

ME4405 - Fundamentals of Mechatronics (Fall 2019)

Lab Assignment Two Getting Started with CCS and Reading Spec Sheets

Due Thursday, September 12th, 2019

Objective: The objectives of this lab are to (1) **get acquainted Code Composer Studio (CCS)** which will be used throughout the rest of the course and (2) **practice reading and interpreting component specification sheets**. This lab covers installing CCS, running a first program, debugging the program along with the MSP432 device, and building a theoretical mechatronic system by selecting components that satisfy the system operating requirements and constraints.

Deliverables and Grading:

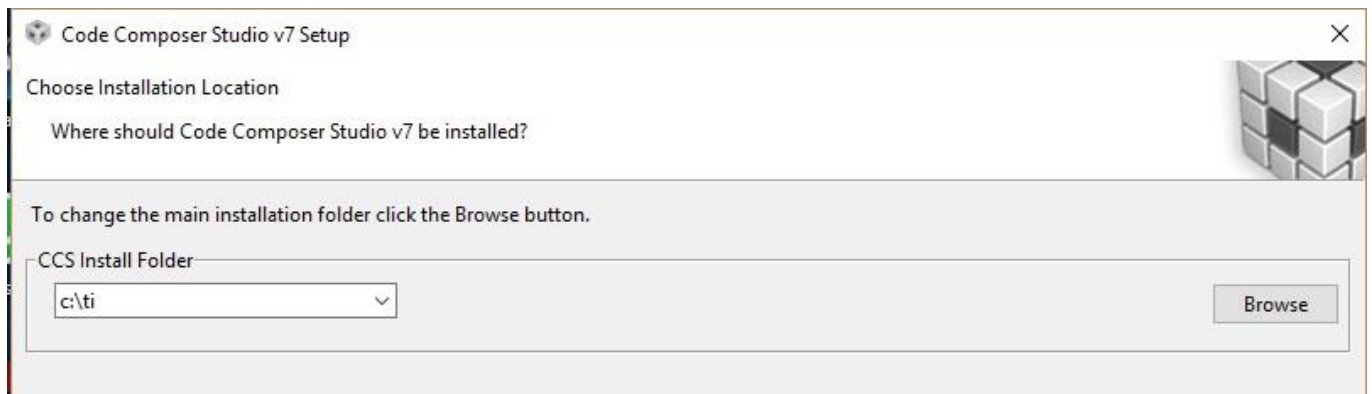
To get credit for this lab assignment you must:

- 1) Turn in a copy of your typed report answering the questions at the end of the lab manual. This report is due **by 5pm** on Canvas on the above due date. **(60 points)**

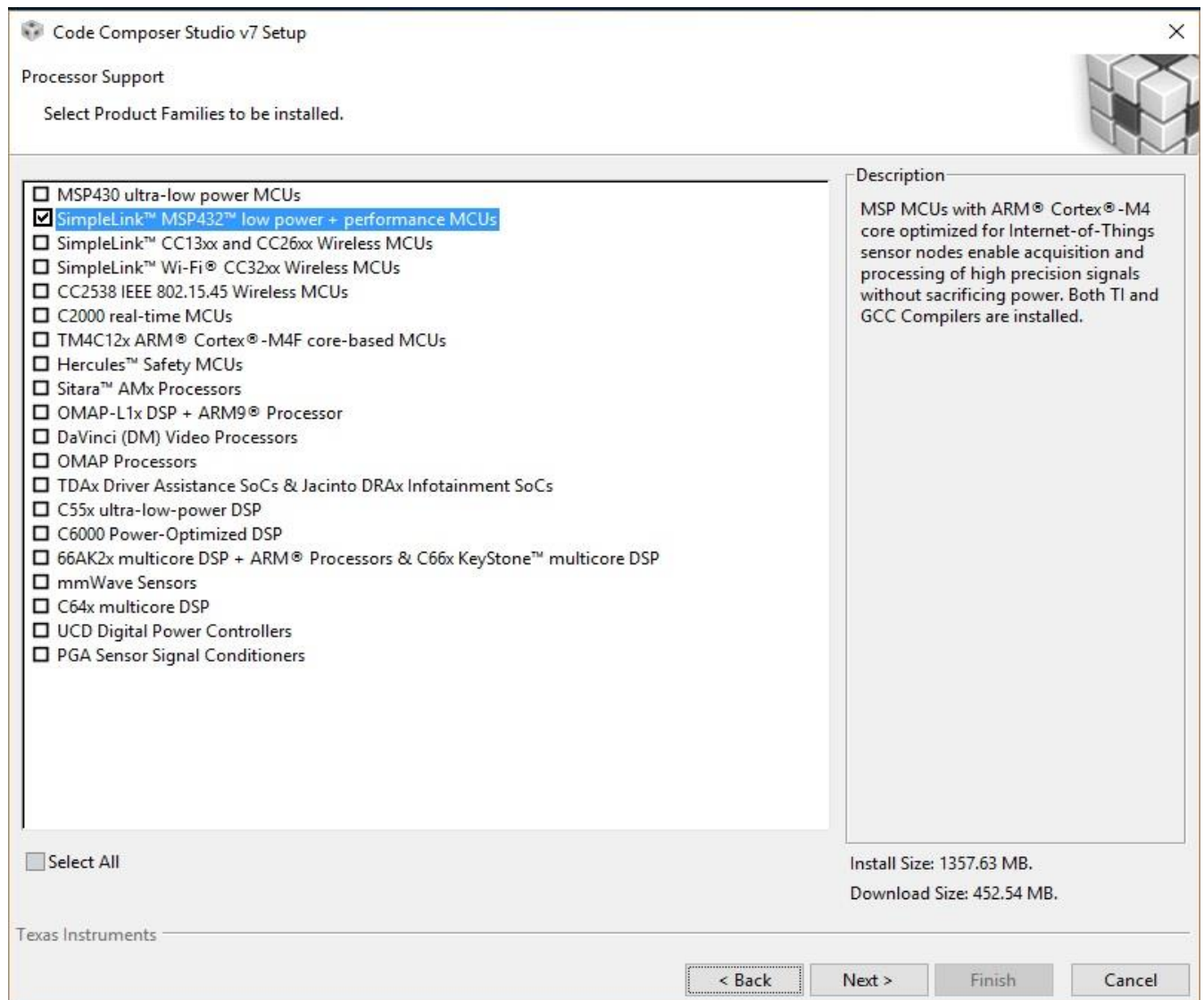
Procedure:

Installation and setting up of the Code Composer Studio (CCS) Integrated Development Environment (IDE):

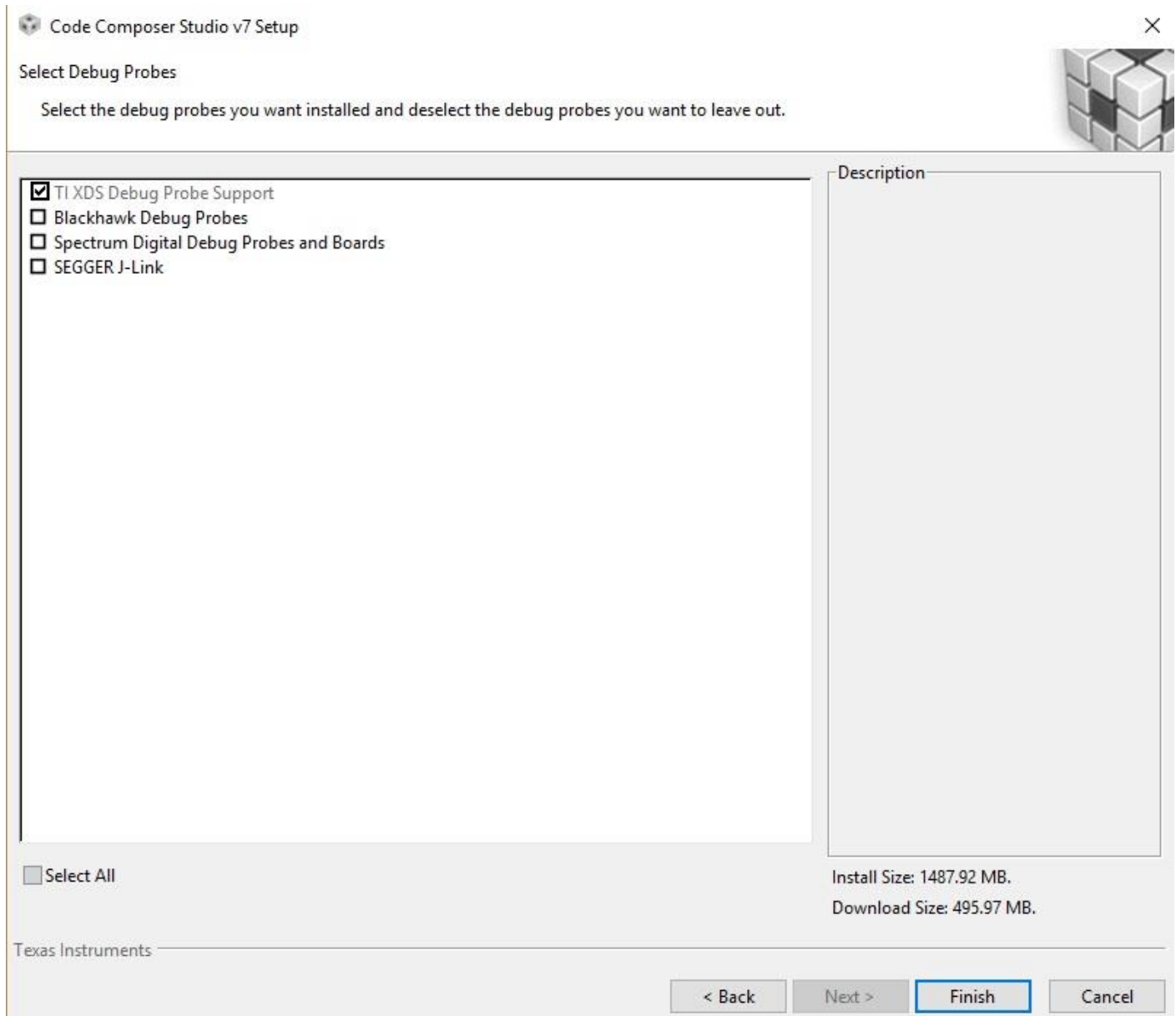
1. Download the latest version of **Texas Instruments CCS IDE version** from this link http://processors.wiki.ti.com/index.php/Downloading_CCS.
2. Extract the folder to an appropriate location. **It is best to disable any real-time anti-virus protection software before beginning the installation.**
3. Double click the “ccs_setup....exe” file to start installing CCS. Read and accept all the terms and conditions.
4. Choose where to install CCS by browsing to the desired path as shown in the image below.



5. Select the option: **SimpleLink MSP432 low power + performance MCUs**



6. Install (only) the default probes.



11. Click Finish to begin installation. This will take a while as it downloads and installs all the necessary components.

Download MSP432 driverlib

1. Download the latest version of the MSP432 driver library from this link: http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP432_Driver_Library/latest/index_FDS.html

As shown below:

Yes! MSP432Ware installations can exist alongside each other. You should only see the latest installed version within the TI Resource Explorer window of Code Composer Studio however.

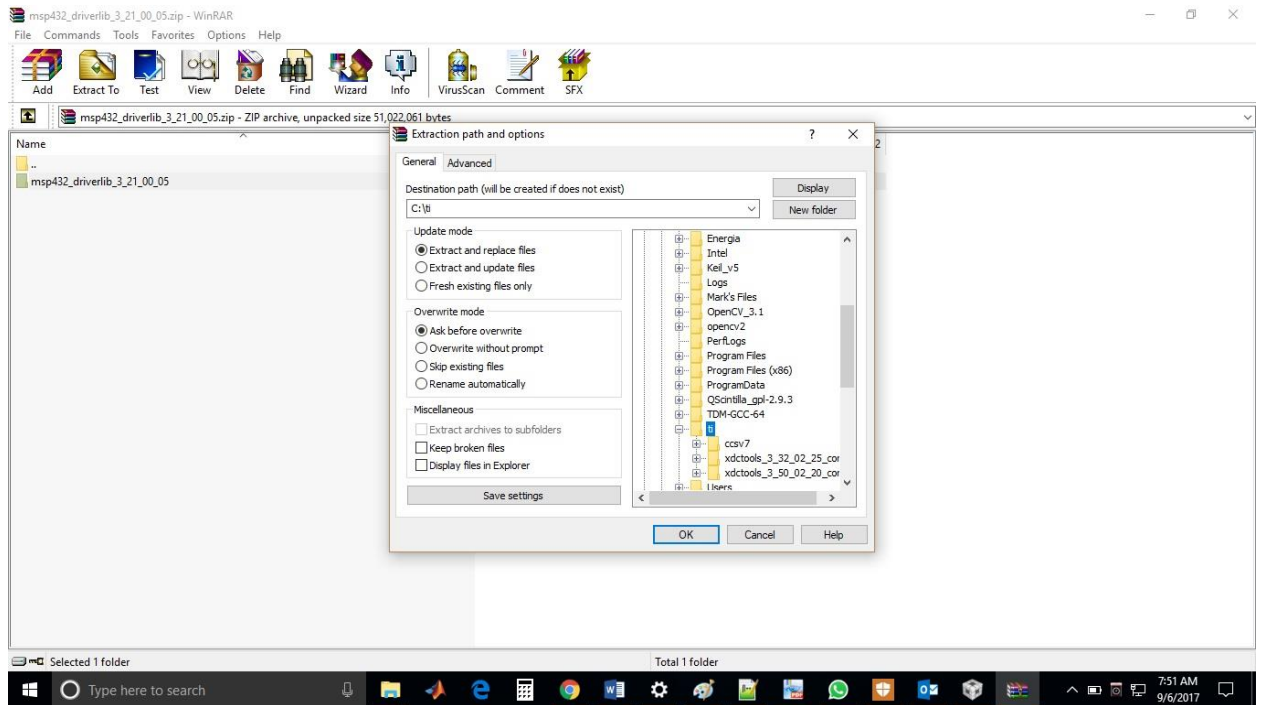
How can I get support for MSP432Ware?

Feel free to use the [TI E2E forum](#) to ask anything MSP432Ware related.

MSP432_Driver_Library Product downloads

Title	Description	Size
msp432_driverlib_3_21_00_05.zip	BSD Licensed Driver Library	10392K
Code Composer Studio	Recommended CCS Download	
MSP432Ware	Recommended MSP432Ware Download	
MSP430 Driver Library	BSD Licensed Driver Library for MSP430	
MSP432 Driver Library Documentation		
MSP432 Driver Library Release Notes		
MSP432 Driver Library MSP432P4xx		
MSP432 Driver Library MSP432P4xx API Guide		

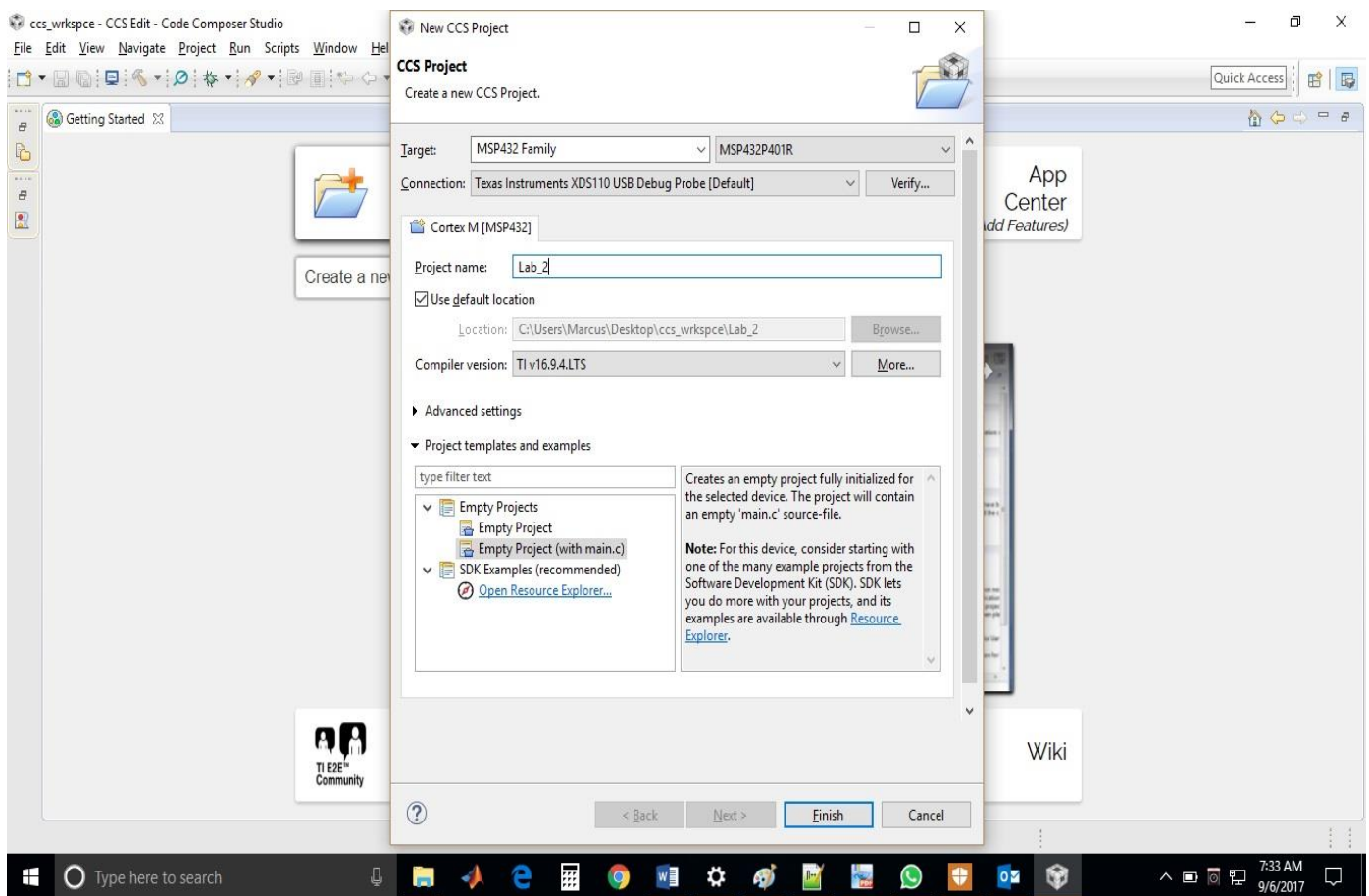
- Download and extract the zip file to the same directory (For example C:\ti\) as the CCS installation directory.



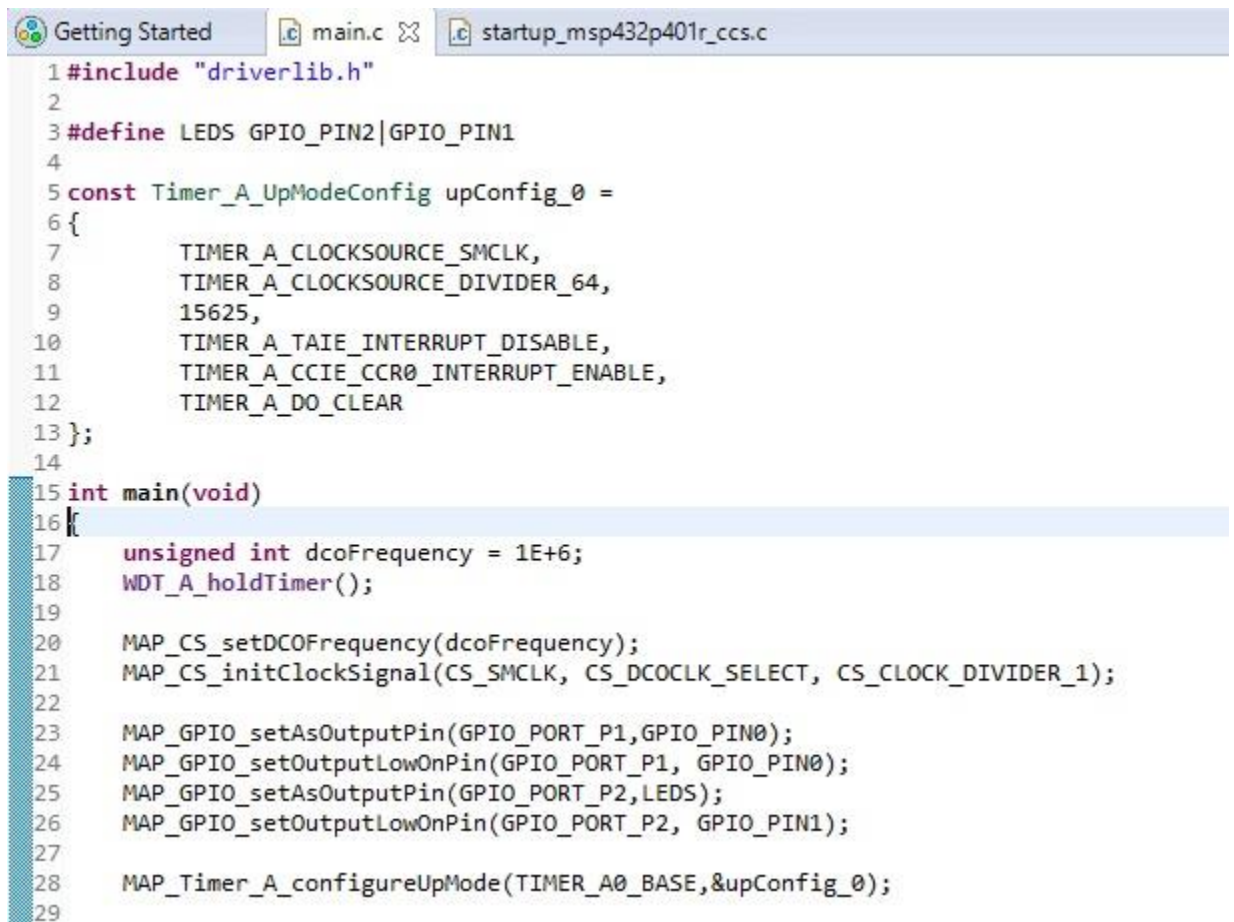
Starting the Application:

- Launch Code Composer Studio software by double clicking on the Code Composer Studio icon on the desktop.

2. 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.
3. When launched for the first time, CCS will prompt you to run a few updates. Go ahead and select all the available updates, read and agree to the terms & conditions and install the updates. Following this, CCS will restart.
4. An empty CCS workbench will open with a Getting Started Window.
5. Click on the New Project button. This should open a window with the title “New CCS Project”
6. Give Project name as Lab_2. In the Target dialogue box, put MSP432P401R.



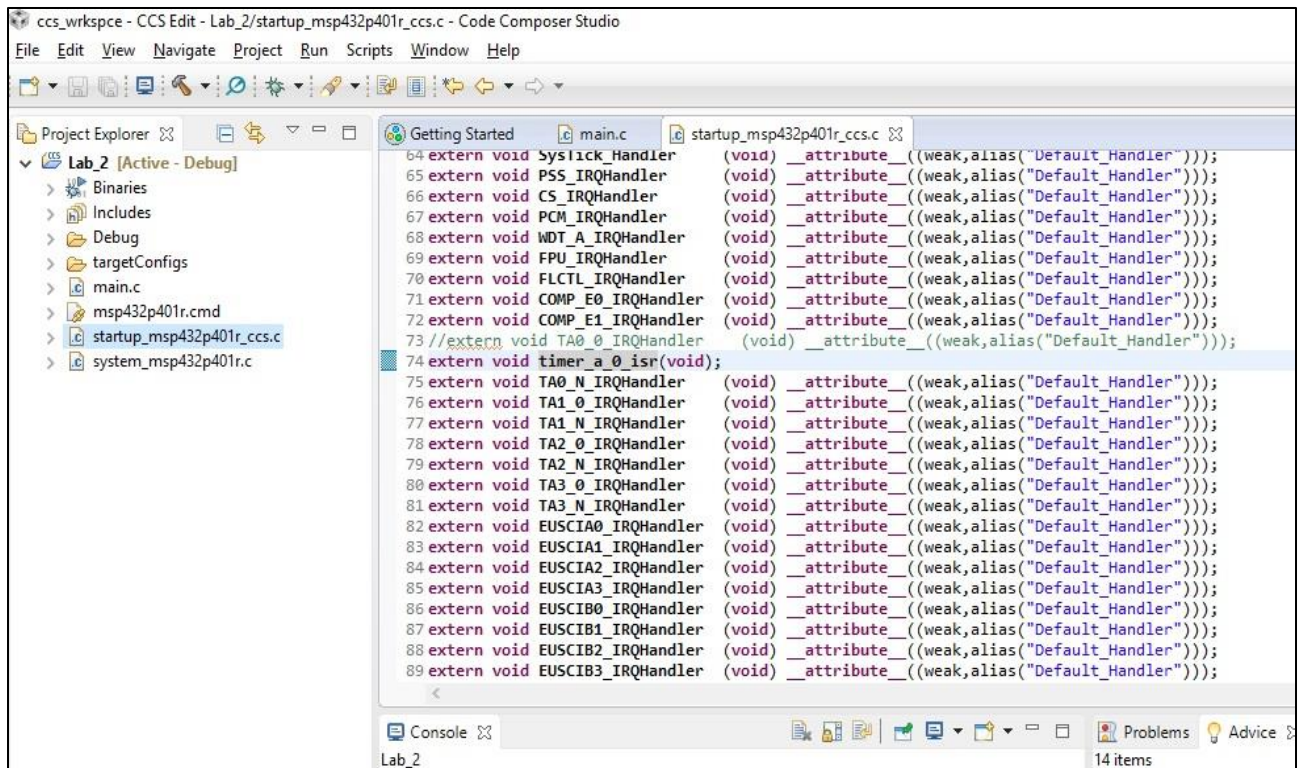
7. In the adjacent drop down menu select MSP432P401R.
8. In the Project templates, select **Empty Project (with main.c)**. Click Finish.
9. A main.c file opens with a part of code already written in the editor.
10. Delete the contents created in the main.c by default and replace it with the main.c provided with this lab manual on Canvas.



```
1#include "driverlib.h"
2
3#define LEDS GPIO_PIN2|GPIO_PIN1
4
5const Timer_A_UpModeConfig upConfig_0 =
6{
7    TIMER_A_CLOCKSOURCE_SMCLK,
8    TIMER_A_CLOCKSOURCE_DIVIDER_64,
9    15625,
10    TIMER_A_TAIE_INTERRUPT_DISABLE,
11    TIMER_A_CCIE_CCR0_INTERRUPT_ENABLE,
12    TIMER_A_DO_CLEAR
13};
14
15int main(void)
16{
17    unsigned int dcoFrequency = 1E+6;
18    WDT_A_holdTimer();
19
20    MAP_CS_setDCOFrequency(dcoFrequency);
21    MAP_CS_initClockSignal(CS_SMCLK, CS_DCOCLK_SELECT, CS_CLOCK_DIVIDER_1);
22
23    MAP_GPIO_setAsOutputPin(GPIO_PORT_P1,GPIO_PIN0);
24    MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);
25    MAP_GPIO_setAsOutputPin(GPIO_PORT_P2,LEDS);
26    MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN1);
27
28    MAP_Timer_A_configureUpMode(TIMER_A0_BASE,&upConfig_0);
29
```

11. Now, open the **startup_msp432p401r_ccs.c** file in the Project Explorer by double clicking on it and scroll down to line 73. Comment out the existing command by prepending the line with a **double-backslash (//)** as shown in the image below. On the next line, type the following : **extern void timer_a_0_isr(void);** as shown in the image below.

DO NOT FORGET THE SEMI-COLON !



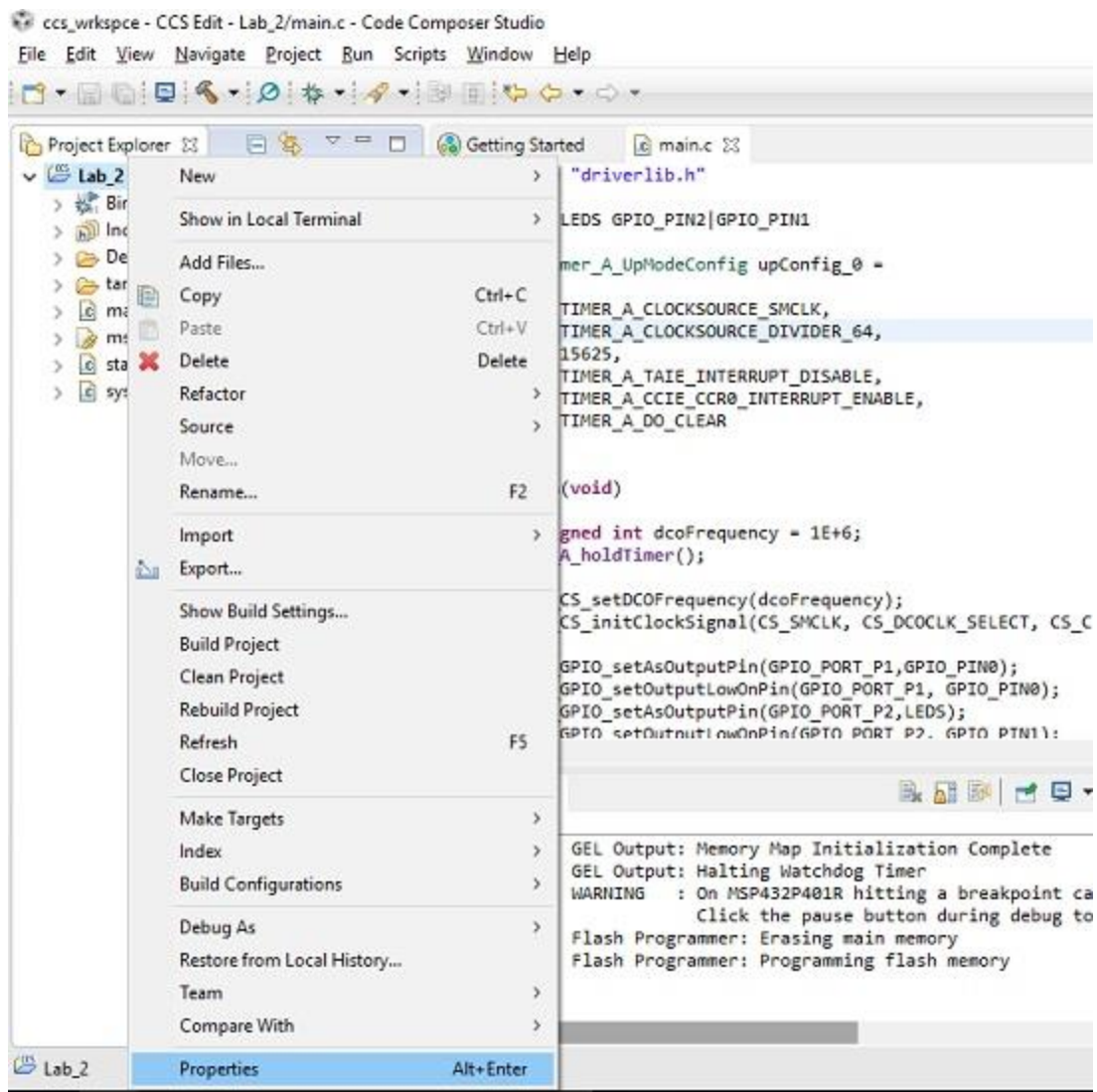
12. Next scroll down to line 140 and change the existing line to **timer_a_0_isr** as shown below:

```

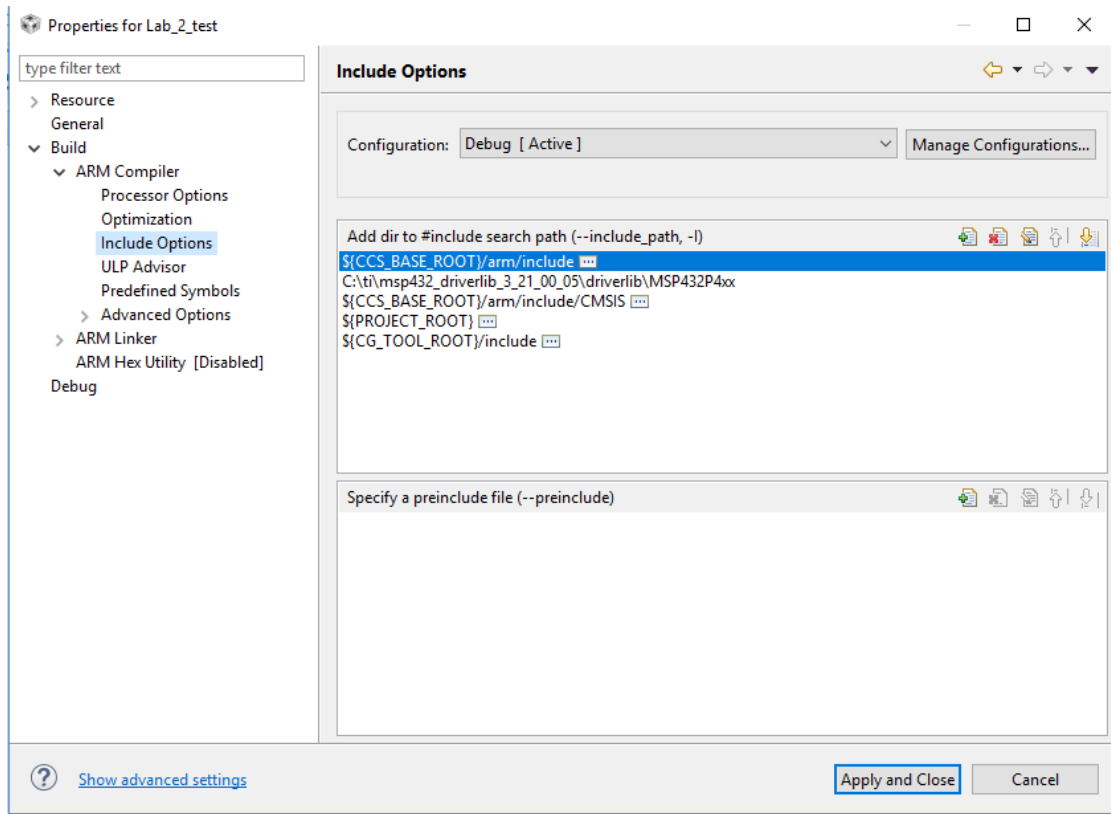
120 MemManage_Handler,      /* The MPU fault handler */
121 BusFault_Handler,       /* The bus fault handler */
122 UsageFault_Handler,     /* The usage fault handler */
123 0,                        /* Reserved */
124 0,                        /* Reserved */
125 0,                        /* Reserved */
126 0,                        /* Reserved */
127 SVC_Handler,            /* SVC call handler */
128 DebugMon_Handler,       /* Debug monitor handler */
129 0,                        /* Reserved */
130 PendSV_Handler,         /* The PendSV handler */
131 SysTick_Handler,        /* The SysTick handler */
132 PSS_IRQHandler,         /* PSS Interrupt */
133 CS_IRQHandler,          /* CS Interrupt */
134 PCM_IRQHandler,         /* PCM Interrupt */
135 WDT_A_IRQHandler,       /* WDT_A Interrupt */
136 FPU_IRQHandler,         /* FPU Interrupt */
137 FLCTL_IRQHandler,       /* Flash Controller Interrupt*/
138 COMP_E0_IRQHandler,     /* COMP_E0 Interrupt */
139 COMP_E1_IRQHandler,     /* COMP_E1 Interrupt */
140 timer_a_0_isr,          /* TA0_0 Interrupt */
141 TA0_N_IRQHandler,       /* TA0_N Interrupt */
142 TA1_0_IRQHandler,       /* TA1_0 Interrupt */
143 TA1_N_IRQHandler,       /* TA1_N Interrupt */
144 TA2_0_IRQHandler,       /* TA2_0 Interrupt */
145 TA2_N_IRQHandler,       /* TA2_N Interrupt

```

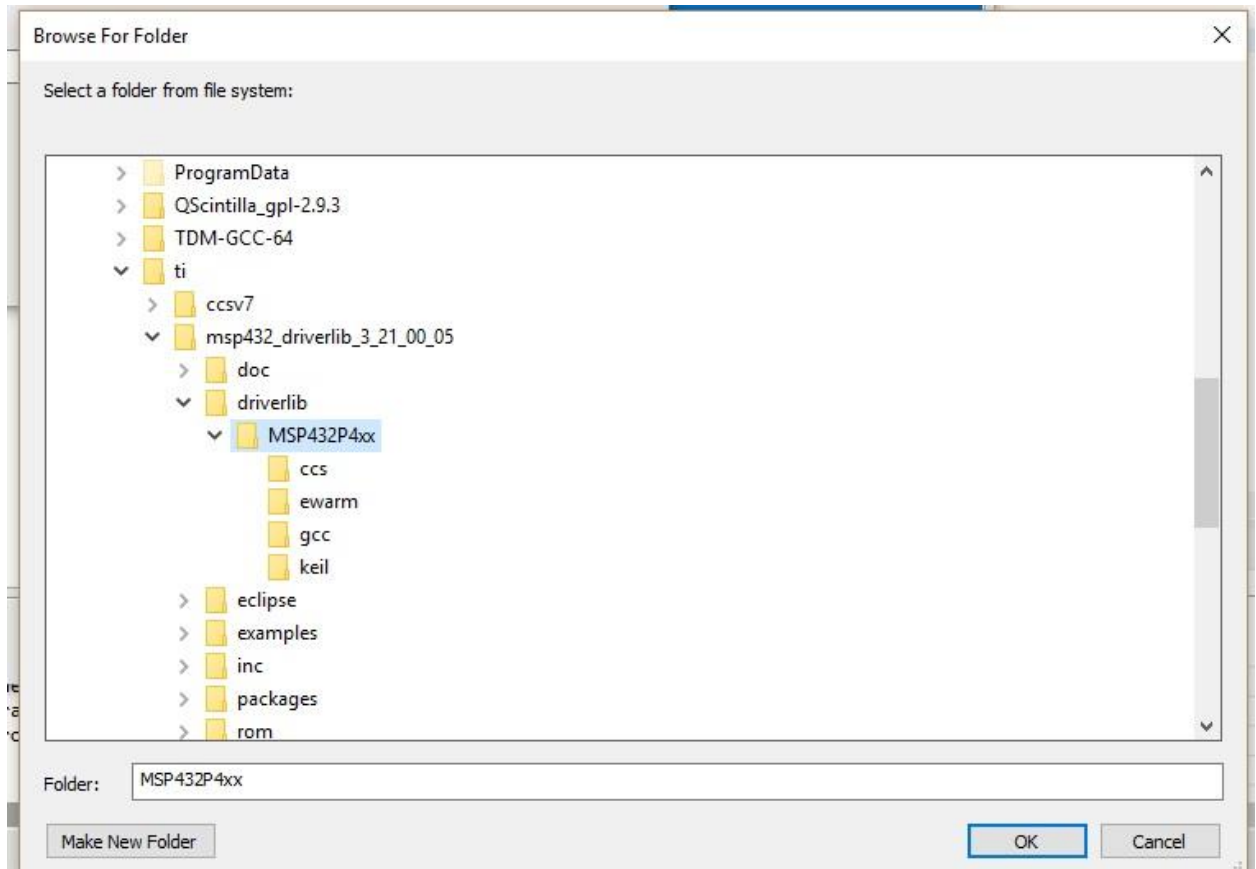
13. Now right click on the Lab_2 folder in the Project Explorer and click on **Properties**



14. Click on **Include Options** under the **ARM Compiler** tab, which is under the **Build** tab. In the adjacent mini-window with the title, “Add dir to #include search path (--include_path, -I)”, click on the green plus icon (Add) as shown below:

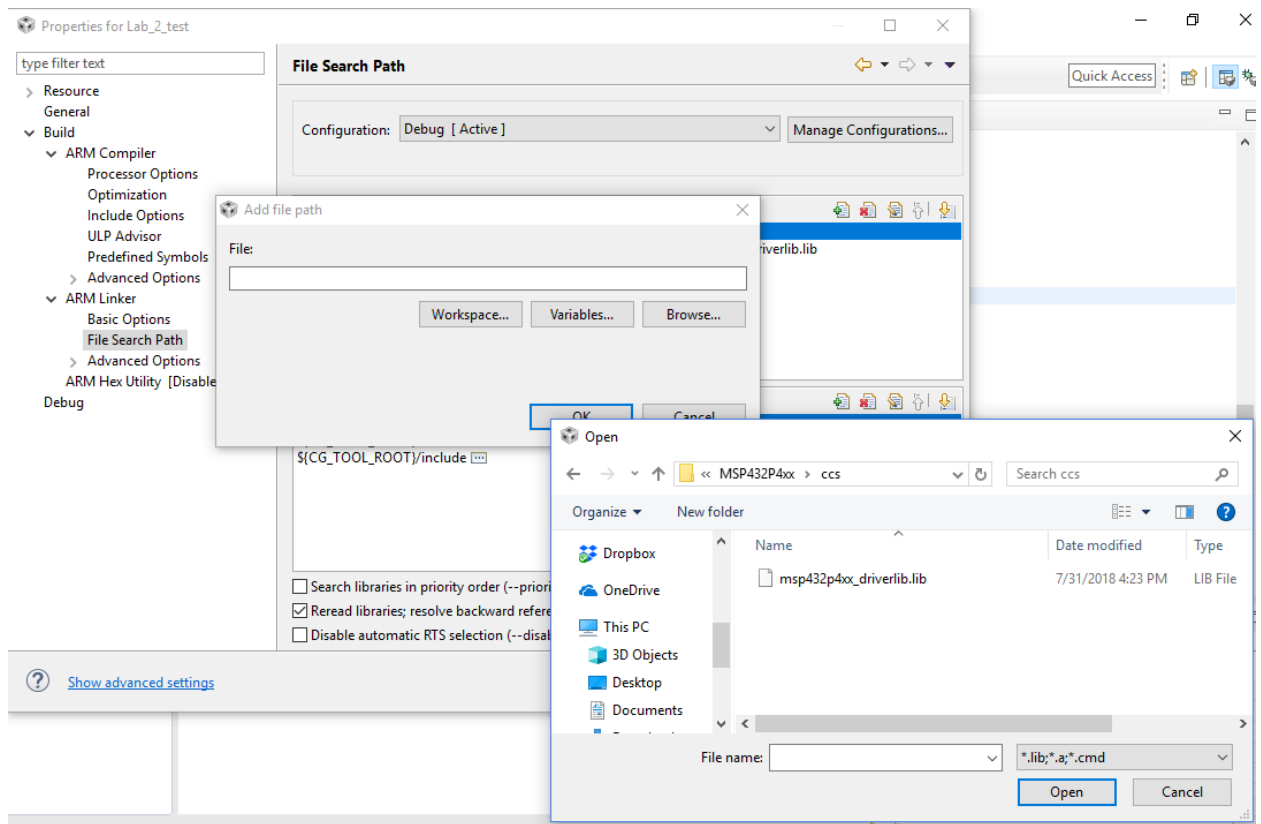


Browse to the folder msp432_driverlib_3_21_00_05 containing the extracted MSP432 driver lib as shown below:



Click on the **MSP432P4xx** folder which is within the **driverlib** folder and then click on OK as shown in the image above.

15. Similarly, under the **ARM Linker** tab, click on **File Search Path**. In the “**Include library file or command file as input**” mini-window click on the green plus icon for Add. Browse to the same **.../driverlib/MSP432P4xx/** folder as above. In this directory go to the **ccs** folder and click on **msp432p4xx_driverlib**. Click on open.



16. Finally click “Apply and Close”.

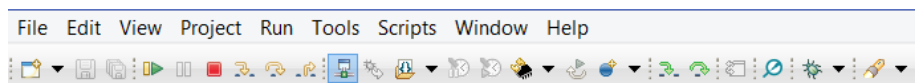
Flashing the board:

17. Click on the Debug button, shown by a green bug on the Quick Access panel below the menu bar.

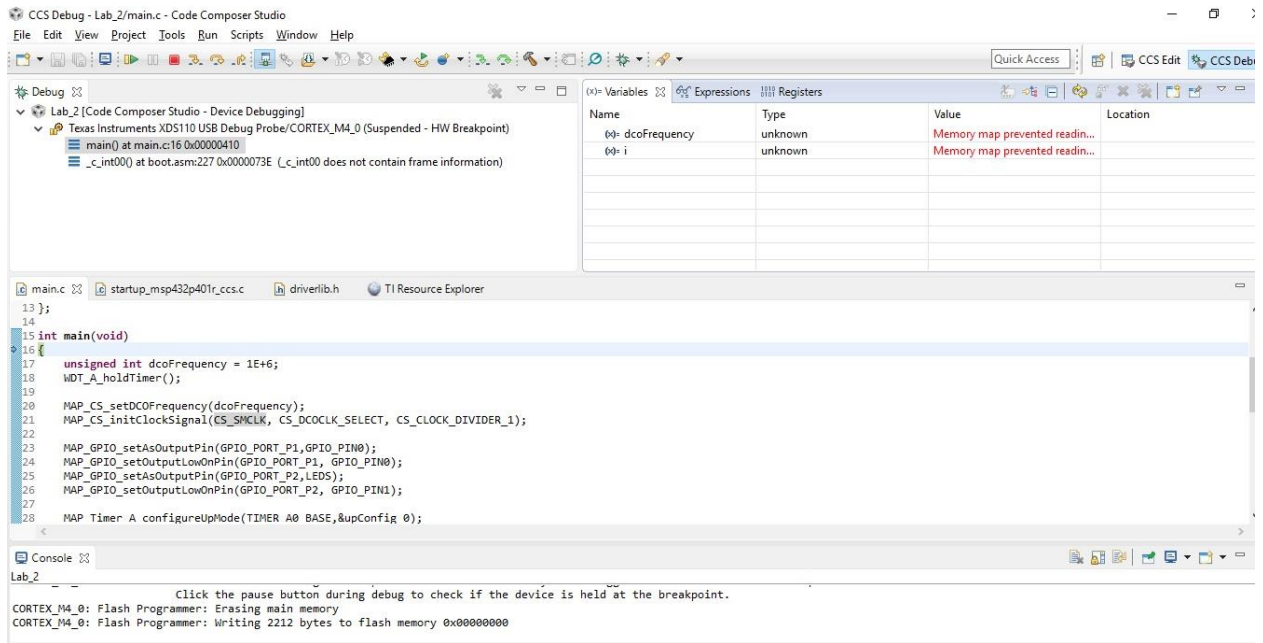


18. The project should build without errors. The project is now flashed onto the chip. This code will now execute each time the controller is powered unless it is flashed with a new code in a similar manner.

19. The perspective of the CCS will change. The CCS is now in Debug mode. Additional buttons are now available in the quick access panel.

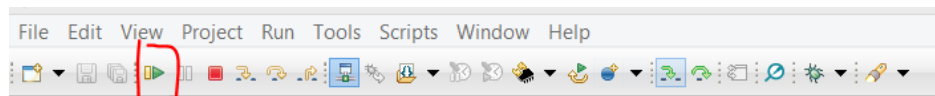


20. The code is stopped at the first line of the main function. It will continue execution if the resume button is pressed.



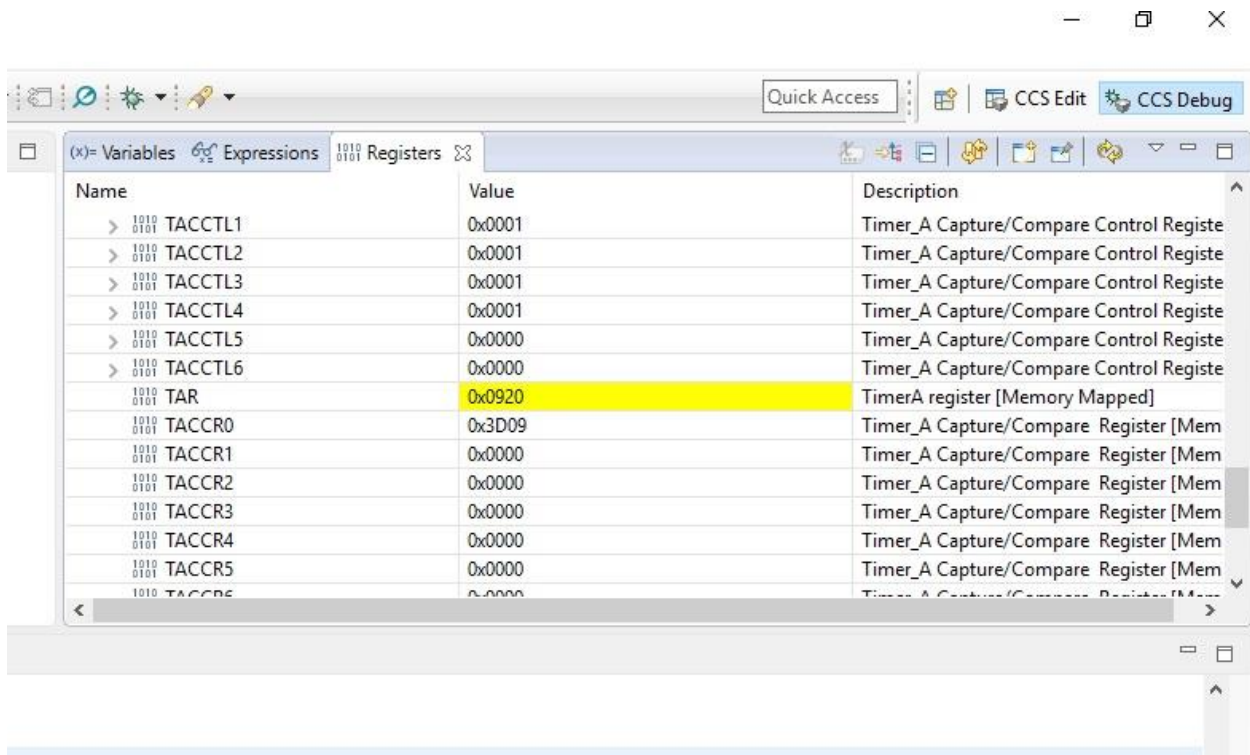
Pausing and debugging the program:

21. Press the resume button.



Both the LEDs on the board will start blinking. Pause button will suspend the execution of the code at current step.

22. The current register values can be checked in the registers window embedded in the screen. 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.



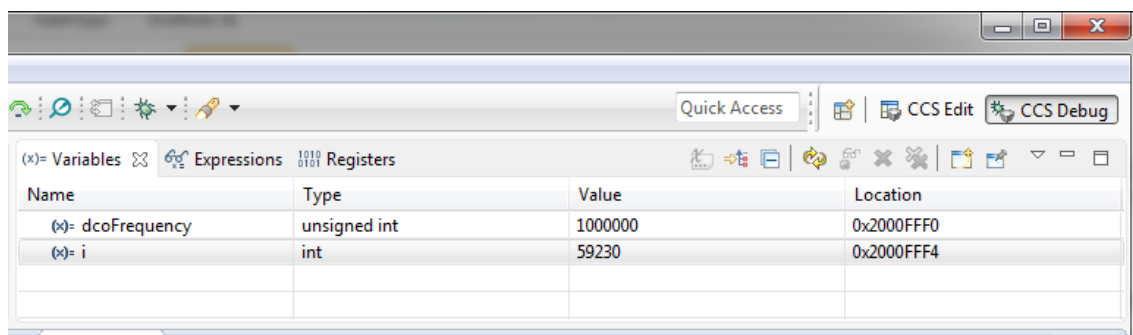
23. This window can also be used to check the values of variables and place breakpoints during debugging. It may also be observed that under the variables tab, the variable *i* changes value every time the program execution is paused.
24. When paused, code can be executed step by step using the step button. Step by step execution causes the registers whose values have changed to be highlighted in yellow. The Step Into button or F5 (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 or F6 (next to the Step Into button).
25. The Terminate button (red square) terminates the debug session.
26. To set up the debug session to answer the questions below, terminate the debugging session (if you are running one). Set breakpoints at line 37 and 41. This can be done by double clicking next to the line number in the blue bar. Refer to the image below.

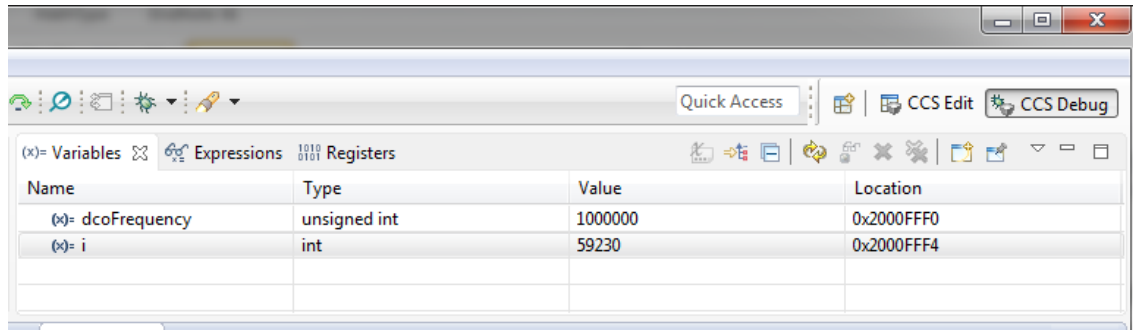
```

22
23  MAP_GPIO_setAsOutputPin(GPIO_PORT_P1,GPIO_PIN0);
24  MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);
25  MAP_GPIO_setAsOutputPin(GPIO_PORT_P2,LEDS);
26  MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN1);
27
28  MAP_Timer_A_configureUpMode(TIMER_A0_BASE,&upConfig_0);
29
30  MAP_Interrupt_enableSleepOnIsrExit();
31  MAP_Interrupt_enableInterrupt(INT_TA0_0);
32
33  MAP_Timer_A_startCounter(TIMER_A0_BASE, TIMER_A_UP_MODE);
34
35  MAP_Interrupt_enableMaster();
36
37  int i=10;
38  while(1)
39  {
40      i += 10;
41      if(i > 65000)
42      {
43          i=10;
44      }
45  }
46 }
47

```

27. Now press the debug button. Once the code is paused in the main function, press F8 to progress to line 37. The program should pause execution at line 37.
28. Click on the Variables tab in the watch window as shown below. Two variables should be visible: dcoFrequency and i. (The value of i in the picture may not match your initial value of i.)





Questions:

Debugging Questions

1. In the variables window, right click on the Value property of 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). **(10 pts)**
2. You should currently be at line 37 after pressing F8 after starting a new debug session as per the instructions. Now press F5 (Step Into) five times, monitoring the value of *i* in the Variables window. What is the value of *i* after pressing F5 five 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)? **(10 pts)**

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 53 of Lecture 2, with appropriate modifications. Assume that the MSP432 GPIO pins will supply the input V_{in}) **(30 pts)**
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 the important details** in the datasheet of the device that proves the usability of the device. **(5 pts)**
5. Find a solid state (semiconductor) switching device on the internet that cannot be used in the above circuit. **Attach the datasheet** for this device. **Point out the important details** in the datasheet which show that this component is not suited to this application. **(5 pts)**