Introduction to Mechatronics (ME/AE 6705)
Lab Assignment 2

Getting started with Code Composer Studio and MSP432 LaunchPad

Contents

2.1 Objective 2
2.2 Deliverables and Grading 2
2.3 Setup 2
2.4 Procedure 3
- 2.4.1 Installing CCS 3
- 2.4.2 Creating a new CCS project 4
- 2.4.3 Including the DriverLib library in the project 6
- 2.4.4 Modifying the search path to include the `driverlib` folder 6
- 2.4.5 Building (compiling) the project and programming the MSP432 Launchpad 9
- 2.4.6 Debugging and pausing the program 10
- 2.4.7 Setting breakpoints, checking variables and registers 11

2.5 Questions 13

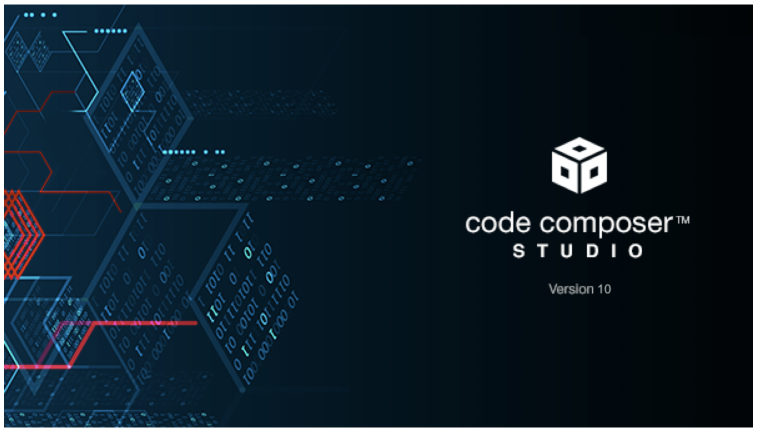


Figure 1: Code Composer Studio (CCS) IDE (Image courtesy of Texas Instruments).

2.1 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 rest of 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.

2.2 Deliverables and Grading

To get credit for this lab assignment you must submit a typed report as a PDF file to Canvas, answering the questions at the end of the lab assignment (4 pages max, can be shorter). This is due at the beginning of class on the due date specified on Canvas. (80 points)

2.3 Setup

This lab requires *Code Composer Studio* and the *MSP432P401R LaunchPad*. 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 the DriverLib library are provided with this lab assignment, which are available on Canvas. You will work with the provided example code after installing the Code Composer Studio.

2.4 Procedure

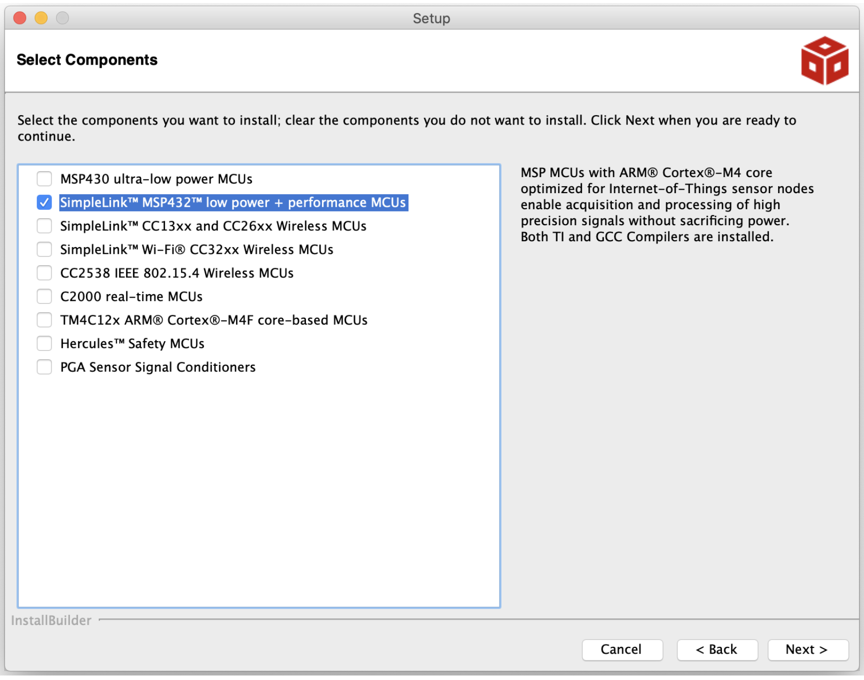
2.4.1 Installing CCS

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 course. 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 2).



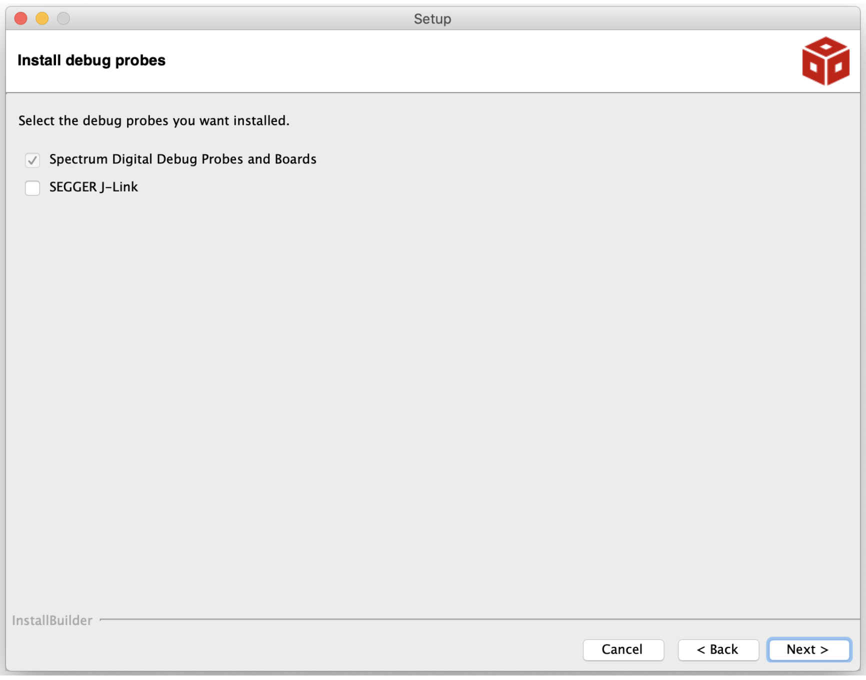
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] MSP430 ultra-low power MCUs
- [✓] SimpleLink™ MSP432™ low power + performance MCUs
- [ ] SimpleLink™ CC13xx and CC26xx Wireless MCUs
- [ ] SimpleLink™ Wi-Fi® CC32xx Wireless MCUs
- [ ] CC2538 IEEE 802.15.4 Wireless MCUs
- [ ] C2000 real-time MCUs
- [ ] TM4C12x ARM® Cortex®-M4F core-based MCUs
- [ ] Hercules™ Safety MCUs
- [ ] PGA Sensor Signal Conditioners

MSP MCUs with ARM® Cortex®-M4 core optimized for Internet-of-Things sensor nodes enable acquisition and processing of high precision signals without sacrificing power. Both TI and GCC Compilers are installed.

Figure 2: 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 3).



Install debug probes

Select the debug probes you want installed.

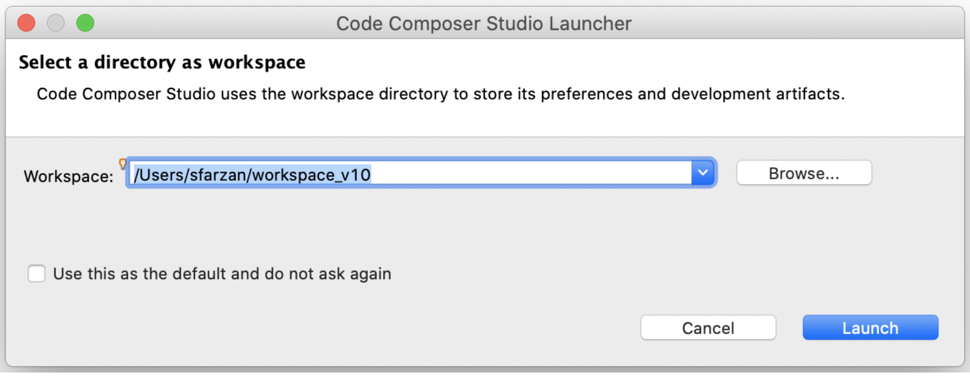
[✓] Spectrum Digital Debug Probes and Boards
[ ] SEGGER J-Link

InstallBuilder

Figure 3: Debug Probes installation step.

2.4.2 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 4).



Select a directory as workspace

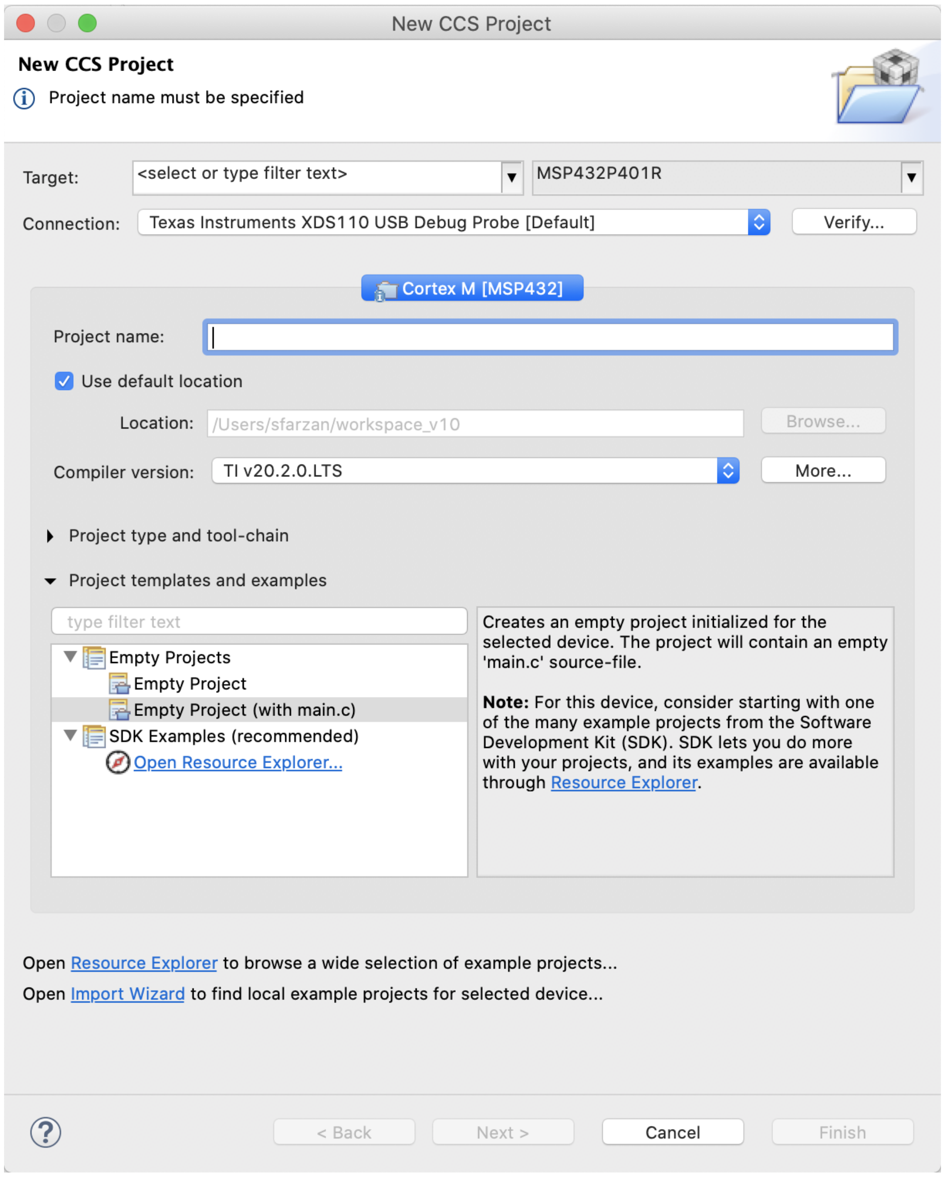
Code Composer Studio uses the workspace directory to store its preferences and development artifacts.

Workspace: /Users/sfarzan/workspace_v10

[ ] Use this as the default and do not ask again

Figure 4: 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 5), type or select the Target device as `MSP432P401R`.
  - Type in a Project name, for example `Lab2`. 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.


New CCS Project

New CCS Project

Project name must be specified

Target: <select or type filter text> MSP432P401R

Connection: Texas Instruments XDS110 USB Debug Probe [Default] Verify...

Cortex M [MSP432]

Project name:

☑ Use default location

Location: /Users/sfarzan/workspace_v10 Browse...

Compiler version: TI v20.2.0.LTS More...

Project type and tool-chain

Project templates and examples

type filter text

Empty Projects
- Empty Project
- Empty Project (with main.c)

SDK Examples (recommended)
- Open Resource Explorer...

Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Note: For this device, consider starting with one of the many example projects from the Software Development Kit (SDK). SDK lets you do more with your projects, and its examples are available through Resource Explorer.

Open Resource Explorer to browse a wide selection of example projects...

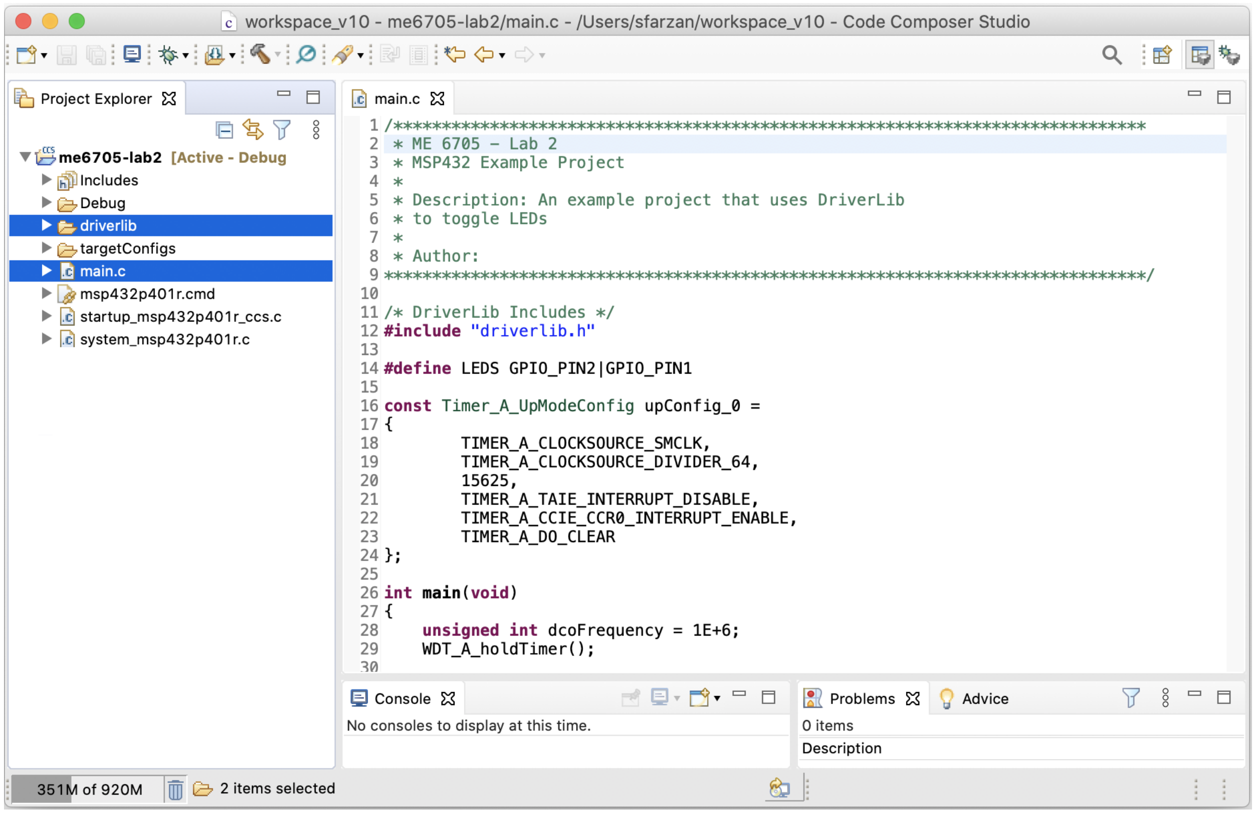
Open Import Wizard to find local example projects for selected device...

Figure 5: 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.

2.4.3 Including the DriverLib library in the project

- Download the `main.c` file and `driverlib` folder provided with the lab assignment (available on Canvas) and unzip them.
- On your computer, find the directory (folder) where the project is created (under the workspace folder). Copy and paste the `main.c` file and the `driverlib` folder into the Lab 2 project folder under the workspace. The original `main.c` source file will be replaced with the new one (Figure 6).



```
/******************************************************************************
 * ME 6705 - Lab 2
 * MSP432 Example Project
 *
 * Description: An example project that uses DriverLib
 * to toggle LEDs
 *
 * Author:
 ******************************************************************************/

/* DriverLib Includes */
#include "driverlib.h"

#define LEDS GPIO_PIN2|GPIO_PIN1

const Timer_A_UpModeConfig upConfig_0 =
{
        TIMER_A_CLOCKSOURCE_SMCLK,
        TIMER_A_CLOCKSOURCE_DIVIDER_64,
        15625,
        TIMER_A_TAIE_INTERRUPT_DISABLE,
        TIMER_A_CCIE_CCR0_INTERRUPT_ENABLE,
        TIMER_A_DO_CLEAR
};

int main(void)
{
    unsigned int dcoFrequency = 1E+6;
    WDT_A_holdTimer();
```

No consoles to display at this time.

0 items
Description

Figure 6: `main.c`and `driverlib` copied into the project.

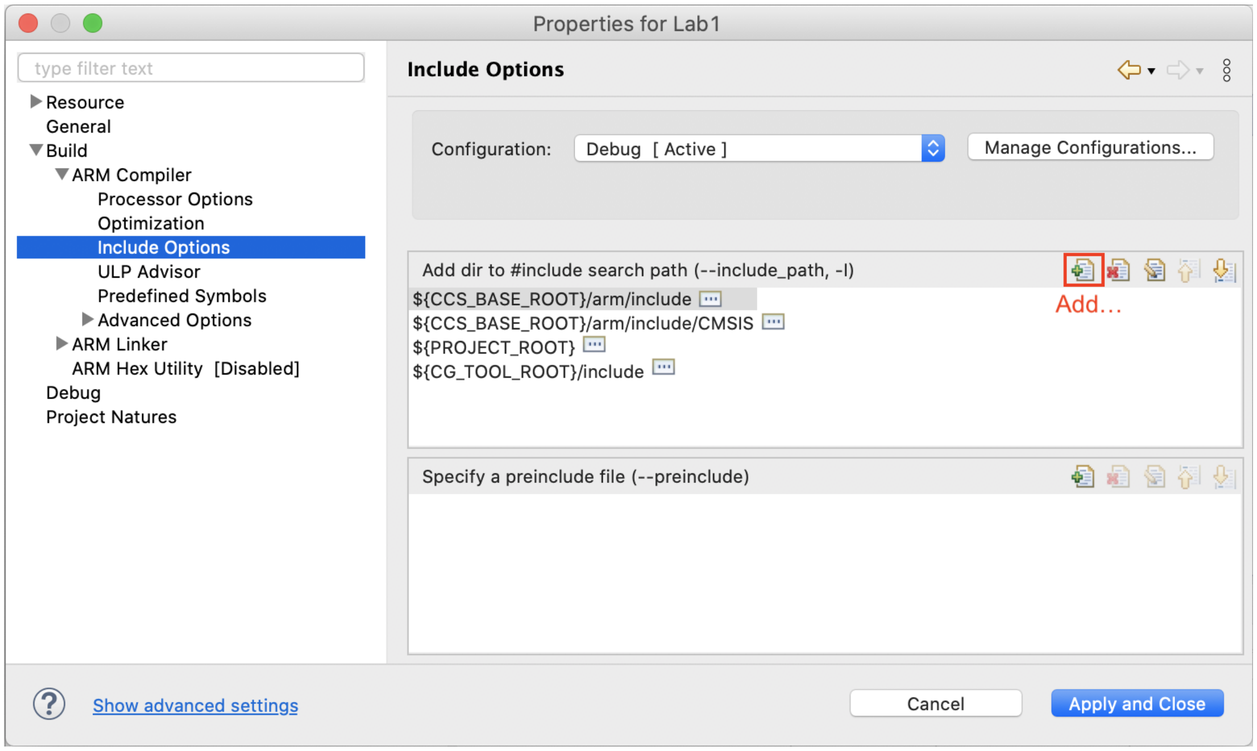
- Make sure that the following line is added to the beginning of the `main.c` code:

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

2.4.4 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 7), select `Build` → `ARM Compiler` → `Include Options`.
- On the right side of the window (shown in Figure 7), click the add icon, select `Workspace...`, and select the folder `Lab2/driverlib/MSP432P4xx` (Figure 8).



Configuration: Debug [Active]

Add dir to #include search path (--include_path, -I)

${CCS_BASE_ROOT}/arm/include
${CCS_BASE_ROOT}/arm/include/CMSIS
${PROJECT_ROOT}
${CG_TOOL_ROOT}/include

Specify a preinclude file (--preinclude)

Figure 7: Adding the "driverlib" folder to `#include` search path, using the `Include Options` build properties.

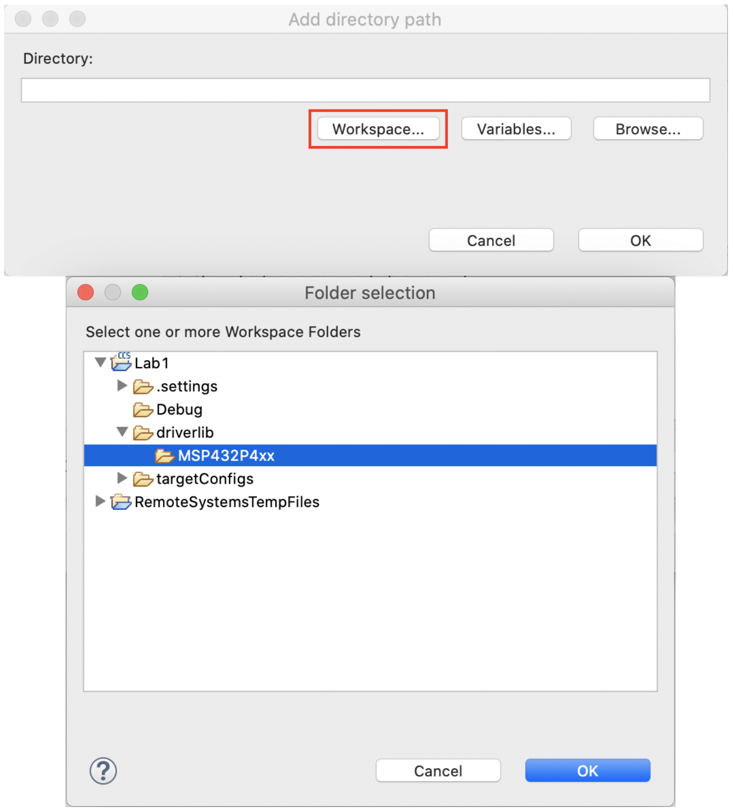
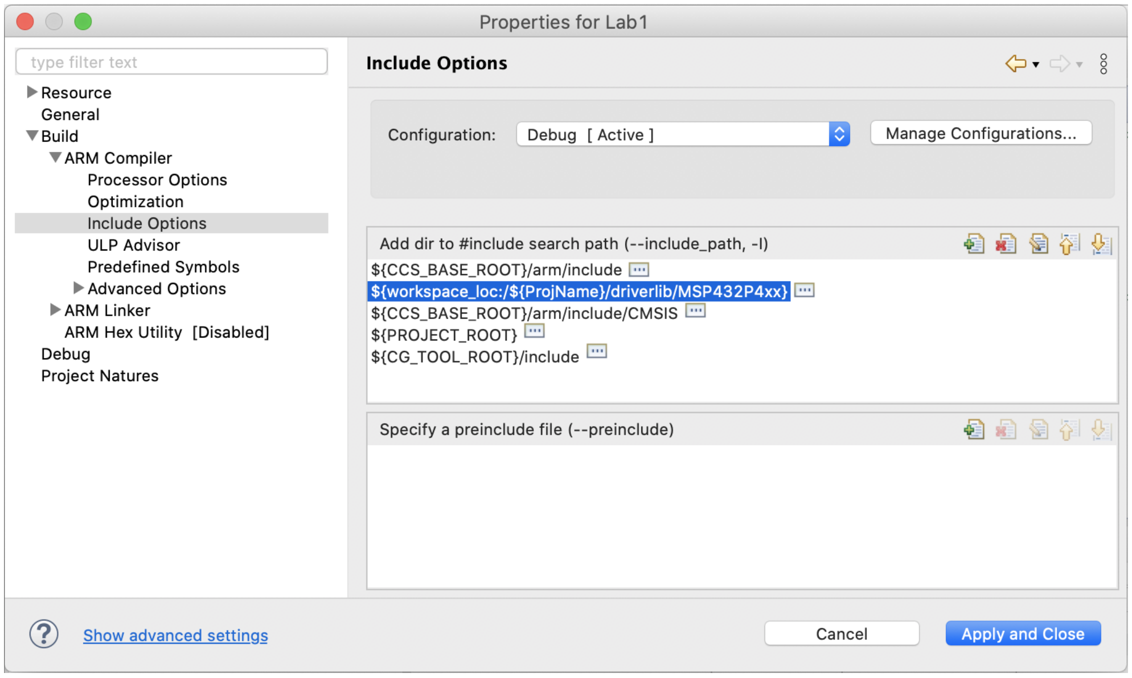


Figure 8: Select the `Lab1/driverlib/MSP432P4xx` folder to be included in the search path.

- 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 9).



```
${CCS_BASE_ROOT}/arm/include
${workspace_loc:/${ProjName}/driverlib/MSP432P4xx}
${CCS_BASE_ROOT}/arm/include/CMSIS
${PROJECT_ROOT}
${CG_TOOL_ROOT}/include
```

Figure 9: 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`

2.4.5 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 10). The project should build without errors.

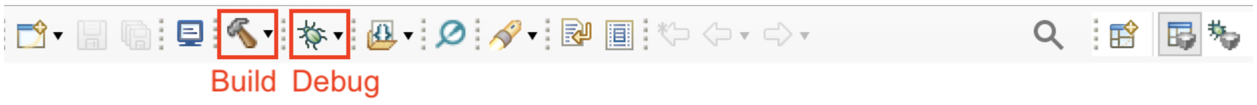
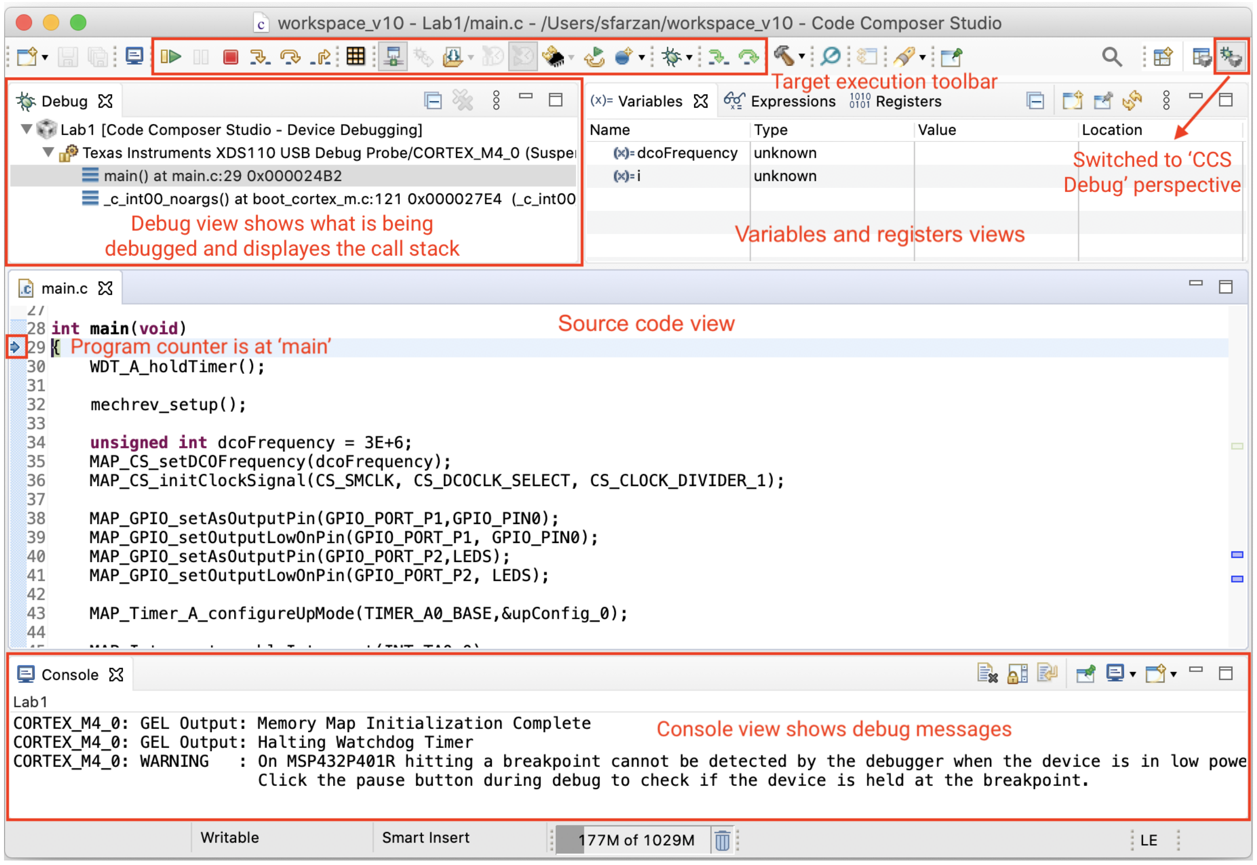


Figure 10: Build and Debug icons on the Quick Access panel.

- 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 10). 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 11). Additional buttons are now available in the Quick Access panel as the Target Execution toolbar.


workspace_v10 - Lab1/main.c - /Users/sfarzan/workspace_v10 - Code Composer Studio
Target execution toolbar
Debug
Switched to 'CCS Debug' perspective
Debug view shows what is being debugged and displays the call stack
Variables
Expressions
Registers
Variables and registers views
main.c
Source code view
Program counter is at 'main'
Console
Console view shows debug messages
Writable
Smart Insert
177M of 1029M
LE

```
Lab1 [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Suspe
    main() at main.c:29 0x000024B2
    _c_int00_noargs() at boot_cortex_m.c:121 0x000027E4  (_c_int00
```

Name	Type	Value	Location
dcoFrequency	unknown		
i	unknown		

```c
int main(void)
{
    WDT_A_holdTimer();

    mechrev_setup();

    unsigned int dcoFrequency = 3E+6;
    MAP_CS_setDCOFrequency(dcoFrequency);
    MAP_CS_initClockSignal(CS_SMCLK, CS_DCOCLK_SELECT, CS_CLOCK_DIVIDER_1);

    MAP_GPIO_setAsOutputPin(GPIO_PORT_P1,GPIO_PIN0);
    MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);
    MAP_GPIO_setAsOutputPin(GPIO_PORT_P2,LEDS);
    MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, LEDS);

    MAP_Timer_A_configureUpMode(TIMER_A0_BASE,&upConfig_0);
```

```
Lab1
CORTEX_M4_0: GEL Output: Memory Map Initialization Complete
CORTEX_M4_0: GEL Output: Halting Watchdog Timer
CORTEX_M4_0: WARNING   : On MSP432P401R hitting a breakpoint cannot be detected by the debugger when the device is in low powe
                         Click the pause button during debug to check if the device is held at the breakpoint.
```

Figure 11: 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.

2.4.6 Debugging and pausing the program

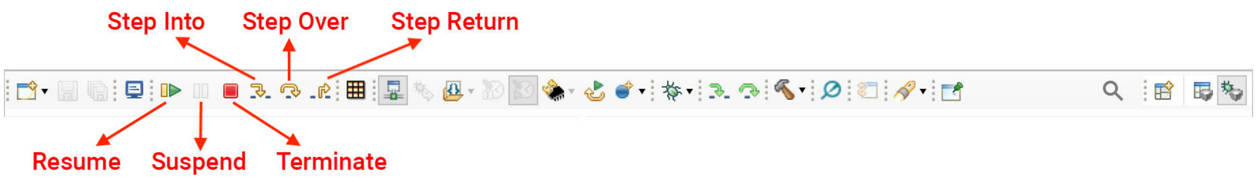


Figure 12: Target Execution toolbar.

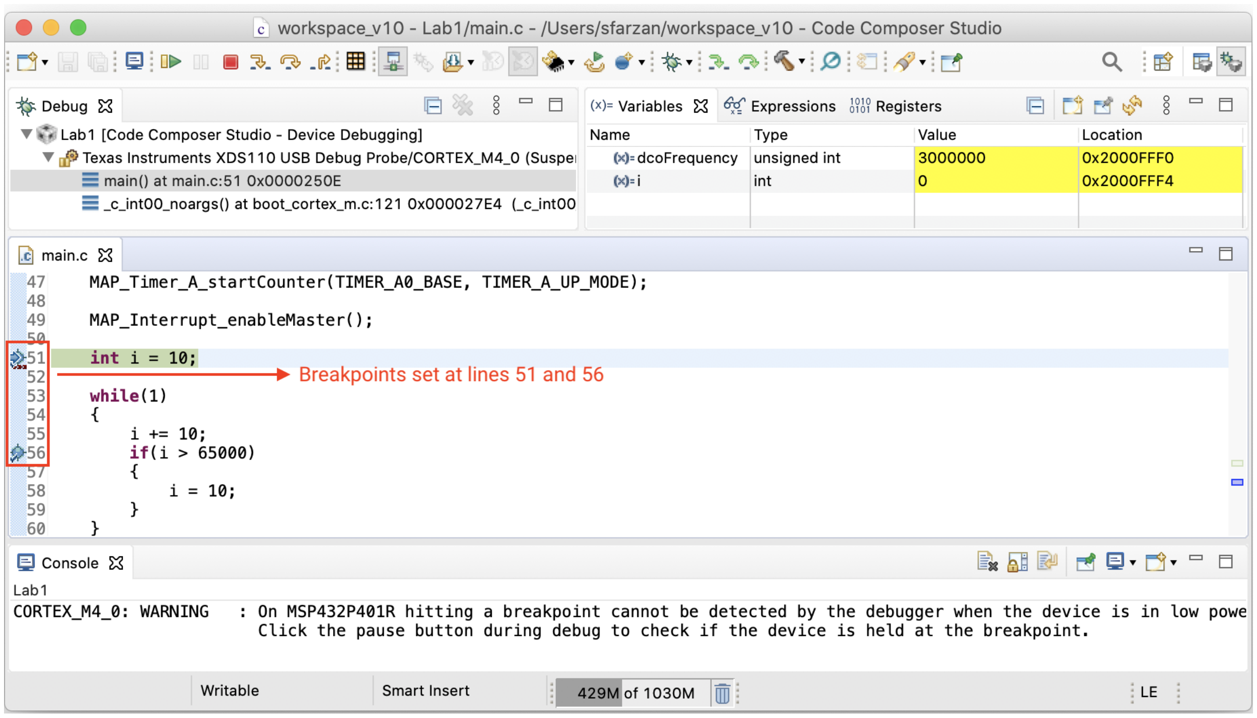
- While in the debug mode, press the Resume button on the Target Execution toolbar (Figure 12), Both the LEDs on the board will start blinking. The Suspend button will suspend the execution of the code at current step.

- 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 12):
  - *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()).

2.4.7 Setting breakpoints, checking variables and registers

- To set up the debug session to answer the Problems for Lab Assignment 2, 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 13). 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 13.
- Click on the Variables tab in the watch window as shown in Figure 14. Two variables should be visible: `dcoFrequency` and `i`.
- The latest register values can be checked in the registers window embedded in the screen (Figure 15). 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.


workspace_v10 - Lab1/main.c - /Users/sfarzan/workspace_v10 - Code Composer Studio
Debug
Variables
Expressions
Registers
main.c
Console
Breakpoints set at lines 51 and 56
Writable
Smart Insert
429M of 1030M
LE

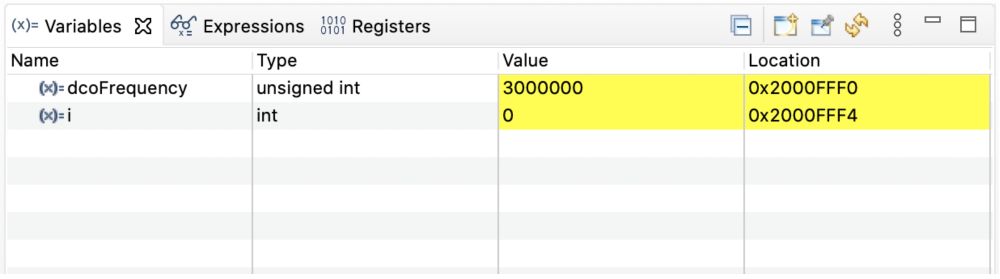
```
Lab1 [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe/CORTEX_M4_0 (Suspe
    main() at main.c:51 0x0000250E
    _c_int00_noargs() at boot_cortex_m.c:121 0x000027E4  (_c_int00
```

Name	Type	Value	Location
dcoFrequency	unsigned int	3000000	0x2000FFF0
i	int	0	0x2000FFF4

```
47      MAP_Timer_A_startCounter(TIMER_A0_BASE, TIMER_A_UP_MODE);
48
49      MAP_Interrupt_enableMaster();
50
51      int i = 10;
52
53      while(1)
54      {
55          i += 10;
56          if(i > 65000)
57          {
58              i = 10;
59          }
60      }
```

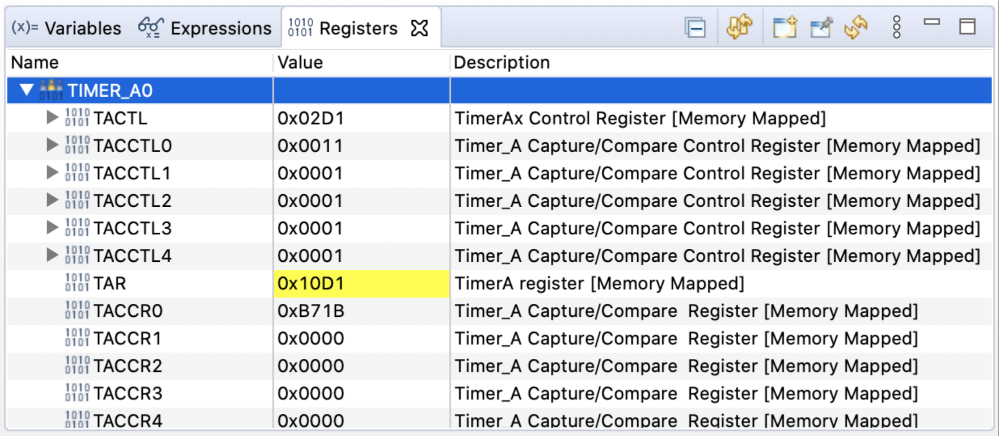
```
Lab1
CORTEX_M4_0: WARNING   : On MSP432P401R hitting a breakpoint cannot be detected by the debugger when the device is in low powe
                         Click the pause button during debug to check if the device is held at the breakpoint.
```

Figure 13: Setting breakpoints in debug mode.



Name	Type	Value	Location
dcoFrequency	unsigned int	3000000	0x2000FFF0
i	int	0	0x2000FFF4

Figure 14: Variables window in debug mode.



Name	Value	Description
TIMER_A0		
TACTL	0x02D1	TimerAx Control Register [Memory Mapped]
TACCTL0	0x0011	Timer_A Capture/Compare Control Register [Memory Mapped]
TACCTL1	0x0001	Timer_A Capture/Compare Control Register [Memory Mapped]
TACCTL2	0x0001	Timer_A Capture/Compare Control Register [Memory Mapped]
TACCTL3	0x0001	Timer_A Capture/Compare Control Register [Memory Mapped]
TACCTL4	0x0001	Timer_A Capture/Compare Control Register [Memory Mapped]
TAR	0x10D1	TimerA register [Memory Mapped]
TACCR0	0xB71B	Timer_A Capture/Compare Register [Memory Mapped]
TACCR1	0x0000	Timer_A Capture/Compare Register [Memory Mapped]
TACCR2	0x0000	Timer_A Capture/Compare Register [Memory Mapped]
TACCR3	0x0000	Timer_A Capture/Compare Register [Memory Mapped]
TACCR4	0x0000	Timer_A Capture/Compare Register [Memory Mapped]

Figure 15: Registers window in debug mode.

2.5 Questions

Debugging Questions:

1. In the variables window, right click on the Value property of variable `dcoFrequency`. Change the number format to hexadecimal. What is the value shown? Confirm that this value is equal to the decimal value displayed (show your work).

2. Press `F8` (or the Resume button) and proceed to the breakpoint at line 51. Now press `F8` four more times, monitoring the value of variable `i` in the Variables window. What is the value of `i` after pressing `F8` four more times (repeating the loop)? After changing the displayed format to binary, how many bits are shown? Why is this number of bits displayed (recall data types)?

Circuit Design Questions:

3. Design a circuit that uses a solid state (semiconductor) device to switch a DC motor ON or OFF. The circuit should be able to handle high current spikes. The voltage source available is a 30VDC power supply. The motor specifications are given in the "DC motor.pdf" file (use the second motor requiring rated 24VDC supply). Notice the starting current of the motor is a high value. (Hint: Consider using a transistor-based switching circuit such as that shown in Slide 38 of Lecture 2, with appropriate modifications as desired.)

4. Find a solid state (semiconductor) switching device on the internet that can be used in the above circuit. Attach the datasheet for this device. Point out to the important details in the datasheet of the device that proves the usability of the device.