

## ME/AE 6705 Lab 2

### Getting started with Code Composer Studio

#### Objective

The objective of this lab is to get acquainted Code Composer Studio (CCS) which will be used throughout the rest of the course. This lab covers installing CCS, running a first program, and debugging the program along with the MSPM0 device.

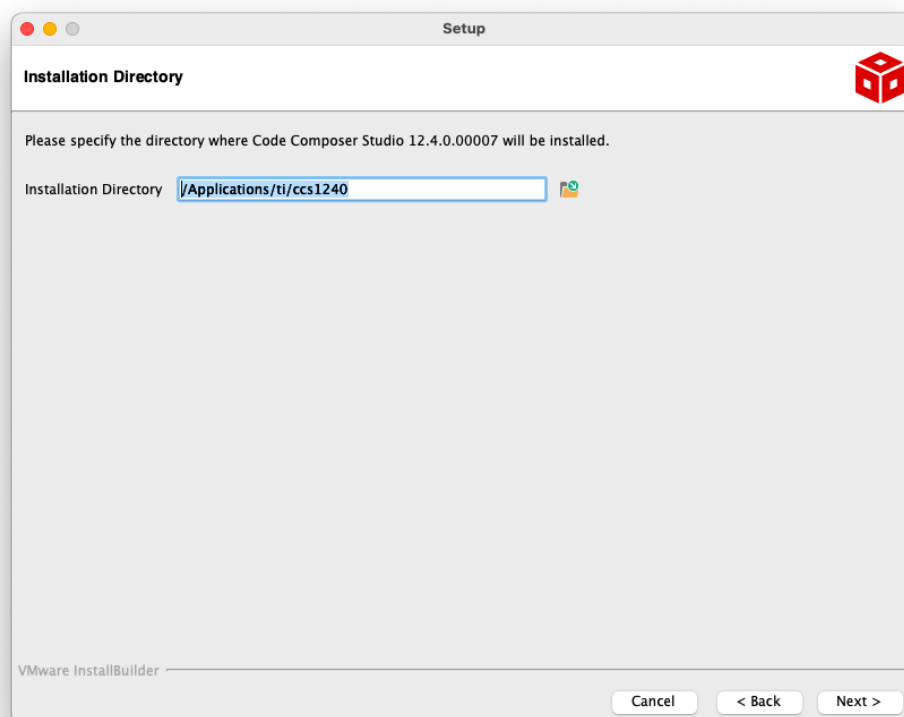
#### Deliverables and Grading

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 above due date. (60 points)

#### Procedure

##### Setup CCS:

1. On your personal computer, download TI Code Composer Studio (CCS) v12.4.0 from the TI website at: <https://www.ti.com/tool/download/CCSTUDIO>. IMPORTANT: Do NOT download the latest version. Instead, v12.4.0 (the 10 July 2023 release) by going to the link above, going to the Select a version pane, and clicking 12.4.0. Also, do not download CCSTUDIO-THEIA, only download CCSTUDIO. If you use the link above it will take you to the proper download site. More information can be found at [http://processors.wiki.ti.com/index.php/Download\\_CCS](http://processors.wiki.ti.com/index.php/Download_CCS).  
Note: You can also reach the correct version by clicking here: <https://www.ti.com/tool/download/CCSTUDIO/12.4.0>
2. Select the appropriate version (Windows/Linux/macOS).
3. Initiate the installation either by downloading the web installer or the complete file and double clicking to install it.
4. Read and accept all the terms and conditions in the license agreement.
5. License Agreement: Accept the terms of the license agreement. After clicking next, if any additional installation items (e.g., Rosetta) need to be installed, please do so.
6. Choose Installation Location: Choose where to install CCS by browsing to the desired path. The default location may be used.



7. Setup type: Select Custom Installation and click next.
8. Select Components: Select only “MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs” product family to be installed and click *Next*.
9. Install Debug Probes: Select only “SEGGER J-Link” and click *Next*.
10. Unsupported Boards: Click *Next*.
11. Ready to Install: Click *Next*.

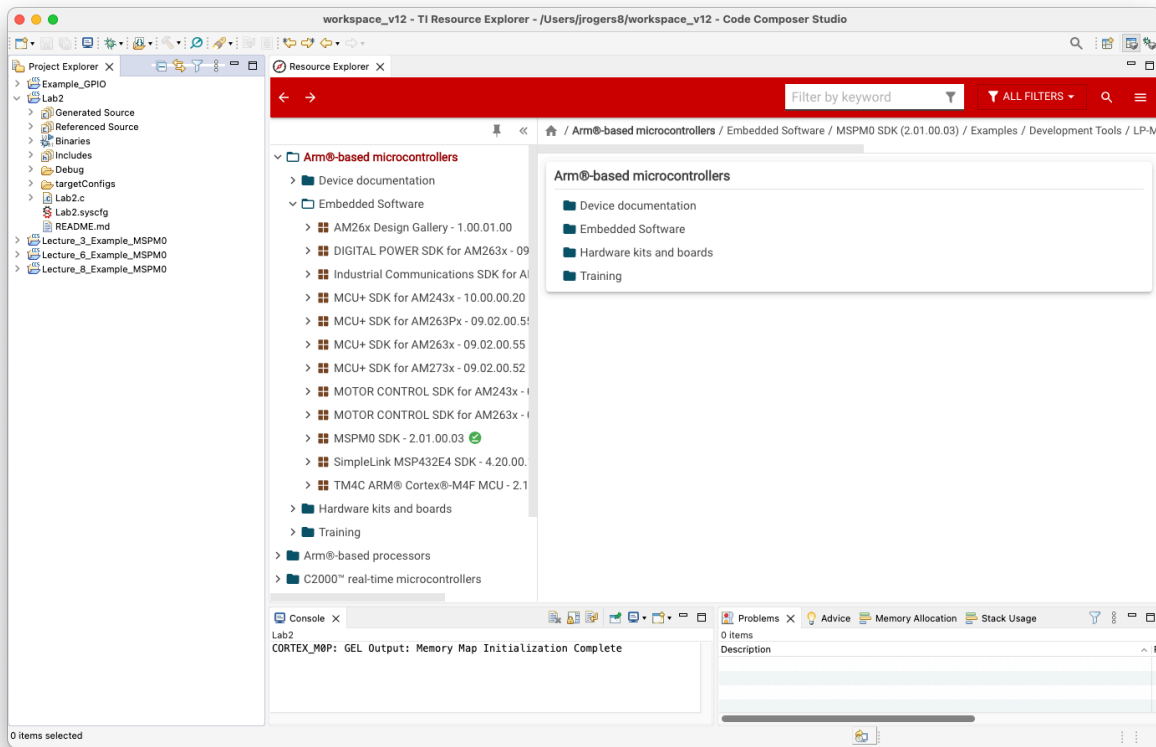
### Starting the Application:

12. Launch Code Composer Studio by double clicking on the Code Composer Studio icon.
13. A new window will pop up. This will ask 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.
14. An empty CCS workbench will open with a Getting Started Window.

### MSPM0 SDK Installation

15. Navigate to Resource Explorer by clicking View then Resource Explorer.
16. In Resource Explorer, Click on Arm-based microcontrollers, then Embedded Software. In the options that come up, you will see MSPM0 SDK – 2.01.00.03. With the cursor over this option, you will see three dots come up to the right. Click on those dots, then choose Download and Install.

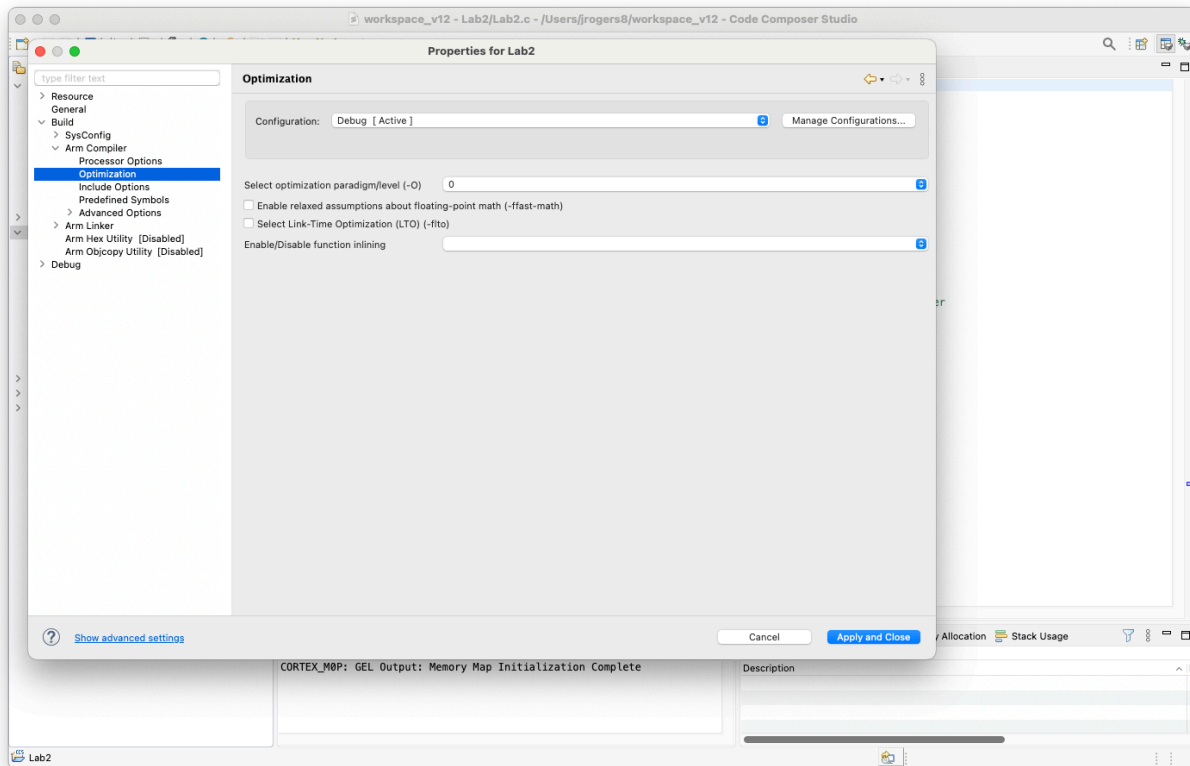
The installation should occur in the background. When it is done, there will be a green checkmark next to this entry as shown below.



## Open a New Project and Configure it

17. Click on the MSPM0 SDK option, then click Examples, then Devices, then MSPM0G1X0X-G3X0X. Then click MSPM0G3507 -> DriverLib -> empty\_mspm0g3507 -> No RTOS -> TI Clang Compiler -> empty\_mspm0g3507. There should now be a small Import button on the top right of the window. Click this button. This will import the project to your workspace.
18. Right click on the name of the project (empty\_mspm0g3507\_nortos\_ticlang) in the Project Explorer window and select Rename. Rename the project to Lab2. Make sure to keep the “update references” box checked.
19. Expand the project using the arrow at the left. Right click on empty\_mspm0g3507.syscfg and select rename. Rename it to Lab2.syscfg.
20. Open the empty\_mspm0g3507.c file by double clicking on it. Delete all the content so it is empty. Open the example Lab2.c file provided on Canvas in a text editor or by opening it in CCS using the File -> Open File command. Copy all the code from the Lab2.c file provided in Canvas into empty\_mspm0g3507.c.
21. Close the Lab2.c file provided in class. Rename the empty\_mspm0g3507.c file to Lab2.c by right clicking on it and selecting Rename.

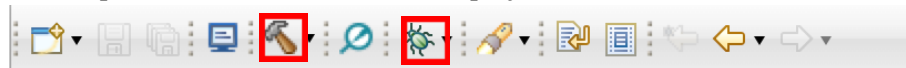
22. Right click on the project name (should now be called Lab2) and select Properties, all the way at the bottom. Click on Build -> Arm Compiler -> Optimization. In Select optimization paradigm/level, select 0 (the default is 2). Then Apply and Close. See screenshot below.



23. Now, connect the board to the computer with the USB cable.

### Building the project and Flashing the board:

24. Build the project (*Project* → *Build Project*), or click on the Build button, shown by a *Hammer* on the Quick Access panel below the menu bar. The project should build without errors.



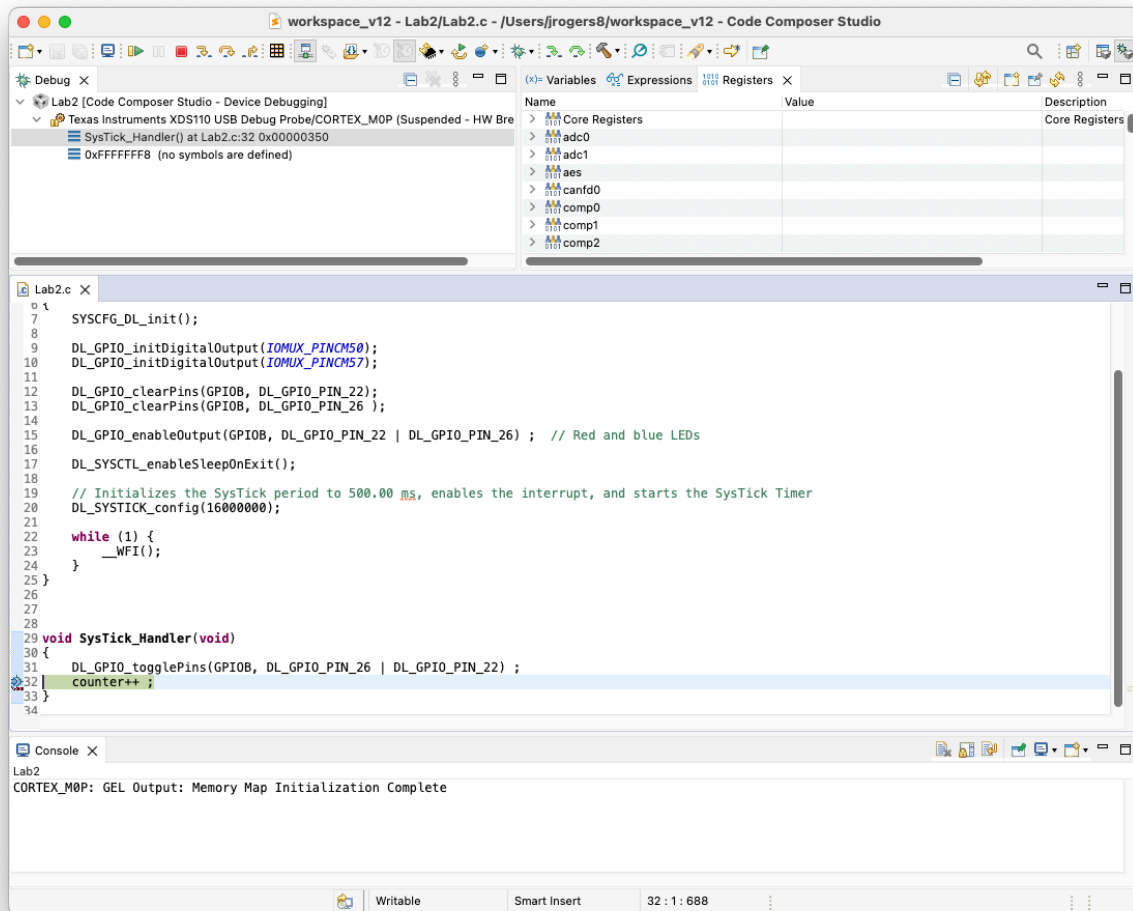
25. Debug the application (*Run* → *Debug (F11)*), or click on the Debug button, shown by a *green bug* on the Quick Access panel below the menu bar.  
This starts the debugger, which gains control of the target, erases the target memory, programs the target memory with the application, and resets the target. For a new board, you may be asked if you want to update some firmware, you can safely click yes. This should only happen once (or not at all).
26. The project is now flashed onto the chip. This code will now execute each time the controller is powered unless it is flashed with a new code in a similar manner.
27. The perspective of the CCS will change. The CCS is now in Debug mode. Additional buttons are now available in the quick access panel.



(Note: Click *Run* → *Resume* (F8) to start the application.

Click *Run* → *Terminate* to stop the application and to exit the debugger. CCS returns to the C/C++.)

28. The code is stopped at the first line of the main function. It will continue execution if the resume button is pressed.
29. When paused, the line to be executed next is pointed to by the arrow on the left side of the line numbers. It is also highlighted in green as shown in the following figure.



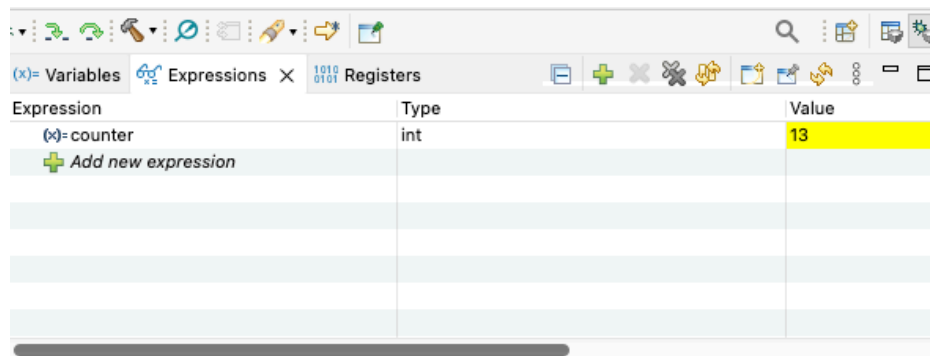
### Pausing and debugging the program:

30. Press the resume button, which the green arrow button below.



A purple LED on the board (actually a blue and red LED next to each other) will start blinking. Pause button will suspend the execution of the code at current step.

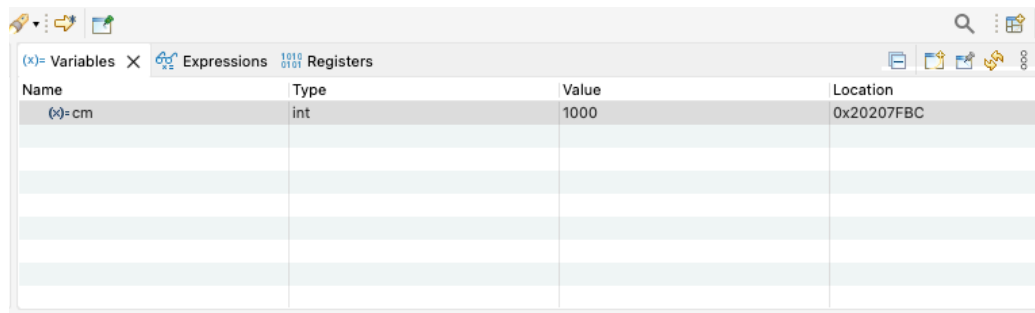
31. The current register values can be checked in the registers window embedded in the screen. To understand how this works, hit Pause, then scroll in the Registers tab to the GPIOB\_DOUT31\_0 register. This 32-bit register holds the output pin values for GPIOB pins – if any pin is high, its corresponding bit in this register is 1. If the code is paused when the lights are on, this register should show 0x04400000, meaning that bits 22 and 26 are high. If the code is paused when the lights are off, the register should show 0x00000000, meaning the pins are low. Test this out and verify you are seeing this behavior.
32. Step by step execution causes the registers whose values have changed to be highlighted in yellow.
33. This window can also be used to check the values of variables and place breakpoints during debugging. For instance, pause the program using the pause button. Then, under the Expressions tab, click Add New Expression and type counter. Then unpause the program, wait a few blink cycles, and hit pause again. The variable counter can be seen to change its value every time the light blinks on or off.



34. When paused, code can be executed step by step using the step button. The Step Into button (yellow arrow pointing downwards) is the button next to the red square (terminate button). This button executes one action at a time. To execute a function directly and proceed to the next step after the function completes, press the Step Over button next to the Step Into button.



35. The Terminate button (red square) terminates the debug session. The code will continue to run on the MCU, but you will not be able to pause, hit breakpoints, etc.
36. To set up the debug session to answer the questions below, terminate the debugging session (if you are running one). Set a breakpoint at line 36. This can be done by double clicking next to the line number in the blue bar.
37. Now press the debug button. Once the code is paused in the main function, press F8 (or the Resume button) to progress to line 36. The program should pause execution at line 36.
38. Click on the Variables tab in the watch window as shown below. Only local variables scoped to this function will appear – this is the variable cm.



The screenshot shows the 'Variables' window in an IDE. It contains a table with the following data:

Name	Type	Value	Location
cm	int	1000	0x20207FBC

## Questions

### Debugging Questions:

1. In the variables window, right click on the Value property of `cm`. 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. Now press F8 four more times (or click on the Resume button four more times), letting the program stop at line 36 before each time you press it. Monitor the value of `counter` in the Expressions window. What is the value of `counter` after you have pressed F8 four times? 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.