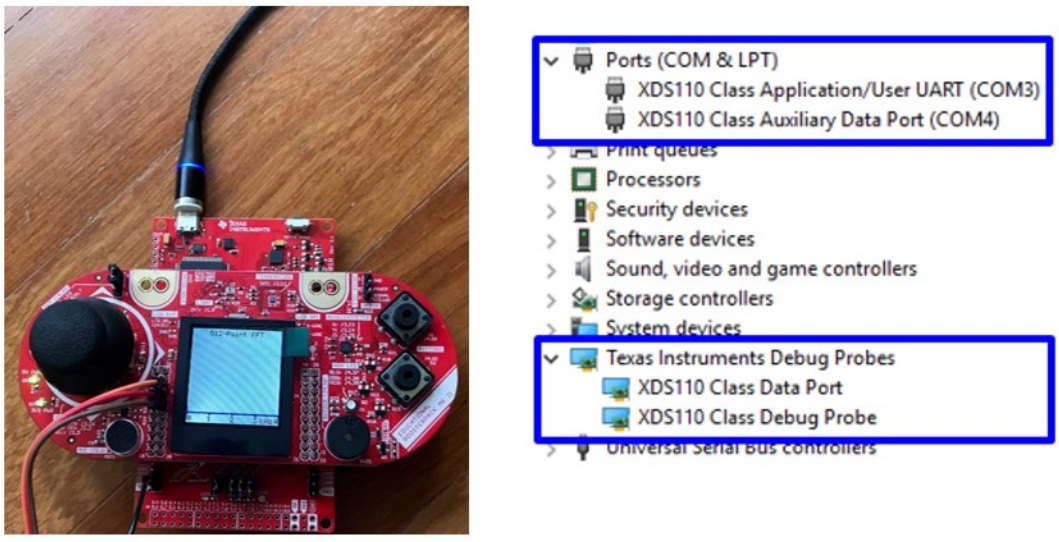

Fig. 3: USB connection and enumeration for MSP432 Launchpad

5.2 Code Composer Studio

When you first start CCS, a window will pop out (Fig. 4) and you will be prompted to select a directory as your workspace, which is basically your working directory. Choose “C:\Cx1106\Labs\Lab4” and click ‘OK’.

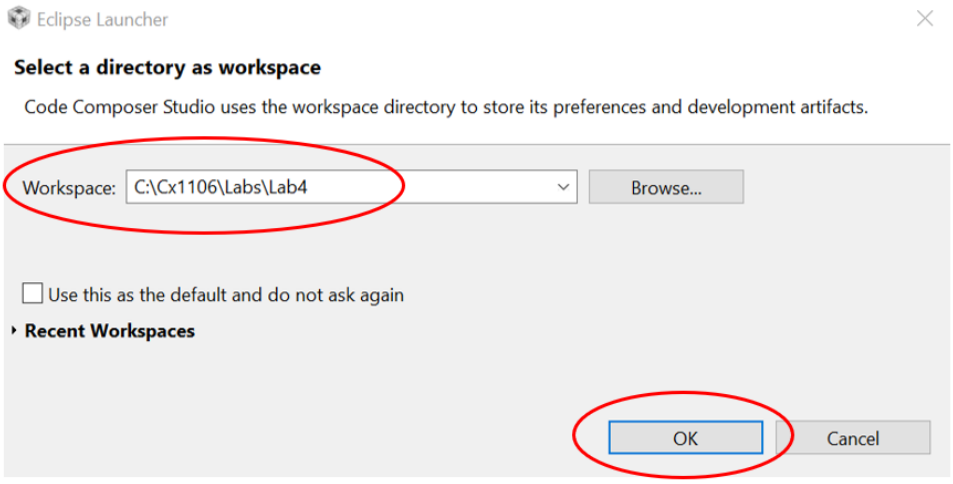

Fig. 4: CCS Workspace Selection

CCS application will come online (Fig. 5) and you should see the projects listed as shown in Fig 6 below. Each project compiles to an application which can be downloaded to the MSP432 Launchpad for execution. In this lab, you’ll be using projects whose name starts with “Lab4_xxx”. Projects with name starting with “Lab4_Optional_xxx” are optional and as per Lab3, lab handouts and quiz will not include specific exercises done in the optional labs.

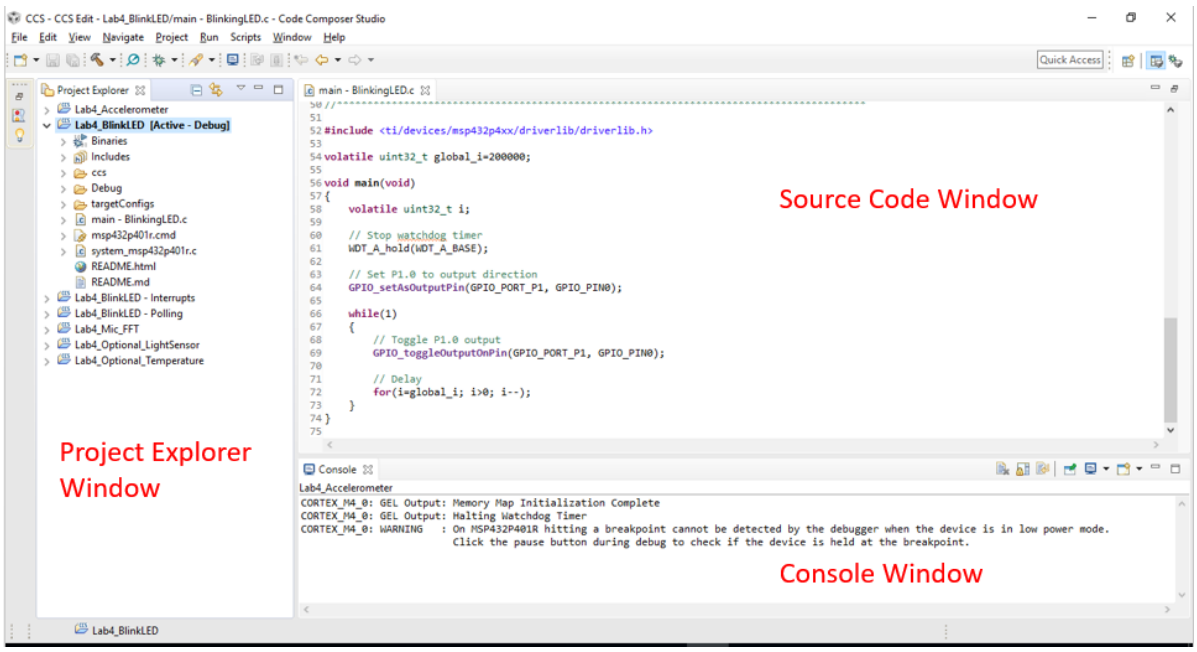

Fig. 5: Code Composer Studio IDE (Edit View)

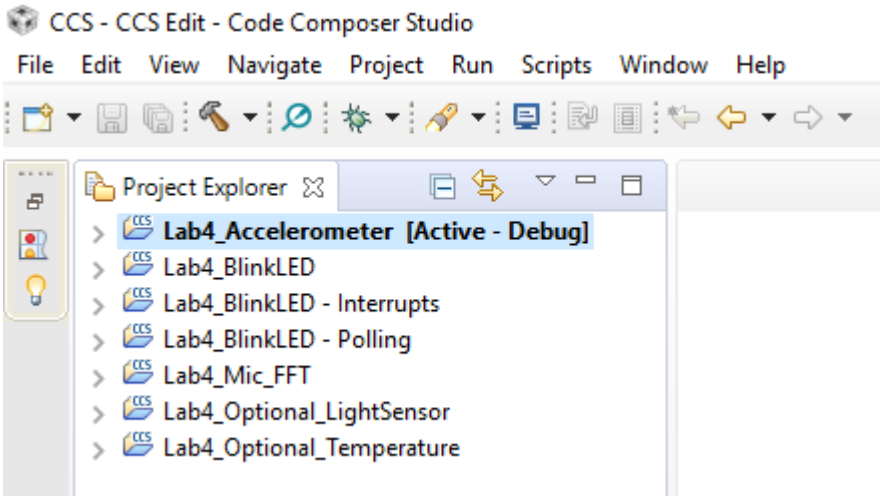

Fig. 6: List of Projects

5.3 Compiling and Downloading code in CCS

You will not be required to write any code in this lab but will need to analyse the code by going through the source code and executing the application.

To build a project, which implies compiling the necessary codes to build the application, go to the 'Project' Drop down menu and select "Clean" followed by "Build Project" (Fig 7). As you get more familiar with the procedure, you may choose to skip the "Clean" step but for now, use the two steps to ensure that you always compile the latest code you have.

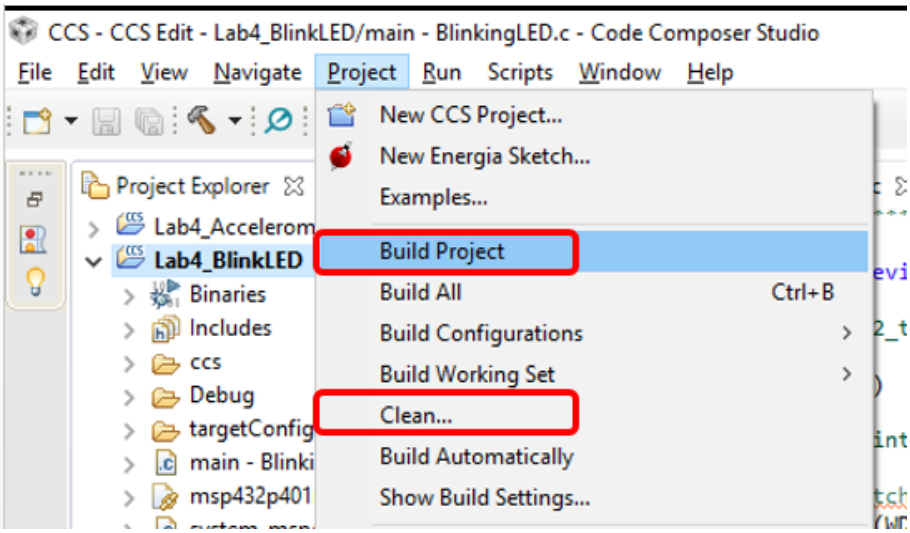

Fig 7: Building a Project in CCS

If the project build completes without error, you can proceed to download the code to the target board for execution/debug. Click on Run->Debug (Fig 8).

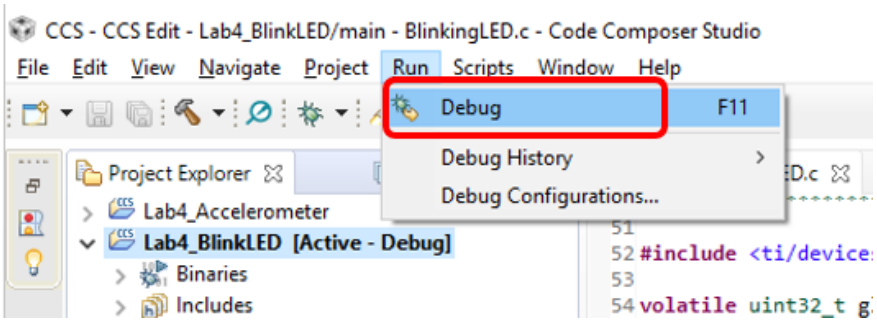

Fig 8: Downloading Code to Target Board

Once the code is downloaded, CCS will switch to the "Debug View" and there will be some changes to the view and layout (Fig 9). There should be a small blue arrow (Program Counter) pointing to the start of the main().

- **Resume**. Run the program from where the Program Counter is pointing to.
- **Suspend**. Halt the program execution temporarily. User can continue execution by clicking on the 'Resume' button.
- **Stop**. Stop the program execution and exit from Debug View.

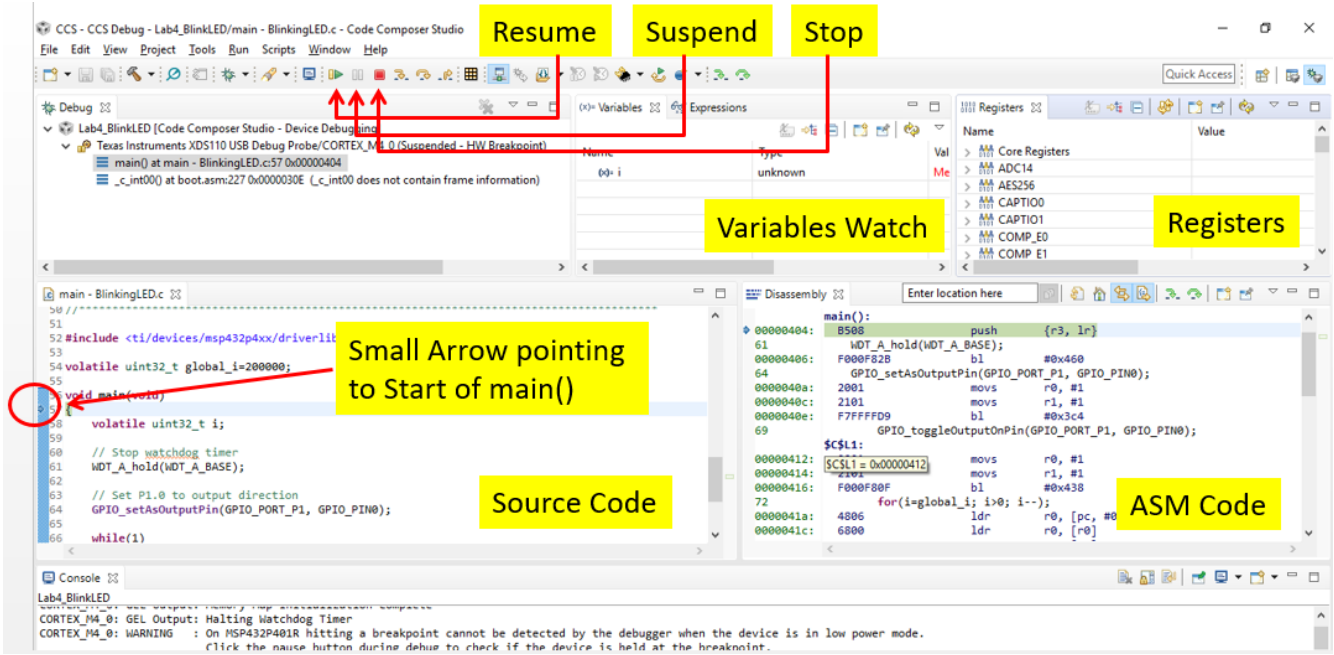

Fig 9: Code Composer Studio IDE (Debug View)

Sometimes, the code might halt at a file which is no in the user project directory, these are typically files from the MSP432 lib and the source file can be found at
C:\ti\simplelink_msp432p4_sdk_2_20_00_12\source\ti\devices\msp432p4xx\driverlib

Direct CCS to this directory when prompted in order for it to locate the necessary files. Code will still work even if you do not point CCS to the relevant directory, just that you will not be able to see the source file.

6. EXPERIMENT

6.1 Blinking LED (yup, I know …)

6.1.1 Overview

This section use a simple LED blinking project to introduce a few key features of an IDE that deal with the operation of a processor, its internal memory and power consumption profile.

6.1.2 Lab4_BlinkLED Project

This project use one of the processor pin (P1.0) to output a square wave (series of logic '1' and '0') to the LED1 (red colour) on the MSP432 Launchpad, giving rise to a blinking effect.

#Task: Build and download the project "Lab4_BlinkLED" into the MSP432 Launchpad. Run the code to check that LED1 is blinking to verify that the code is working properly.

6.1.3 Exploring Processor Internals with CCS

CCS allows you to peek into the internal memory and registers of the processor. The Disassembly is an alternative view for the program memory where the machine code is translated to assembly language for the user. The workspace should have already opened up these windows. If they are missing, you can open them via the 'View' drop down menu as shown in Fig 10 below. Content of these memory/registers will be updated as the code is being executed.

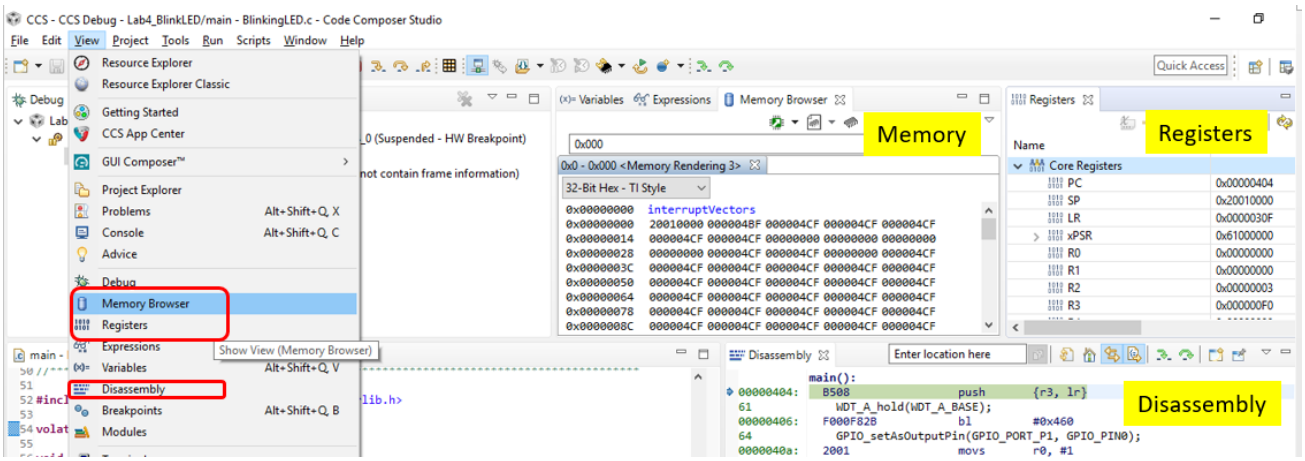

Fig 10: Peek into Processor Internals

Other than running the code via 'Resume' button, you can step through the program one instruction at a time, and you can do that at C statement or ASM instruction level. See Fig 11 below.

- **C-Step-Into**. Step into every C statements and will step into functions encountered.
- **C-Step-Over**. Step into every C statements, will step over any functions encountered. Note that the functions will still be executed though.
- **C-Step-Out**. When in a function, clicking this will executed all C statements before the return statement and control will be passed back to the calling routine.
- **ASM-Step-Into**. Similar to C-Step-Into but operate at ASM instruction level.
- **ASM-Step-Over**. Similar to C-Step-Over but operate at ASM instruction level.

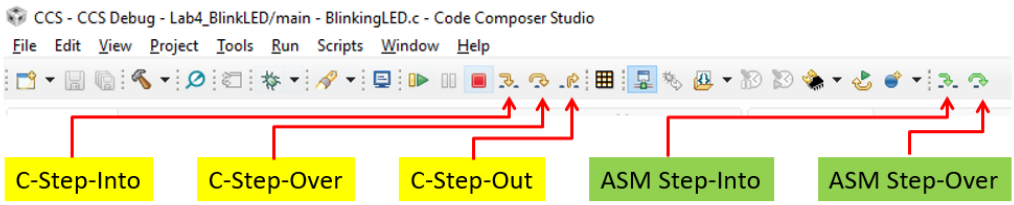

Fig 11: Different methods to Step through the program

#Task: Explore the various features mentioned above.

The processor has two pieces of internal memory block (MEM-A and MEM-B) at location 0x00000000-0x00004000 and 0x20000000-0x20010000. See Fig 12 below.

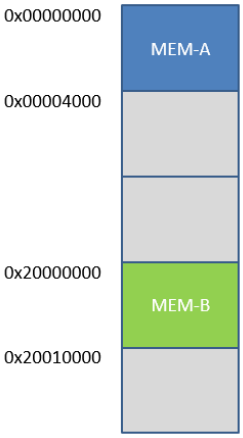

Fig 12: Internal Memory Blocks of MSP432 processor

#Question: Use the CCS memory browser (and some other operations) to find out whether MEM-A and MEM-B is volatile or non-volatile. Make a guess of the memory type of MEM-A and MEM-B.

6.1.4 Energy Trace

#Task: CCS is able to measure the power consumption of the board. To enable energy tracing, go to the Tools drop down menu to select "Energy Trace" (Fig. 13).

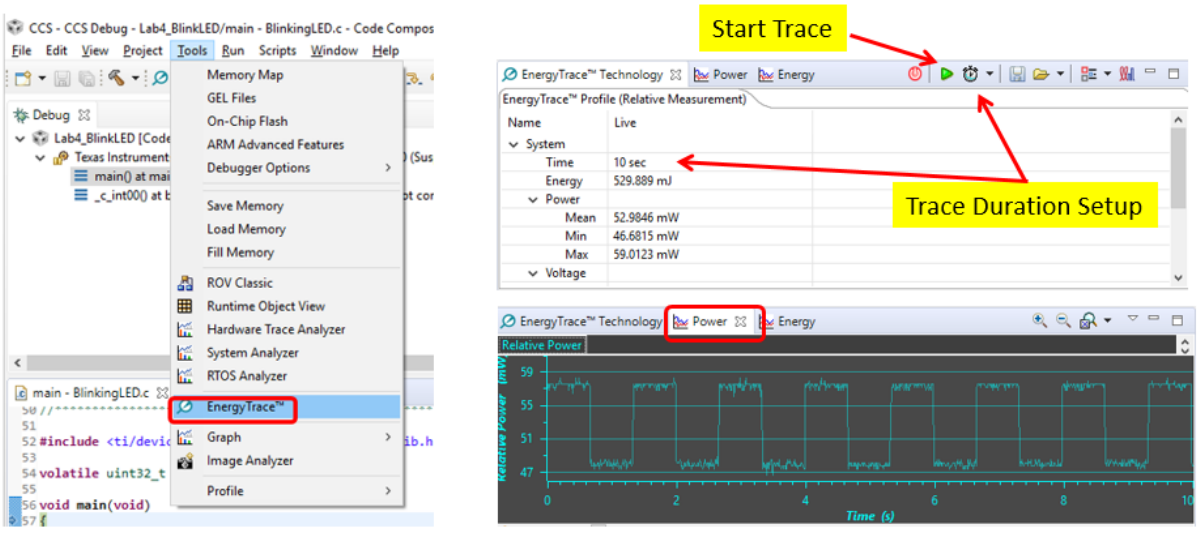

Fig 13: Energy Trace Tool

The energy tracing will start automatically when you click the 'Resume' button to run the program, there is no need to explicitly press the "Start Trace" button on the Energy Trace Window. Default duration is 10sec but you can configure it if needed. Try a trace and you will observe that the power consumption of the board varies with time when running the Lab4_BlinkLED project. Why is that so?

#Question: Explain why the power consumption waveform in Fig13 is a wave with fixed period.

6.1.5 A look at C to ASM translation

#Task: Put a breakpoint at the "for loop" as shown in Fig 14. CCS will stop the processor execution whenever it hits a breakpoint. This feature allows user to halt the processor at a point within the code for analysis purpose. To set a break point, position the cursor at the target source line and press "Ctrl+Shift+B". There are other ways to set a breakpoint within CCS, you are welcomed to explore. A small blue round icon will appear to the left of the line number when the break point is set successfully. See Fig 14.

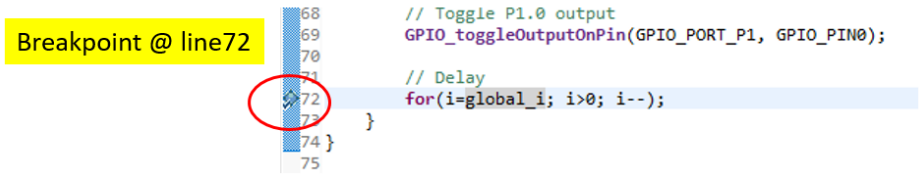

Fig 14: Breakpoint

Hit the 'Resume' button and processor will halt at the "for loop".

The disassembly window shows the actual ASM code that is being executed on the processor (Fig 15), together with the corresponding C statement that the ASM code is implementing. You can see that it takes multiples ASM instructions to implement just one C statement.

#Question: What is purpose of a breakpoint in an IDE?

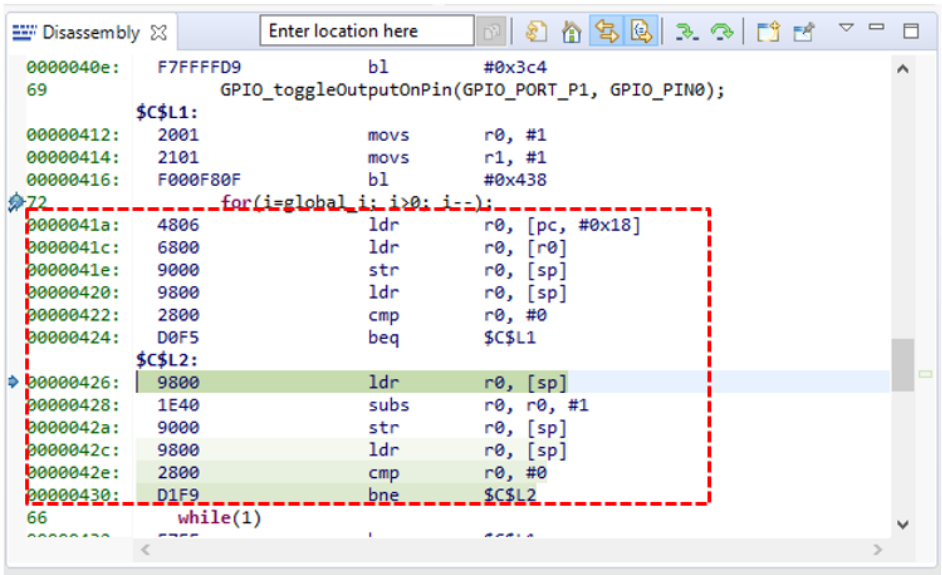

Fig 15: C to ASM

#Question: What is the main purpose of 'r0' in the for loop implementation? And what is the starting address of the GPIO_toggleOutputOnPin() function?

6.2 Polling Mechanism

6.2.1 Overview

In this and the next section, we will look at two projects: Lab4_BlinkLED-Polling and Lab4_BlinkLED-Interrupts to explore the difference between these two data transfer mechanism. Both projects use the two user buttons (S1 and S2) on the daughter board to control LED1 ON/OFF. See Fig 1 for MSP432 Launchpad and EDUMKII Daughter board components illustrations.

In the project code as illustrated in Fig 16, we can see that the software repeatedly poll the pins connected to the S1 and S2 buttons to check their status and will ON/OFF the led if the button input goes low i.e. the button is pressed. In other words, 100% of the CPU time is spend in polling the status of the S1 and S2 buttons.

Note that the connection of the buttons are such that the default level is HIGH and will go LOW if the button is pressed.

```
while(1)
{
    // Poll button S2
    if(!GPIO_getInputPinValue(GPIO_PORT_P3, GPIO_PIN5))
    {
        // Turn LED ON
        GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN0);
        for(i=0;i<debounce_delay;i++);
    }

    // Poll button S1
    if(!GPIO_getInputPinValue(GPIO_PORT_P5, GPIO_PIN1))
    {
        // Turn LED OFF
        GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);
        for(i=0;i<debounce_delay;i++);
    }
}
```

Fig 16: Polling code

#Question: Give one possible reason why the input pin connected to the button has a default HIGH logic value. That is, it reads a logic HIGH when the button is not pressed.

6.2.2 Power consumption

#Task: Build and download the project “Lab4_BlinkLED-Polling”. Press the S1 and S2 button to verify the code. If LED1 turns ON and OFF successively corresponding to the pressing of S2 and S1 buttons, the code is behaving correctly.

#Task-A: With LED1 in OFF state, Halt the processor, enable the Energy Trace, clicked on ‘Resume’ and do a 10sec Energy Trace. Note down the min, max and mean current consumed.

#Task-B: With LED1 in OFF state, Halt the processor, enable the Energy Trace, clicked on ‘Resume’ and do a 10sec Energy Trace. Only difference is that press S1 and S2 repeatedly at approximately 1 second interval when power consumption is being measured. Note down the min, max and mean current consumed.

6.3 Interrupts Mechanism

6.3.1 Interrupts Control Flow

Recall from your lecture that interrupt is a trigger to the CPU asking for its attention. If CPU decides to service a particular interrupt request, it will temporarily halt the current program, fetch the starting address of the interrupt service routine (ISR) handling the interrupt from the interrupt vector table and branch off to the ISR.

6.3.2 Interrupt Vector Table

#Task: Build and download the project "Lab4_BlinkLED-Interrupts". Press the S1 and S2 button to verify the code. If LED1 turns ON and OFF successively corresponding to the pressing of S2 and S1 buttons, the code is behaving correctly.

Put a breakpoint at line 101, i.e. Interrupt_enableInterrupt(INT_PORT5) and Run the program.

```
100
101     Interrupt_enableInterrupt(INT_PORT5);
102
```

In CCS Memory Browser, you can find the interrupt vector table by keying in the label "**g_pfnRAMVectors**". The ISR address of Port3 and Port5 interrupt correspond to the address 0x200000D4 and 0x200000DC respectively. This address is written into the interrupt vector table by the function GPIO_registerInterrupt(). See Fig 17 below.

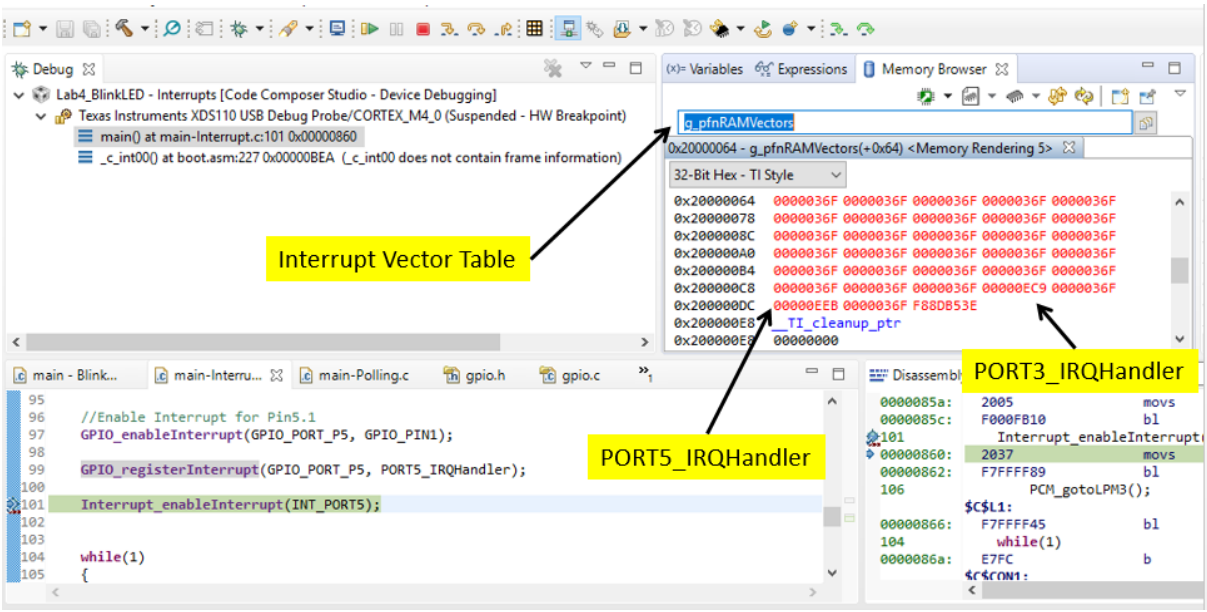

Fig 17: Interrupt Vector Table

6.3.3 Power Consumption

#Task-A: With LED1 in OFF state, Halt the processor, enable the Energy Trace, clicked on 'Resume' and do a 10sec Energy Trace. Note down the min, max and mean current consumed.

#Task-B: With LED1 in OFF state, Halt the processor, enable the Energy Trace, clicked on 'Resume' and do a 10sec Energy Trace. Only difference is that press S1 and S2 repeatedly at approximately 1 second interval when power consumption is being measured. Note down the min, max and mean current consumed.

#Task: Compare the results you obtained in 6.2.2 (Polling) and 6.3.3 (Interrupts).

- **Task-A** represent current consumed when LED is OFF. That of 6.2.2 should be higher than 6.3.3 since 6.3.3 code puts MSP432 into sleep mode (lower power consumption) while 6.2.2 code is actively polling the processor pins. Absolute value for 6.3.3 is still relatively high (more than 10mA) because the debug module of the MSP432 Launchpad still needs to be operating.
- **Task-B** represent actual use case of user pressing buttons. You should notice a difference of about 1-2mA in mean current consumption between 6.2.2 and 6.3.3. This difference between power consumption of polling and interrupt mechanism will be larger if the active power consumption of the processor is higher. Hence, using interrupt mechanism will result in significant power saving.

#Question: Describe what happen (control and data flow) when button S1 is pressed.

6.4 ADC and Accelerometer

6.4.1 Overview

In this section, you will look into the interface between the MSP432 Processor and the Accelerometer Chip. The Accelerometer chip used in the Daughter board output the acceleration information in analog format. The X-, Y- ad Z-axis output will have a different analog voltage level depending on how much resultant force is experienced along individual axis. See Fig 18.

Since acceleration is proportional to force, the value also represent the acceleration information. This analog voltage is picked up by the Analog-to-Digital Converter on the processor and converted to its digitized form. Accelerometer output is commonly used to detect orientation of an equipment. This leverage on the fact that the chip is always act upon by the earth's gravitational force, hence the X-, Y- and Z-axis will experience varying amount of acceleration from the effect of gravity when chip is orientation in different directions.

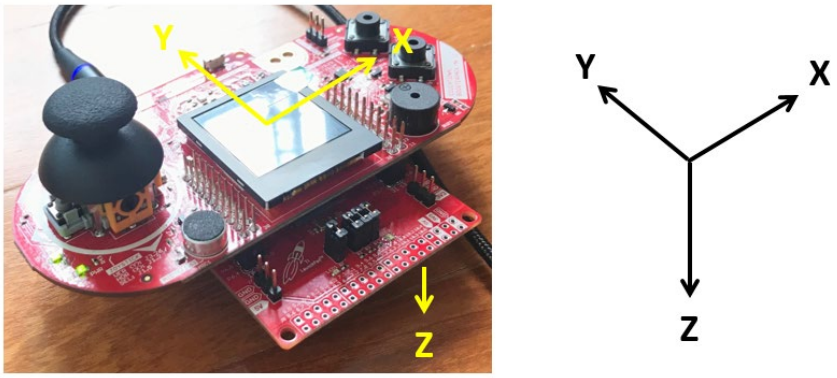

Fig 18: Accelerometer axes wrt daugther board

6.4.2 Analog-to-Digital conversion

The accelerometer chip is a 3V device, from its datasheets, we can find out that it operates between 0-2.4V when detecting +/- 1G acceleration. The ADC on the processor is a 14-bit ADC, i.e. it quantize the analog input into 2^14 levels. The full range of the ADC is determined by the reference voltage it used. For this case, the ADC is configured to use the processor's Analog Voltage Supply (AVcc) as the reference voltage source. AVcc = 3.3V. So a +3.3V signal input to the ADC will be digitized to a value of 0x3FFF (14 bits) and a 0V signal will be digitized to 0x0000. The actual values are subjected to the tolerance of AVcc, ADC accuracy and Accelerometer analog output voltage. Note that the comment in the source code about the ADC using 2.5V as reference is wrong and should be ignored.

#Task: Build and download the project "Lab4_Accelerometer". This project read the analog accelerometer output via the processor chip ADC, digitize it and display the digitized value on the LCD. Check that the X-, Y- and Z-axis value change when the board is in different orientations.

#Task:

- Using a scope to find out the analog voltage corresponding to 1G, -1G and 0G. 'G' refers acceleration due to one unit of earth's gravitational force.
- What is the corresponding digitized value after passing through the ADC?

To observe the signals using a scope, connect the circuit as per Fig 19.

- The three ADC inputs is at P6.1, P4.0 and P4.2 of the process.
- These three pins can be probed at J3.23, J3.24 and J3.25 respectively on the daughter board.
- You can tap a ground signal from any ground pins on MSP432 Launchpad or the daughter board.
- This is a DC signal so which trigger mode should you use?

- And what voltage scale should you use since the accelerometer chip is a 3V device?
- As there are only two probes on each scope, you can only measure two axes at one time.

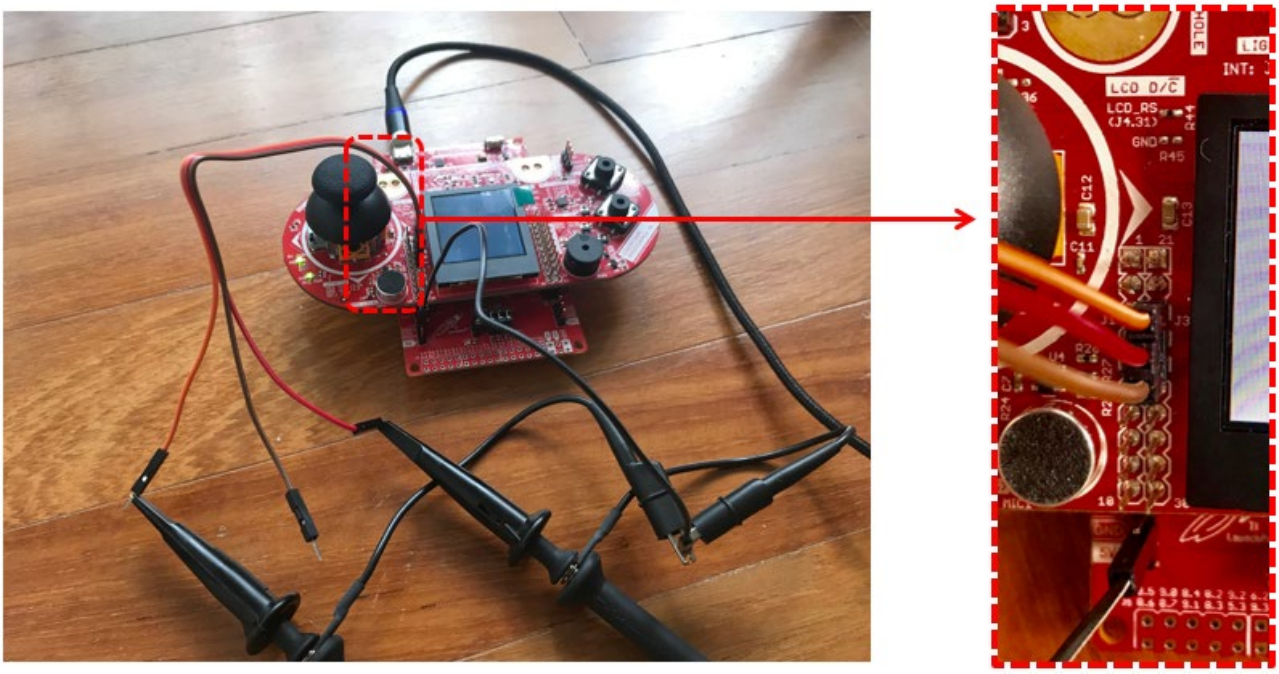

Fig 19: Probe connections to ADC pins and ground

7. Optional Tasks Section

In this optional tasks section, you will explore additional input devices that can be used for different real world applications. Note again that this part will NOT be tested in the Lab quiz.

7.1 Microphone

7.1.1 Overview

In this project, external audio/voice is captured by the microphone and amplified. The amplified analog audio/voice is send to the ADC which sample the audio at 8Khz sampling rate. The digitized data is passed into a module that perform a Fast Fourier Transform (FFT) processing on the audio samples to derive its frequency component, i.e. to find out which frequencies does the audio consist of. The derived frequency is plot onto the LCD. See Fig1 for position of the microphone on the daughter board.

7.1.2 Software

#Task: Build and download the project "Lab4_Optional_Mic". Check that the LCD would display the frequency components detected when you whistle or speak into the microphone.

#Task: The program uses DMA mechanism to transfer data. Look through the comments in the code and answer the following questions.

- How did the analog audio get into the processor and digitized?
- In what manner is the digital audio processed to derive its frequency components? Single sample? Or multiple samples at a time?
- Which part of the data transfer is the DMA involved?
- What is the maximum frequency that can be displayed on the LCD? Is this also the limit of what this project could detect? Which theorem is this limitation associated to?

7.2 Light Sensor

7.2.1 Overview

This project connects a light sensor to the processor via a serial bus known as the I2C bus. It's a 2-line synchronous serial bus that supports a topology of multi-masters, multi-slaves. A very popular serial bus interface for sensor IC chips. Since its I2C so digital data is transferred, which means the transducing of the light intensity (analog) to digital illuminance value is done in the light sensor IC chip. The illuminance value (in lux) is displayed on the LCD.

#Task: Build and download the project "Lab4_Optional_LightSensor". Check the position of the light sensor from Fig 1. If program is successfully loaded and run, the lux value will be displayed on the LCD and should change when you cover the light sensor with your finger. The brightness of RGB LED on the daughter board should also vary according to the light intensity sensed.

#Task: Try to modify the code such that the RGB brightness will be completely OFF when surroundings are 'relatively' bright and will turn ON progressively when the ambient light is low. This is what is done for some of the street lights where the lights will be turn ON when environment gets dark.

7.3 Temperature Sensor

7.3.1 Overview

This project connects a temperature sensor to the processor via the I2C bus. Unlike most temperature sensors, the TMP006 sensor on the daughter board does not require contact with the object it is measuring. It uses a very sensitive thermopile to measure the infrared energy being emitted from the surface of the object.

#Task: Build and download the project “Lab4_Optional_Temperature”. Check the position of the temperature sensor from Fig 1. If program is successfully loaded and run, the temperature detected (in deg F) will be displayed on the LCD. Is the measured temperature accurate? If not, why?

#Task: Try converting the Deg F display to Deg C, and sound off the buzzer if the temperature goes beyond a certain threshold. You have just made yourself a temperature detector for fire hazard or temperature screening!