The Mechatronics Revolution: Fundamentals and Core Concepts
Lab Assignment 1

Getting started with Code Composer Studio and MSP432 LaunchPad

Contents

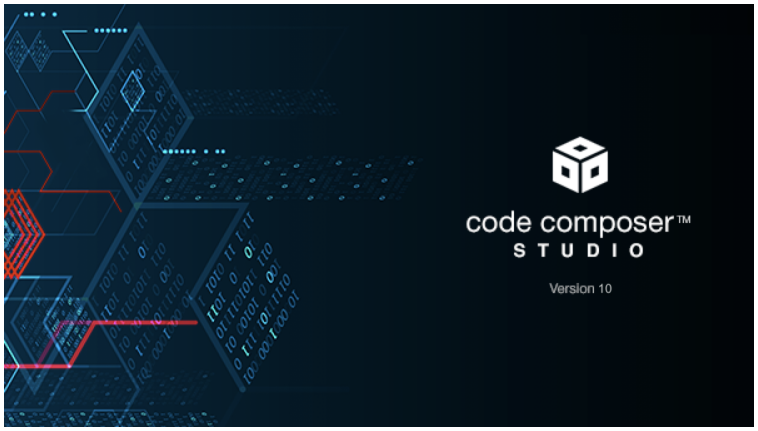

Figure 1: Code Composer Studio (CCS) IDE (Image courtesy of Texas Instruments).

1.1 Objective

The objective of this lab is to get acquainted with Code Composer Studio (CCS) and the MSP432 LaunchPad, which will be used throughout the course and the lab assignments. This lab covers installing Code Composer Studio, loading your first program on the MSP432 LaunchPad, and debugging the program using the debug features of the CCS IDE.

Note: An Integrated Development Environment (IDE) is a software environment to develop and debug embedded applications using tools like a source code editor, debugger, and compiler. Code Composer Studio is an IDE that supports TI's microcontrollers, such as MSP432. Learning how to use an IDE can be considered as the first step before using an embedded platform, such as the MSP432 LaunchPad.

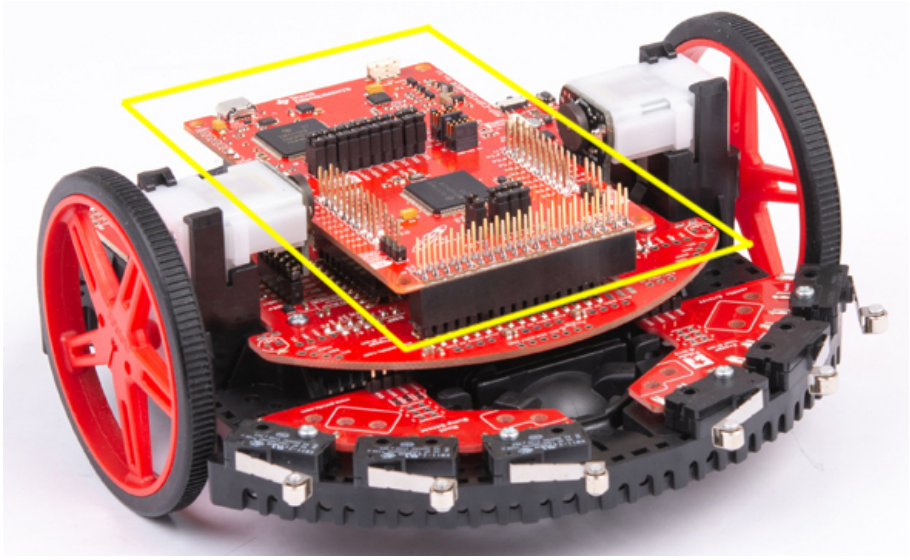

Figure 2: The TI-RSLK-Mechkit includes a MSP432 LaunchPad.

1.2 Setup

This lab requires *Code Composer Studio* and the *MSP432P401R LaunchPad*. You can use the LaunchPad and the USB debug cable included in the TI-RSLK-Mechkit.

As you will install CSS and setup your *workspace* in this lab, it is recommended to use the same computer that you will be using throughout the course.

An example `main.c` source file and a `mechrev.h` header file along with the DriverLib library are provided with this lab assignment, available under Course Resources on edX (lab1-main.zip, mechrev-header.zip and driverlib.zip). You will work with the provided example code after installing the Code Composer Studio.

1.3 Procedure

1.3.1 Installing CCS

Download and install the latest version of Code Composer Studio on your computer by following the installation instructions available at:
http://software-dl.ti.com/ccs/esd/documents/users_guide/index_installation.html
The installation images for Code Composer Studio can be obtained from the CCS download site: http://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html

You can use either the *On-demand (web) installer* or the *Single file (offline) installer*. We recommend to use the On-demand installer, as it will allow you to download only the software components that you require for the recourse. Internet connectivity is required to use the web installer.

Overall, the installation process is the same across Windows, Linux and macOS. Where there are differences, it is noted in the installation steps.

Note: For the "Processor Support" installation step, you only need to select *"SimpleLink MSP432 low power + performance MCUs"* product family to be installed on your computer (as shown in Figure 3).

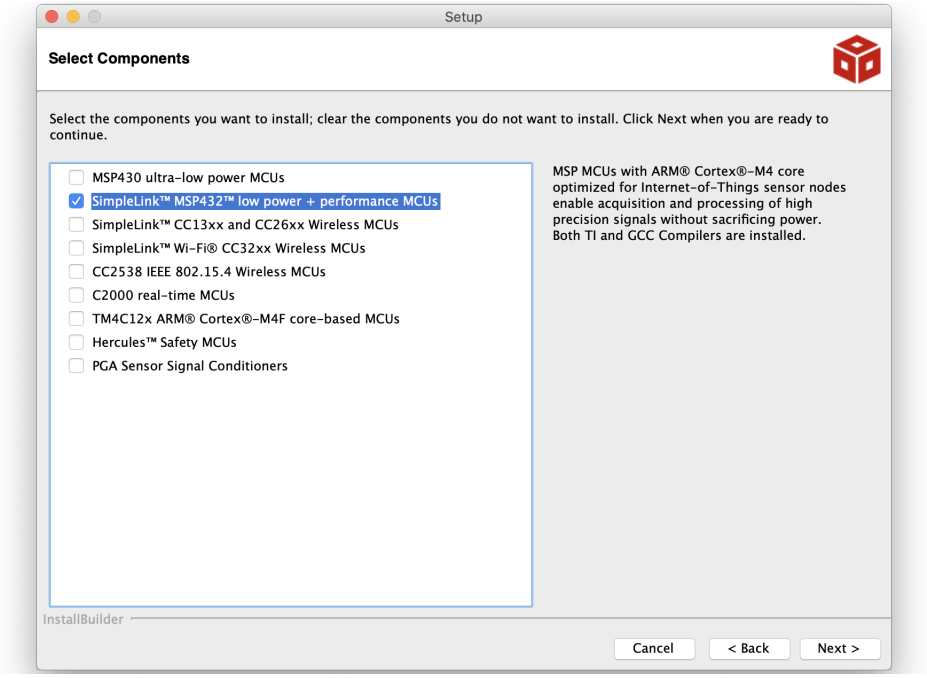

Figure 3: Processor Support installation step.

Note: For the "Debug Probes" installation step, you only need to select *"Spectrum Digital Debug Probes and Boards"* (as shown in Figure 4).

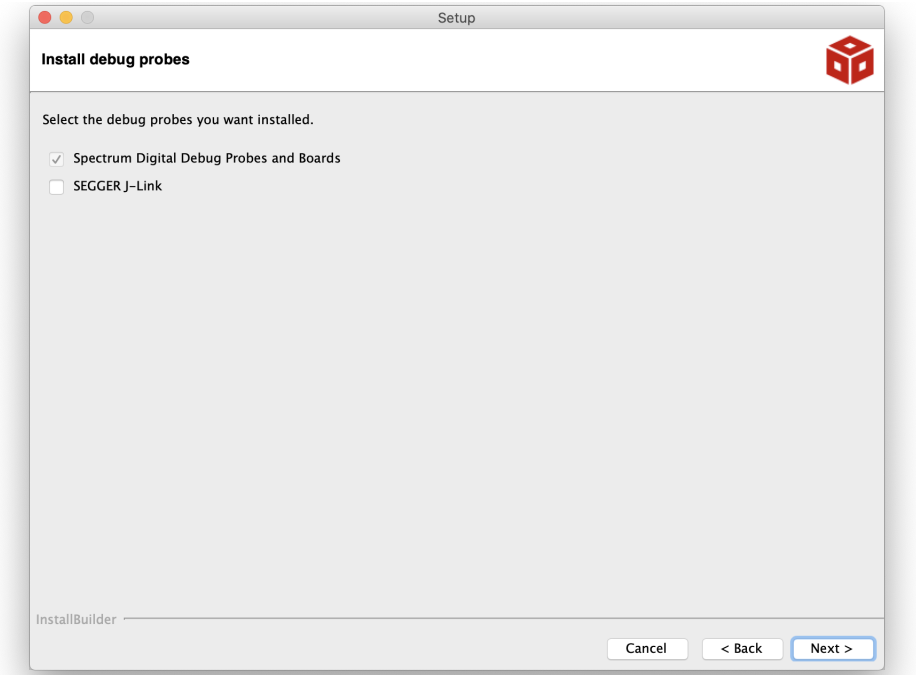

Figure 4: Debug Probes installation step.

1.3.2 Creating a new CCS project

- Launch Code Composer Studio on you computer.
- A new window may pop up which asks you to enter a path for the workspace. All the projects and files that you create from here on will be saved at this location. Enter an appropriate address/location. It is recommended to create a new dedicated folder which will serve as your workspace for this course (Figure 5).

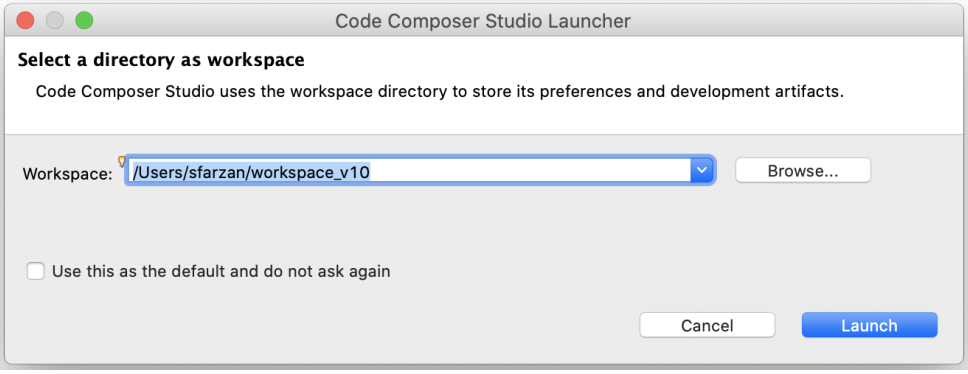

Figure 5: Select a directory as workspace.

- An empty CCS workbench will open with a Getting Started Window.
- Create a new CCS project by following the steps below:
  - Go to menu bar: `Project` → `New CCS Project...` or `File` → `New` → `CCS Project`
  - In the New CCS Project wizard (shown in Figure 6), type or select the Target device as `MSP432P401R`.
  - Type in a Project name, for example `Lab1`. By default, the project will be created inside the workspace directory.
  - Lastly, select the Project template/example as `Empty Project (with main.c)`, and click Finish.

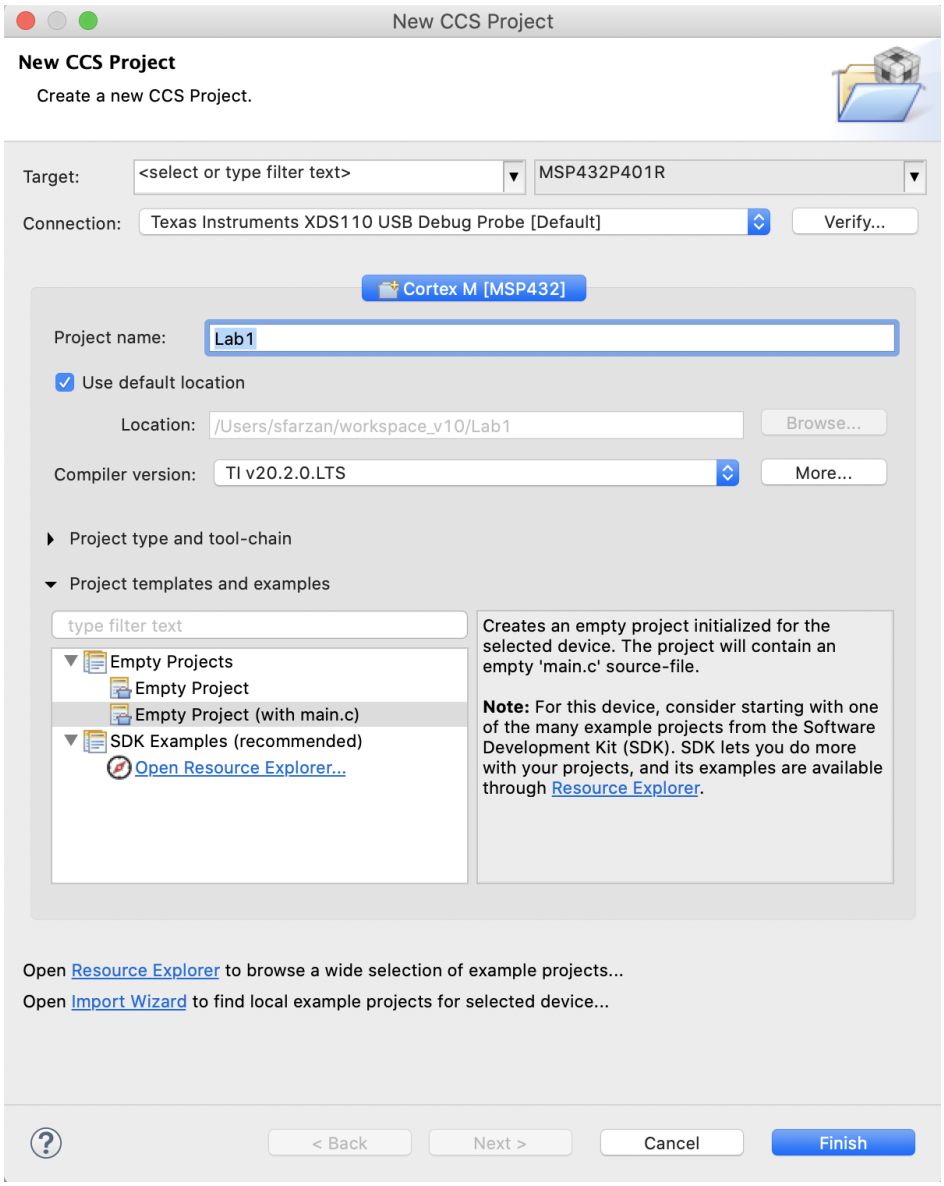

Figure 6: Creating a new CCS project.

- The new project will appear in the Project Explorer view and will be set as the currently active project. A main.c file opens with a part of code already written in the editor.

1.3.3 Including the DriverLib library and the `mechrev.h` header file in the project

- Download the `main.c` file, `mechrev.h` file and `driverlib` folder provided with the lab assignment (available under Course Resources on edX (lab1-main.zip, mechrev-header.zip and driverlib.zip)) and unzip them. Details on how to unzip files and folders are described in the Labs Helper Guide, also available under Course Resources on edX (Labs-Helper-Guide.pdf).

- On your computer, find the directory (folder) where the project is created (under the workspace folder). Copy and paste the `main.c` file, the `mechrev.h` header file and the `driverlib` folder into the Lab 1 project folder under the workspace. The original `main.c` source file will be replaced with the new one (Figure 7).

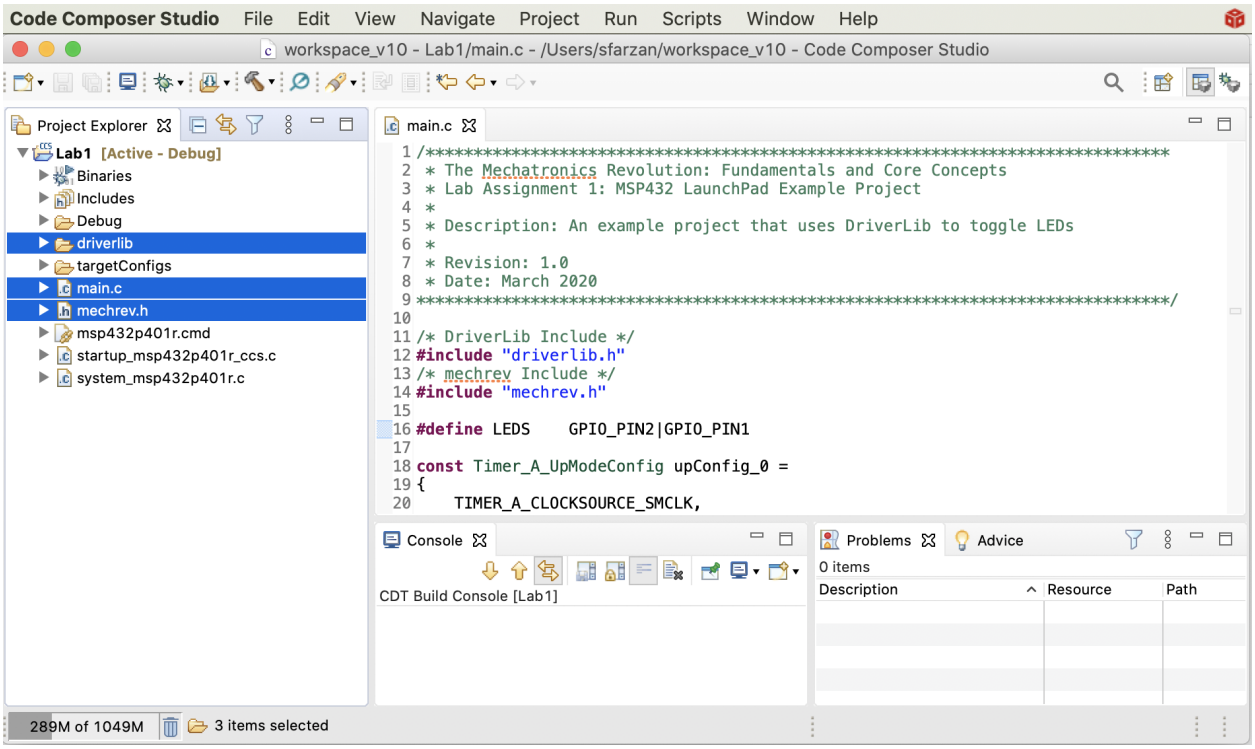

Figure 7: `main.c`, `mechrev.h` and `driverlib` copied into the project.

- Make sure that the following lines are added to the beginning of the `main.c` code:

```
#include "driverlib.h"
#include "mechrev.h"
```

1.3.4 Modifying the search path to include the `driverlib` folder

Add the `driverlib` folder to the `#include` search path by following the steps below:

- Go to menu bar: `Project` → `Properties`, and then on the left column of the window (shown in Figure 8), select `Build` → `ARM Compiler` → `Include Options`.
- On the right side of the window (shown in Figure 8), click the add icon, select `Workspace...`, and select the folder `Lab1/driverlib/MSP432P4xx` (Figure 9).

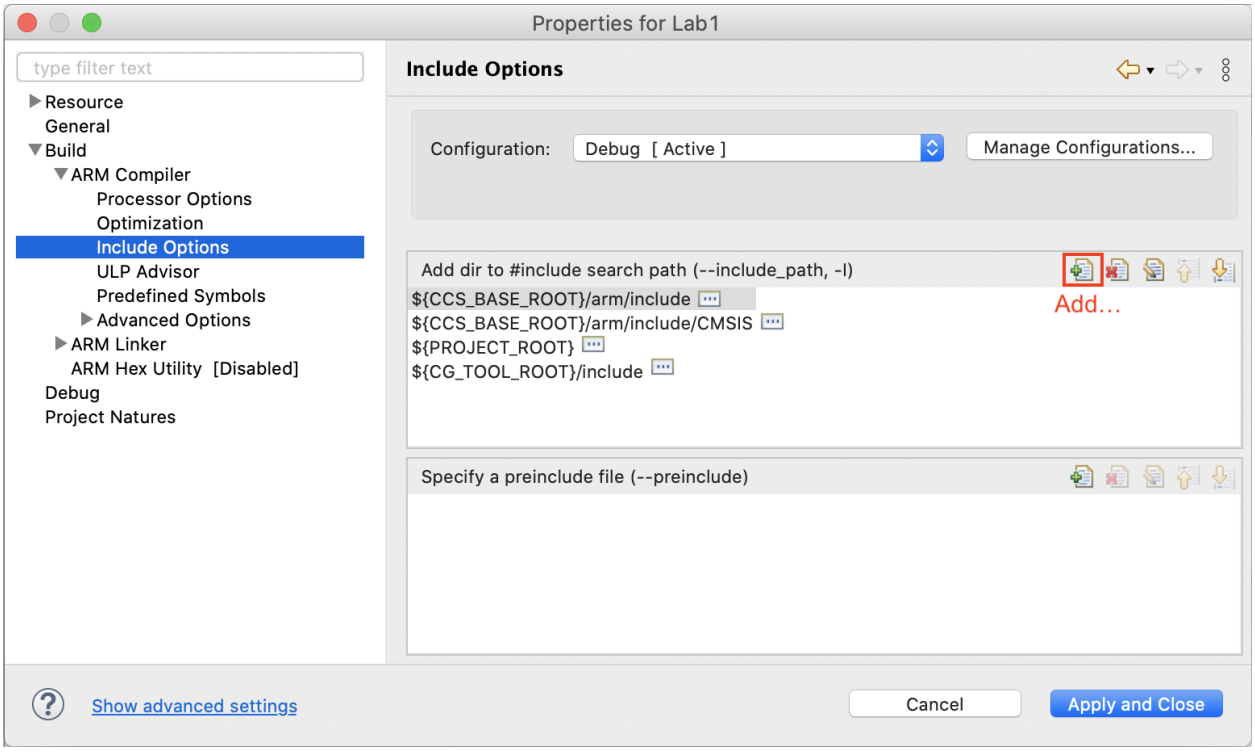

Figure 8: Adding the "driverlib" folder to `#include` search path, using the `Include Options` build properties.

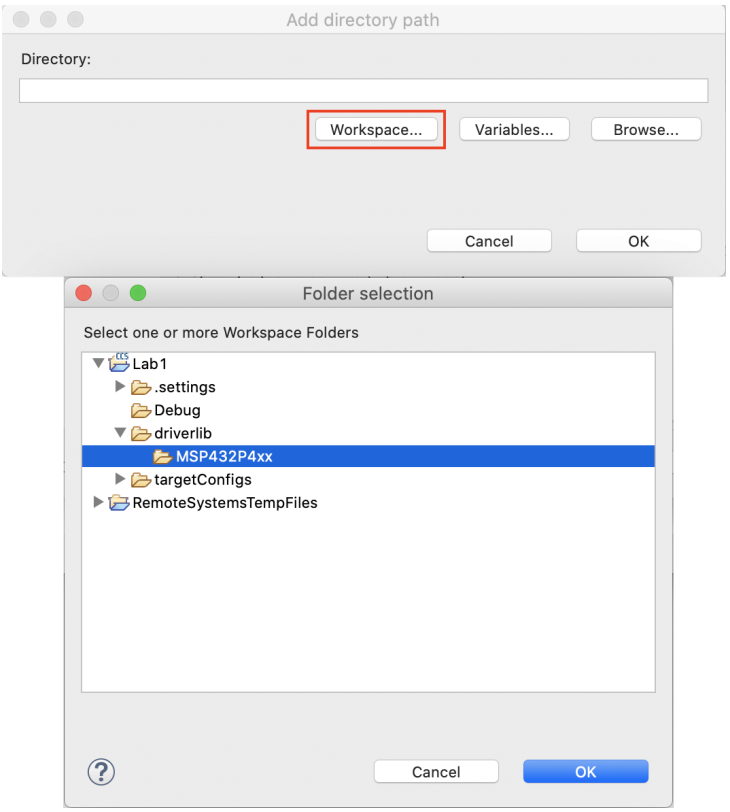

Figure 9: Select the `Lab1/driverlib/MSP432P4xx` folder to be included in the search path.

- Select OK to add `$workspace_loc:/$ProjName/driverlib/MSP432P4xx` directory to the include options. Select "Apply and Close" to close the window.
  The `/driverlib/MSP432P4xx` folder is added to the `#include` search path (Figure 10).

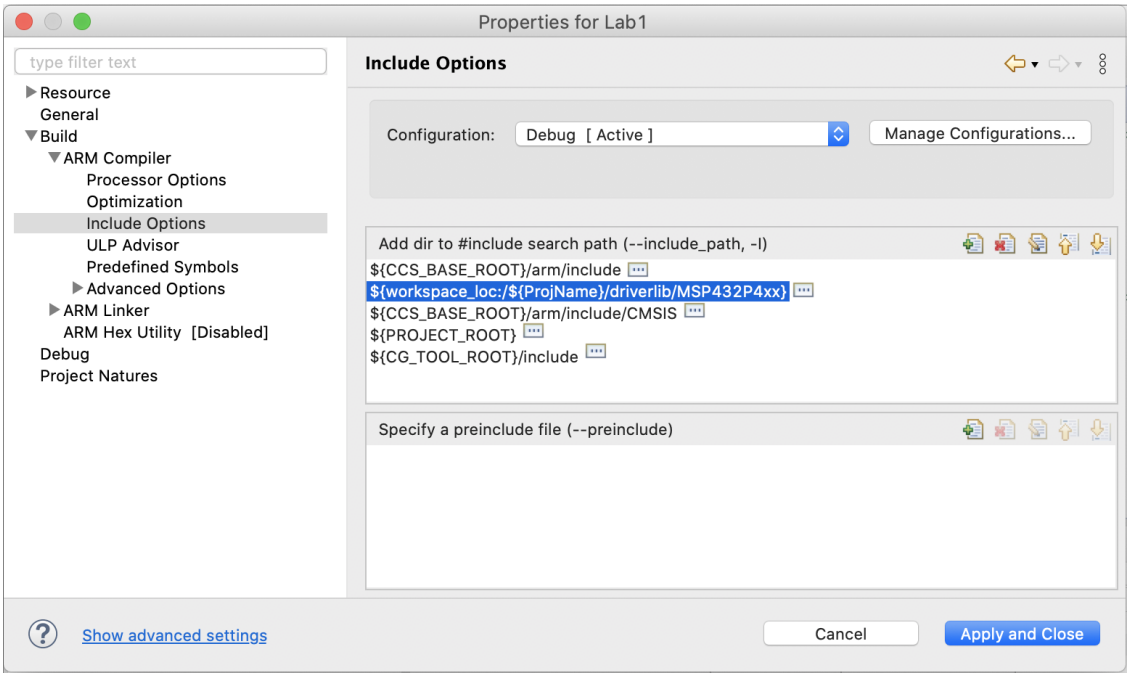

Figure 10: The `/driverlib/MSP432P4xx` folder added to the `#include` search path.

To start using the MSP432 Launchpad, connect the board to the computer with the provided USB cable.

Note: We will discuss the DriverLib library in more details in Lab Assignment 3.

Note: The Code Composer Studio User's Guide can be found at:
http://software-dl.ti.com/ccs/esd/documents/users_guide/index.html

1.3.5 Building (compiling) the project and programming the MSP432 Launchpad

In order to program a project into the MSP432 Launchpad, we need to compile all the files and "build" the project. This can be done by following the steps below.

- Build the project via the menu bar: `Project` → `Build Project`, or click on the Build button on the Quick Access panel below the menu bar (Figure 11). The project should build without errors.

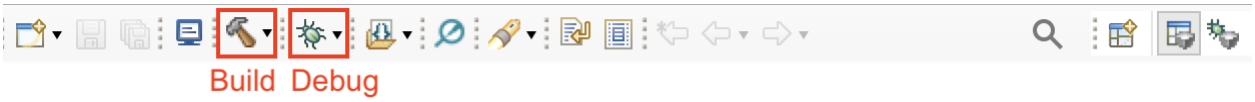

Figure 11: Build and Debug icons on the Quick Access panel.

- Make sure that the LaunchPad is connected to the computer with the provided USB cable. Debug the application using the menu bar: `Run` → `Debug`, or click on the Debug button on the Quick Access panel below the menu bar (Figure 11). This starts the debugger, which gains control of the target, erases the target memory, programs the target memory with the application, and resets the target.
- The project is now flashed onto the MSP432. This code will now execute each time the controller is powered unless it is flashed with a new code in a similar manner.
- The perspective of the CCS will change. The CCS is now in Debug mode (Figure 12). Additional buttons are now available in the Quick Access panel as the Target Execution toolbar.

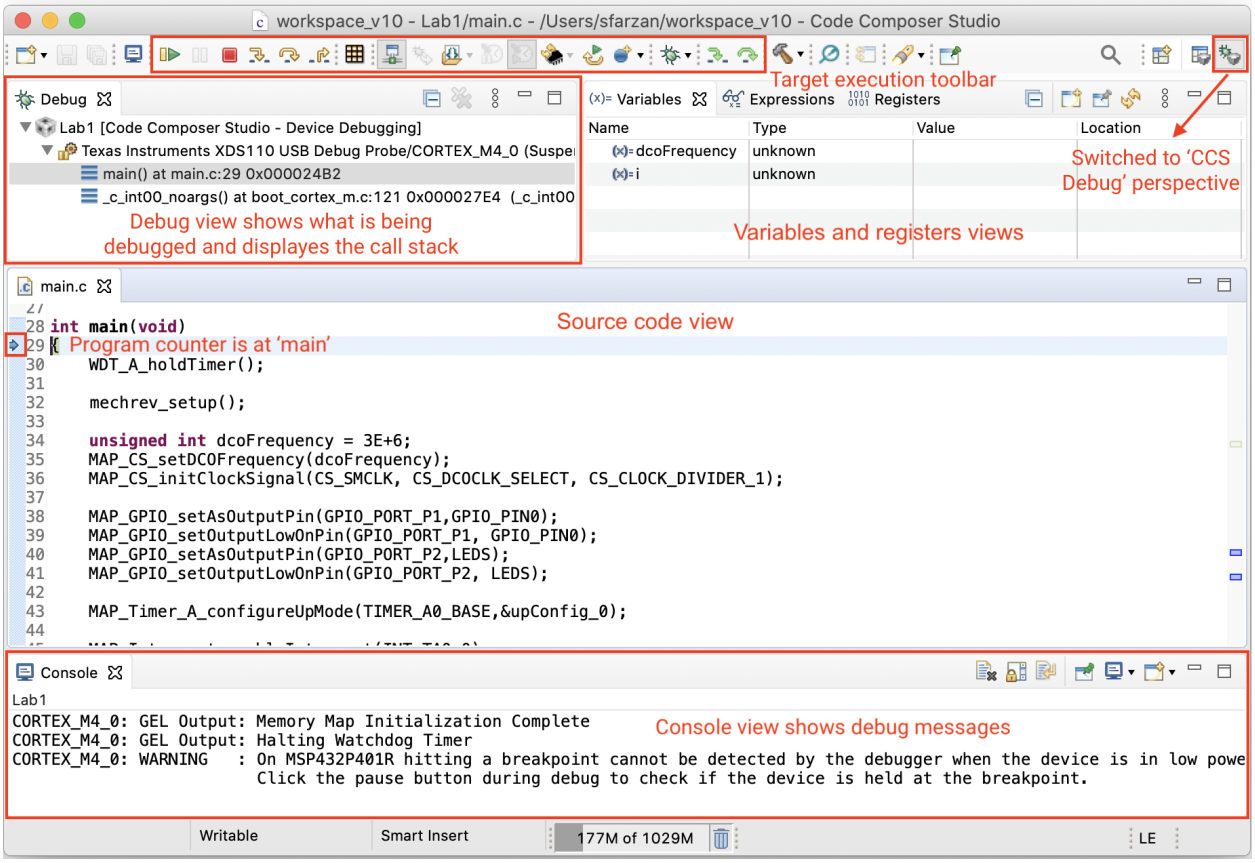

Figure 12: Debug mode perspective and the associated windows/toolbars.

- The code (program counter) is stopped at the first line of the main function. It will continue execution if the Resume button on the Target Execution toolbar is pressed.

1.3.6 Debugging and pausing the program

Figure 13: Target Execution toolbar.

- While in the debug mode, press the Resume button on the Target Execution toolbar (Figure 13), Both the LEDs on the board will start blinking. The Suspend button will suspend the execution of the code at current step.
- The debug mode perspective can also be used to check the values of variables and registers, and also place breakpoints during debugging. It may also be observed that under the variables tab, the variable i changes value every time the program execution is paused.

- When paused, code can be executed step by step using the Step buttons, as described below.
- Step by step execution causes the variables and registers whose values have changed to be highlighted in yellow.
- Basic debugging functionality is provided by the Target Execution toolbar (Figure 13):
  - *Resume*: starts/resumes execution of the target core.
  - *Suspend*: halts the execution of the target core.
  - *Terminate*: disconnects from all hardware (cores, devices, Debug Probes) and terminates the Debug Session. Closes the CCS Debug perspective and returns of the CCS Edit perspective.
  - *Step into*: executes a single C source line, jumping into subroutines or functions, allowing to run its internal code step-by-step.
  - *Step over*: similar as above, but jumping over subroutines or functions, running its internal code at once.
  - *Step return*: similar as above, but running all lines, subroutines or functions until it reaches the caller function (given by the call stack or return()).

1.3.7 Setting breakpoints, checking variables and registers

- To set up the debug session to answer the Problems for Lab Assignment 1 on edX, terminate the debugging session (if you are running one).
- Set breakpoints at lines 51 and 56 of the code. To set a breakpoint, simply "double-click" on the left side (blue area) of the line number in the source code view (Figure 14). A blue circle icon will indicate the breakpoint status and placement.
  You can also right-click on a specific line of the code and select "Breakpoint" from the opened menu.
- Now press the Debug button. Once the code is paused in the main function, press F8 (or the Resume button) to progress to line 51. The program should pause execution at line 51, as shown in Figure 14.
- Click on the Variables tab in the watch window as shown in Figure 15. Two variables should be visible: `dcoFrequency` and `i`.
- The latest register values can be checked in the registers window embedded in the screen (Figure 16). For example, it may be observed that the `TAR` register under the `Timer_A0` changes value each time it is paused. This is the register that holds current value of the timer.

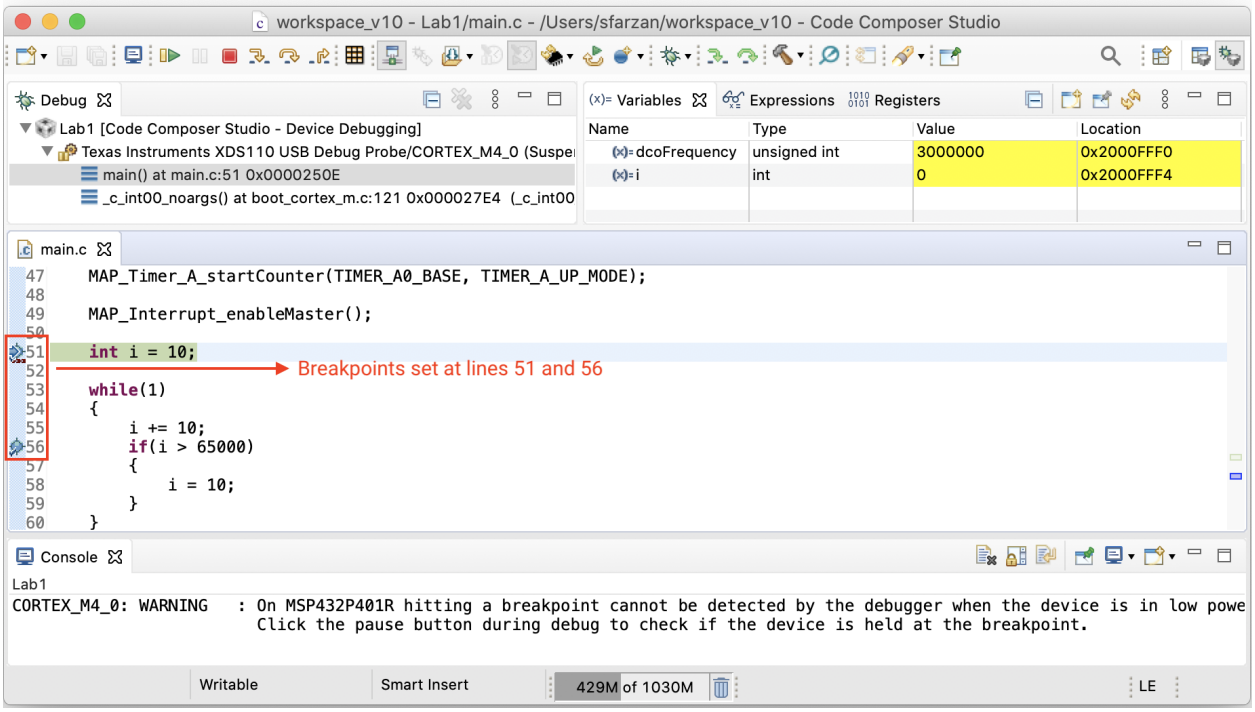

Figure 14: Setting breakpoints in debug mode.

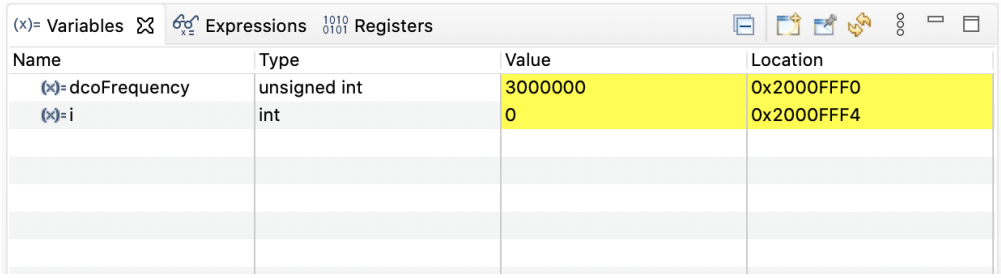

Figure 15: Variables window in debug mode.

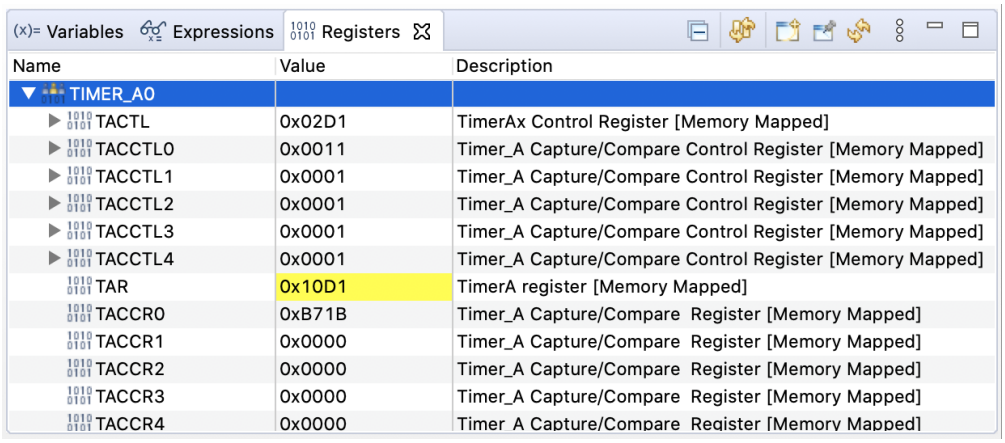

Figure 16: Registers window in debug mode.

1.4 Grading

Proceed to the Problems unit of Lab Assignment 1 on edX and answer the questions.