

EMBSYS 330A

STM32CUBEIDE SETUP INSTRUCTIONS

BASED ON INSTRUCTIONS FROM EMBSYS 330 A MODULE 01 BY LAWRENCE LO

INTRODUCTION

For EMBSYS 330 we will be using the official development tools offered by STMicro: STM32CubeIDE.

STM32CubeIDE is based on the open-source Eclipse toolchain, using the GNU/GCC toolchain and OpenOCD for debugging. Eclipse is a popular alternative IDE to commercial products such as IAR and other free tools such as Visual Studio Code.

For this course we will be using the latest version as of time of writing 1.11.0



















DOWNLOADING

Download the software from the STMicroelectronics website here:

<https://www.st.com/en/development-tools/stm32cubeide.html>

Download STM32CubeIDE-Win and select version 1.11.0 in the dropdown box to ensure you download the version required for this course.

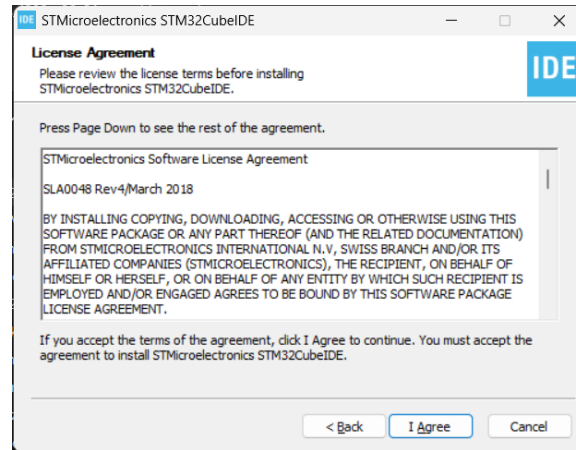
Get Software

Part Number	General Description	Supplier	Download	All versions
 STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	ST		Select version 
 STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	ST		Select version 
 STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	ST		Select version 
 STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	ST		Select version 
 STM32CubeIDE-Win	STM32CubeIDE Windows Installer	ST		Select version  <div> 1.11.0  1.10.1  1.10.0</div>

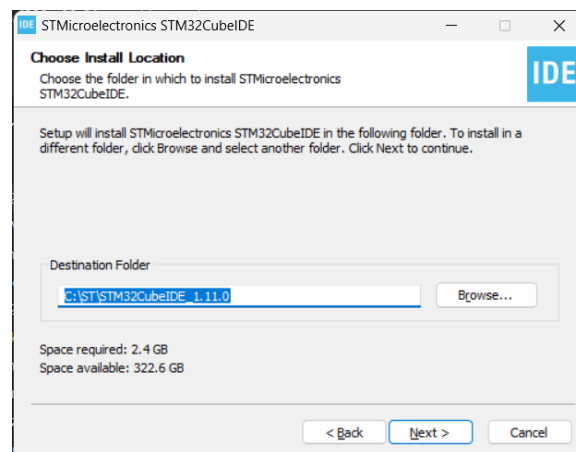
To download the software, you will need to accept the license agreement and log in to your ST Microelectronics account, which you should have from downloading STM32CubeMX in the first semester.

INSTALLATION

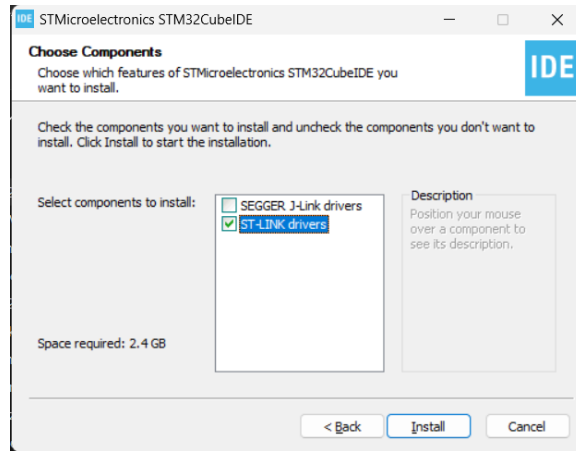
1. Unzip the downloaded installation file and run the installer.
2. Review the license agreement and accept it.



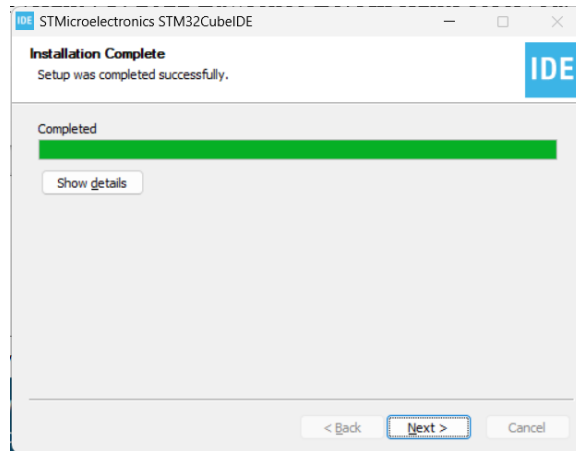
3. Keep the install location as the one suggested: C:\ST\STM32CubeIDE_1.11.0



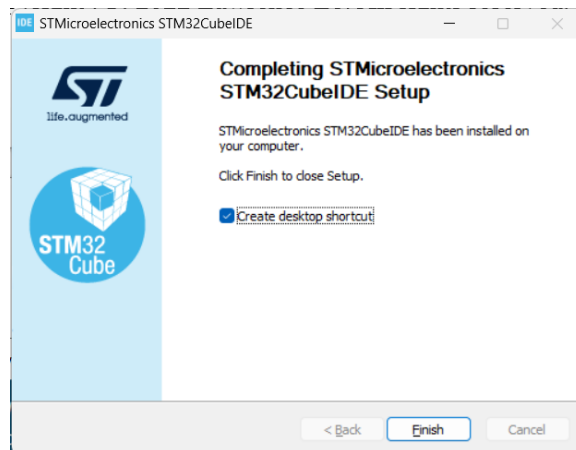
4. On the Choose Components Dialog, ensure ST-LINK drivers are selected. It won't hurt to install the SEGGER J-Link drivers but they will not be used in this course



5. Press Install. The installation window will ask several times for permission to install the ST-Link device drivers. Agree to installation in all cases.
6. Eventually the dialog box will show "Installation Complete". Press "Next >":



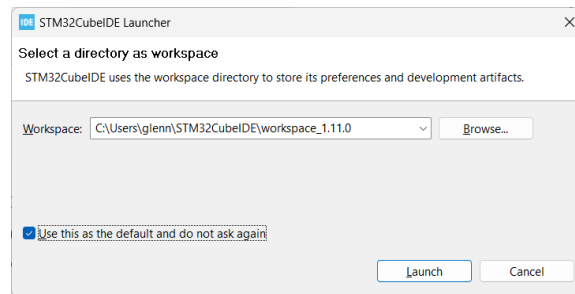
7. You can choose to make a desktop shortcut or not depending on your preferred workflow:



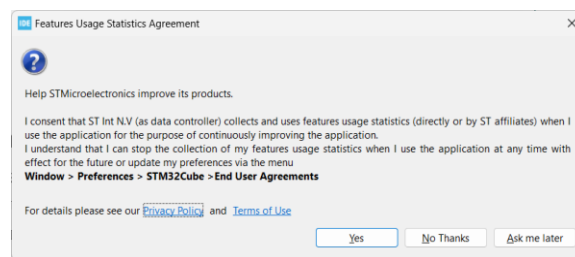
8. Press "Finish" to complete installation

FIRST RUN

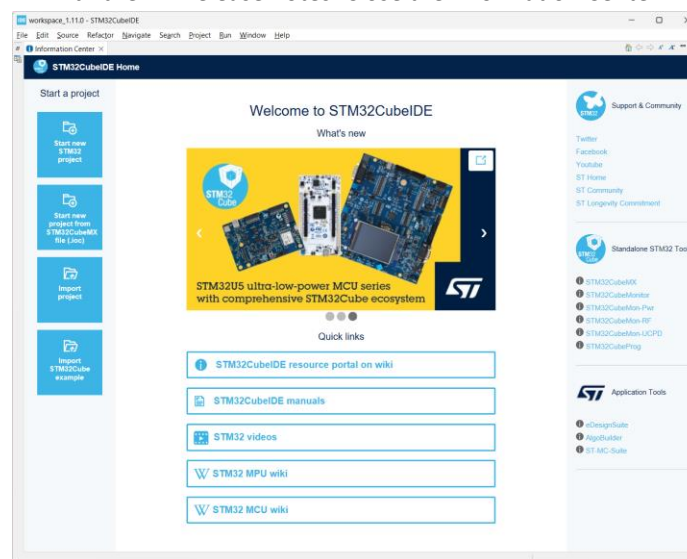
1. STM32CubeIDE will ask to select a directory to use for its workspace. The workspace is where IDE projects are stored, which may be separate from the source code location. The projects in a workspace share common global properties, which allows you to change environments by switching to a different workspace. Use the default location suggested, which for this version is C:\Users\<username>\STM32CubeIDE\workspace_1.11.0 and also check the “Use this as the default and do not ask again” checkbox. This can be undone later if you want to experiment with other environments.



2. Click the “Launch” button to start the main window
3. You may choose to send your usage statistics to STM or not:



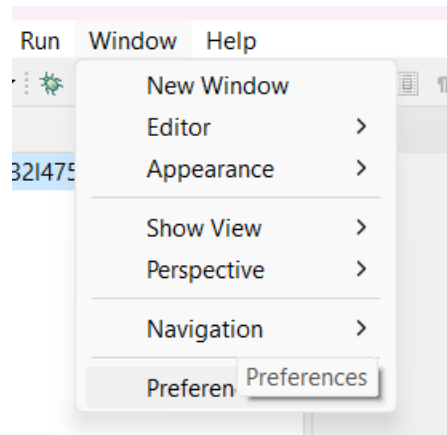
4. STM32CubeIDE starts up displaying the Information Center. On first run it will also open a web browser with the IDE release notes. Close the Information Center.



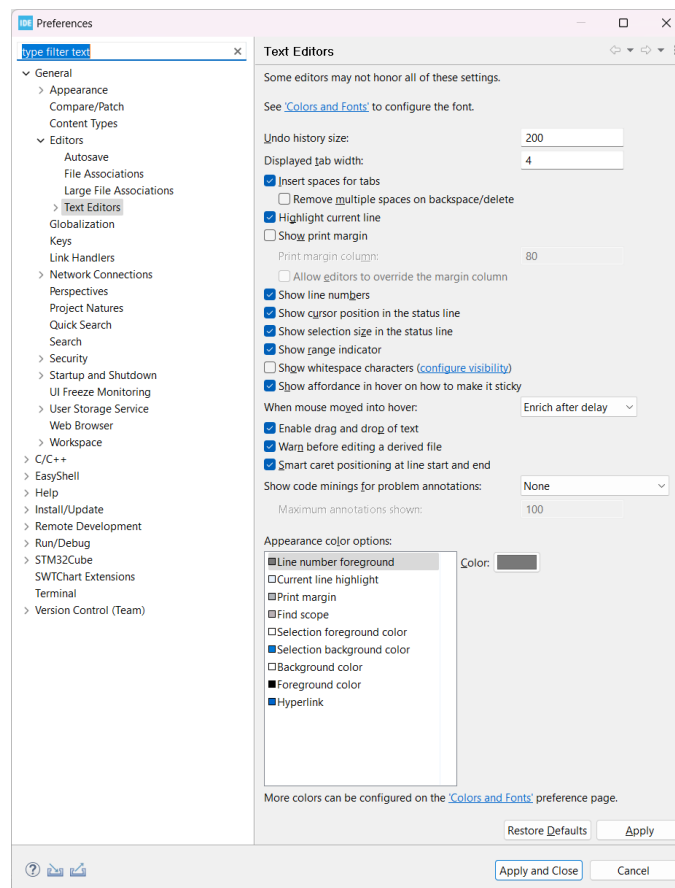
SETTING TAB STOPS

By default, the editor inserts tabs instead of individual spaces when the Tab key is pressed. This messes up the layout when imported to GitHub.

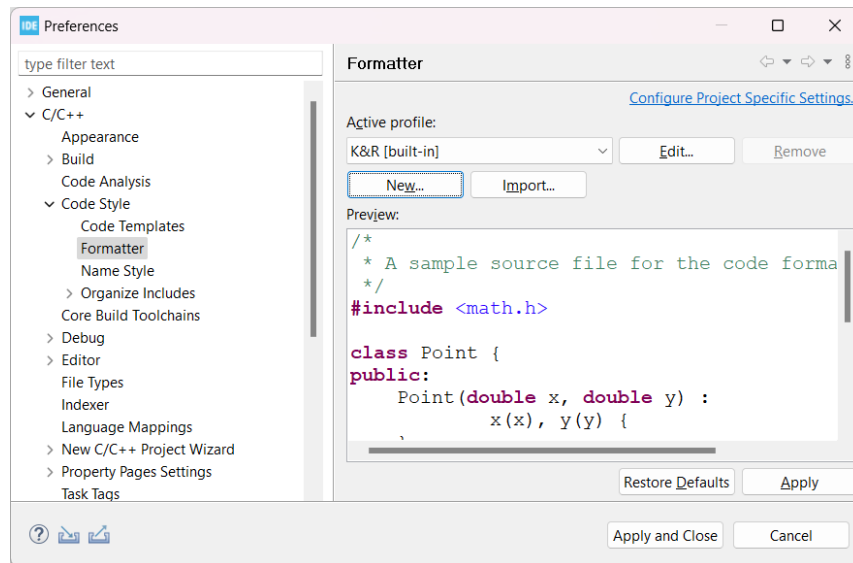
1. Go to the Window Menu at the top of the screen and select Preferences from the bottom:



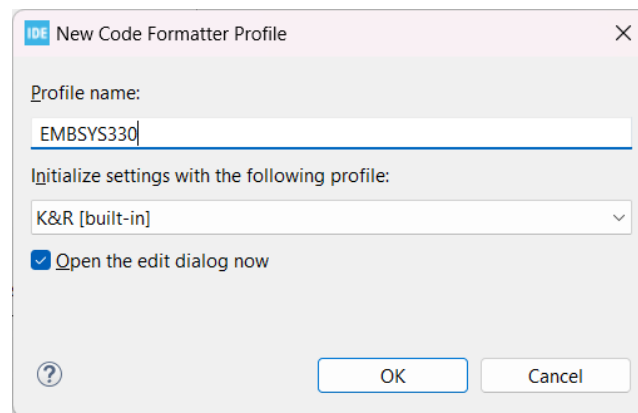
2. In the preferences menu navigate on the left pane to General->Editors->Text Editors
3. Select the “Insert Spaces for Tabs” check box.



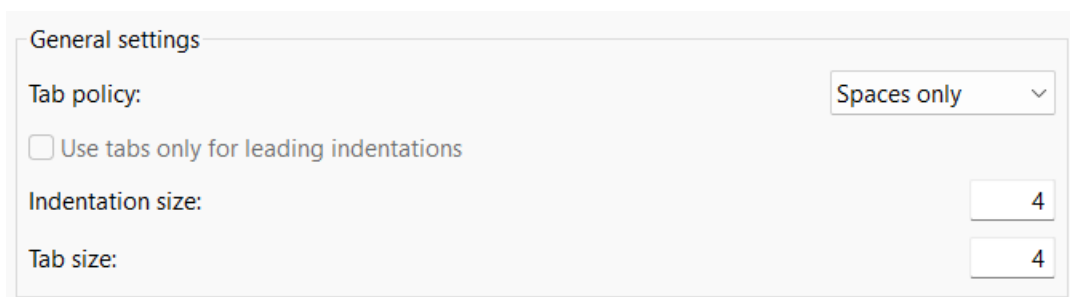
4. Navigate to C/C++->Code Syles->Formatter
5. On the Formatter pane, click the “New” button:



6. In the window that appears, provide a Profile name of EMBSYS330 and confirm “Initialize settings with the following profile:” is set to “K&R [built-in]” and the “Open the edit dialog now” box is checked, then click on OK.



7. In the Indentation tab, under General Settings change the “Tab policy” to “Spaces only”:

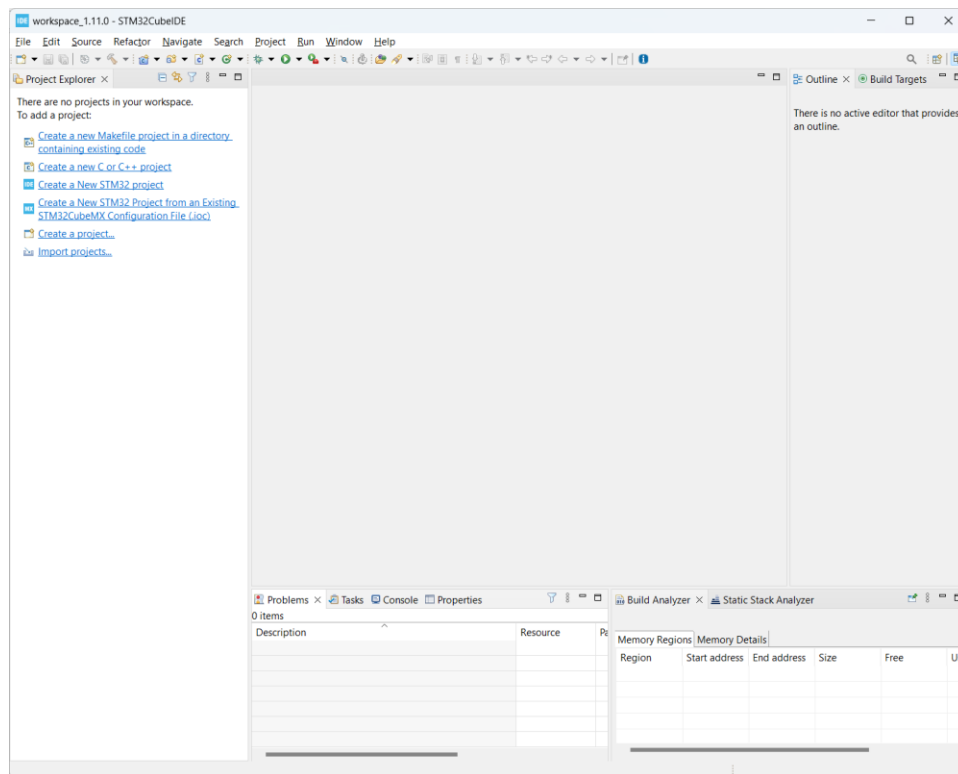


8. Click on OK to close the edit window, then on “Apply and Close” to close the Preferences window.

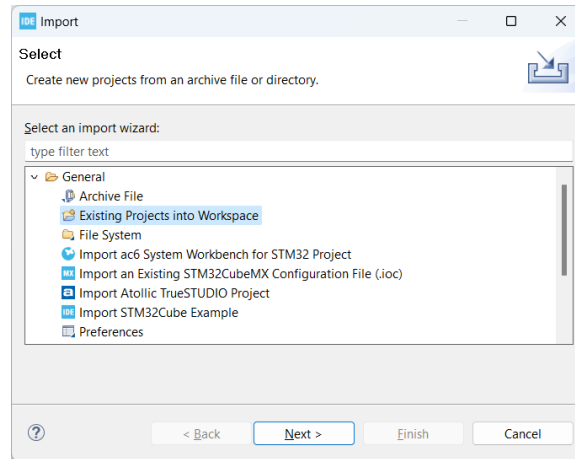
IMPORTING THE PROJECT FILES

For this course we will be using STM32CubeIDE projects that set up the QP framework for you, available on the UW GitHub repository for this course.

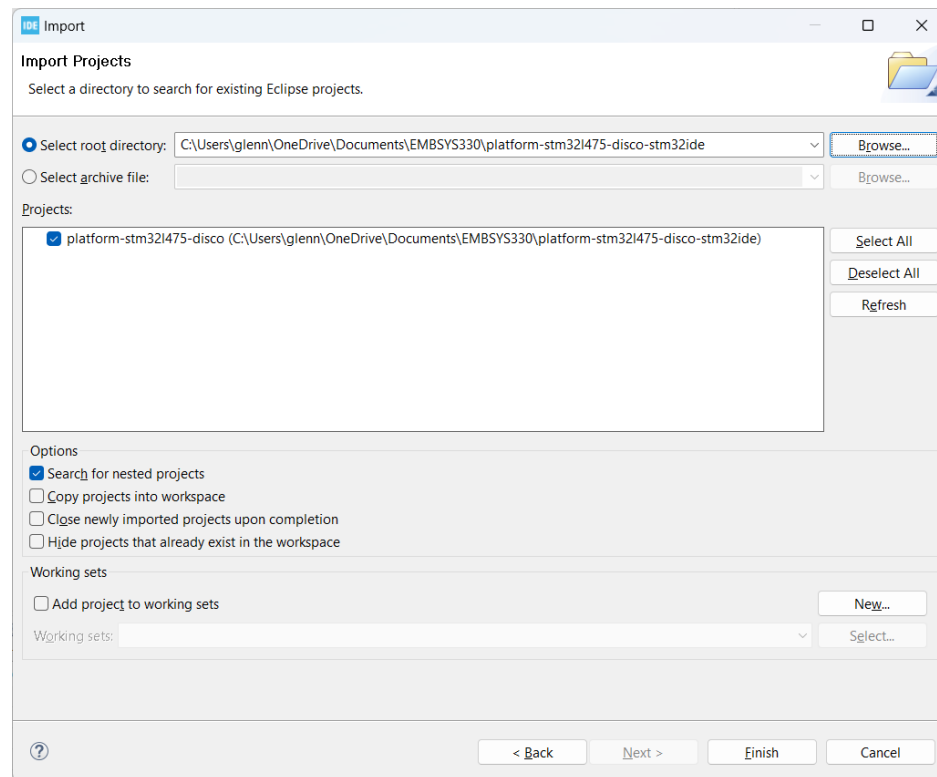
1. Clone the project from the UW GitHub Repository via GitHub Classroom
2. The project has long paths that can stretch the 255-character path name limit of Windows. It is recommended that the projects are cloned to a directory named EMBSYS330 in either the C:\ top level directory or in Documents
3. The name of the project folder should be is platform-stm32l475-disco-stm32ide
4. If the Project Explorer panel is not visible, show it by selecting *Window->Show View->Project Explorer*



5. Click on "Import projects..." at the bottom of the Project Explorer panel
6. Select *General->Existing Projects Into Workspace* and press "Next >"



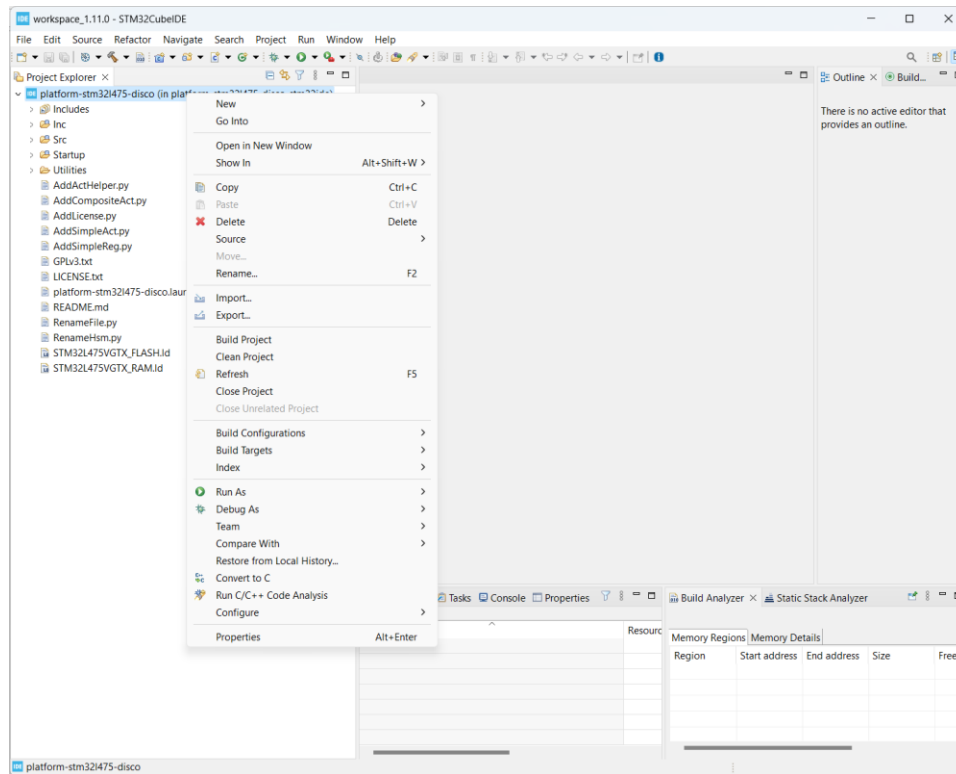
7. A directory selection dialog box will appear. Navigate to the top-level directory of your unzipped project and select it.
8. The Import window will appear. There should be a single project selected in the Projects window.



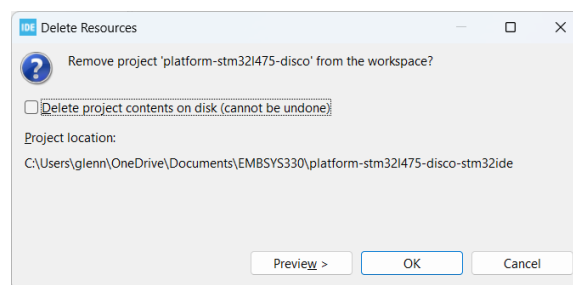
9. Leave all Options as their defaults. Do not select the “Copy projects into workspace” check box if you want to keep the source files in the folder monitored by Git
10. Click on the “Finish” button
11. The imported *platform-stm32i475-disco* project appears in the Project Explorer panel. Click on the > icon next to the project name to expand the project folders.

BUILDING THE PROJECT

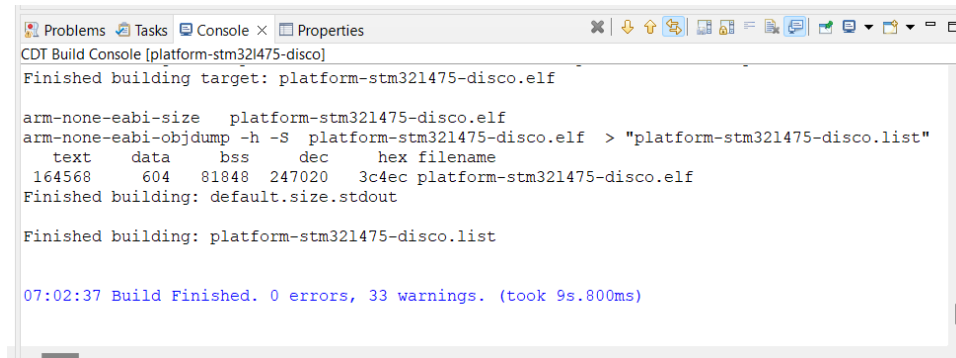
1. Right Click on the project in Project Explorer. A context menu will appear



2. Here are some important context-menu items:
 - a. Build Project – compiles and links the source code to generate the executable files in elf and bin format
 - b. Clean Project – deletes the executables and all intermediate files. You do not want to be copying these files up to GitHub
 - c. Properties – Contains project settings such as compiler and linker flags and environment variables among many other things
 - d. Delete – deletes the project files from the workspace. This can be useful if you have identically-named projects in different folders and have to switch from one to the other. Note: Make sure the “Delete project contents on disk” checkbox is UNCHECKED if you are just switching between projects and do not want all your source code deleted



3. Click on Build Project and wait for the project to build. Eventually the Console panel should show an output similar to that below:



```
CDT Build Console [platform-stm321475-disco]
Finished building target: platform-stm321475-disco.elf

arm-none-eabi-size    platform-stm321475-disco.elf
arm-none-eabi-objdump -h -S platform-stm321475-disco.elf > "platform-stm321475-disco.list"
   text    data     bss     dec     hex filename
 164568    604   81848  247020  3c4ec platform-stm321475-disco.elf
Finished building: default.size.stdout

Finished building: platform-stm321475-disco.list

07:02:37 Build Finished. 0 errors, 33 warnings. (took 9s.800ms)
```

4. Some warnings (caused by third-party libraries) are acceptable for this exercise but there should be no errors and the build should finish successfully

DEBUGGER SETUP

There are three toolbar buttons controlling debugging and running the program:

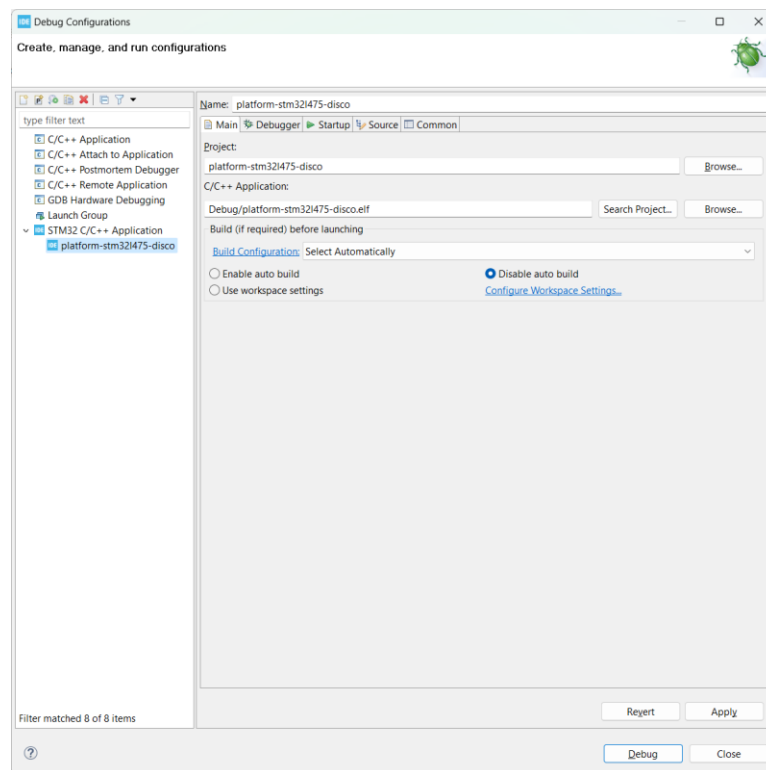


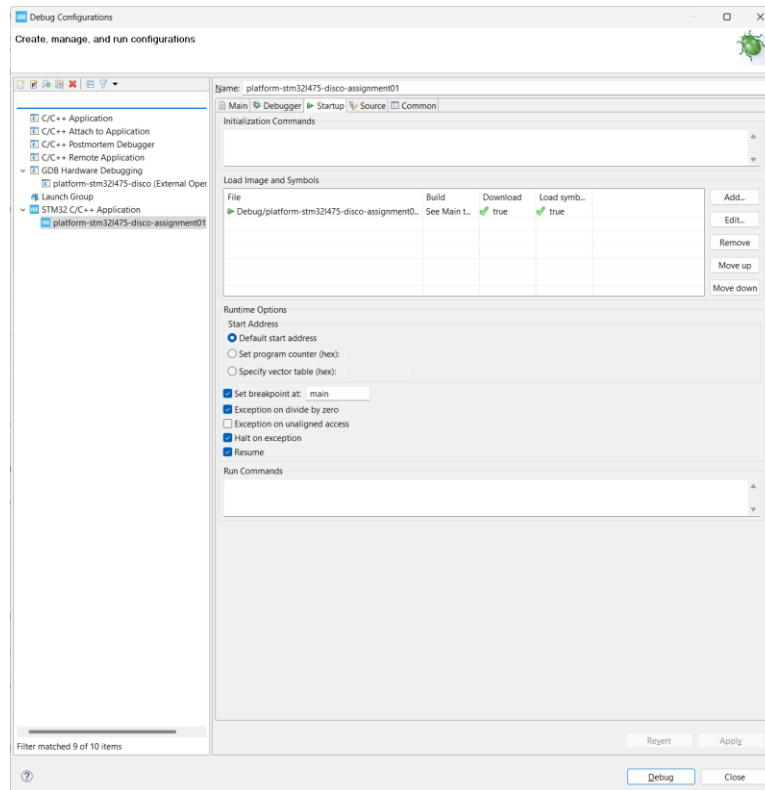
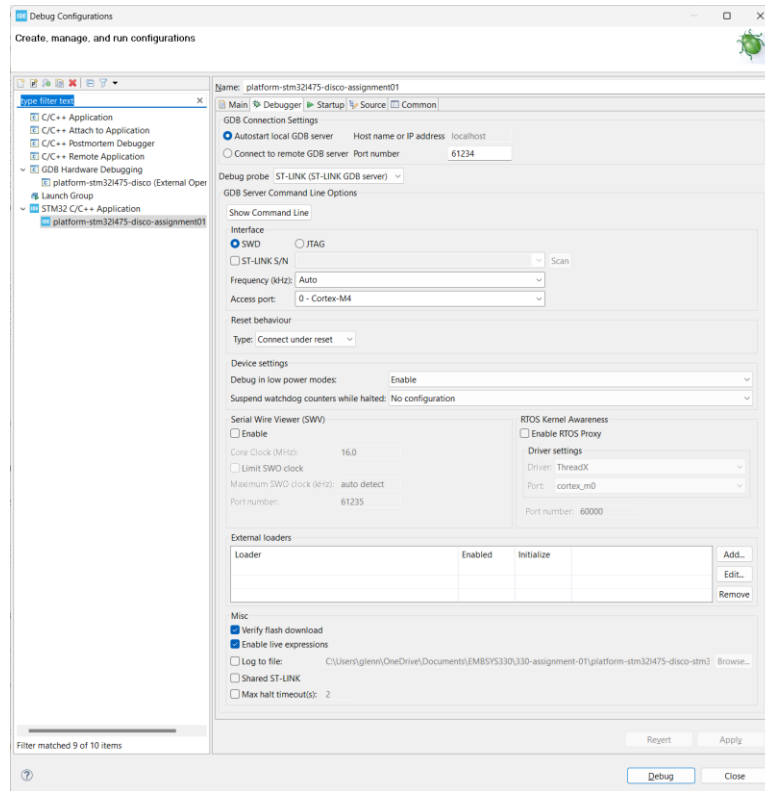
From left to right they are “Debug”, “Run” and “Run last tool”

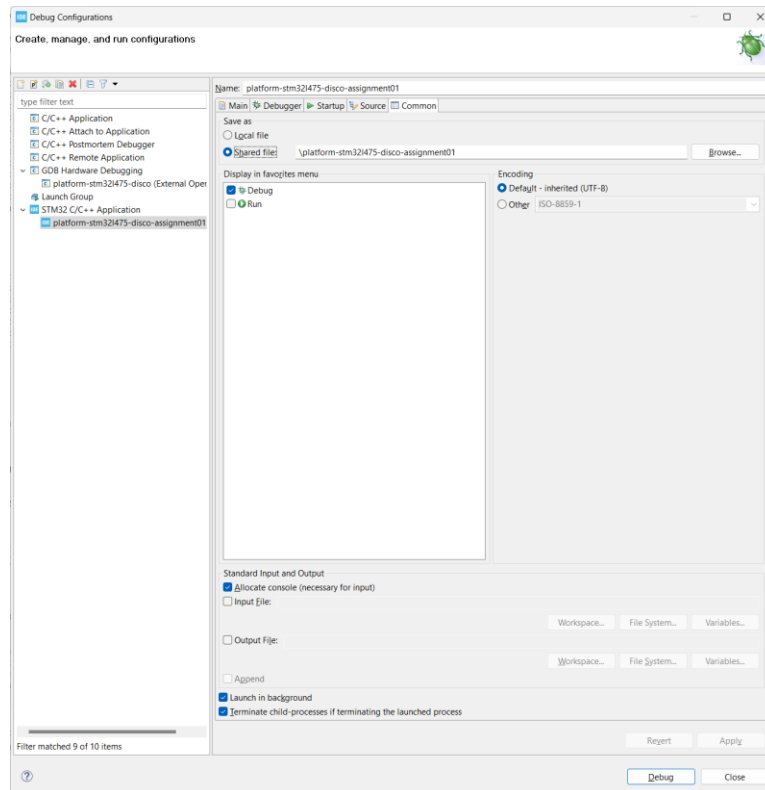
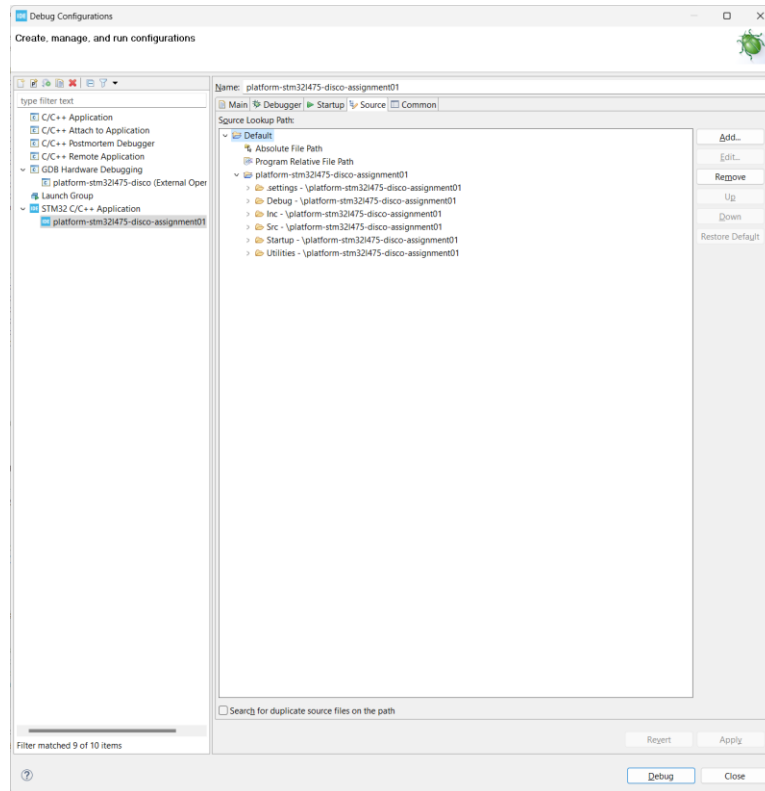
“Debug” downloads the program to the board and attaches the debugger, while “Run” downloads and runs the program without attaching the debugger.

Each button has a down-arrow that opens a sub-menu.

1. Click the down-arrow next to the “Debug” button to open the submenu and select “Debug Configurations...”
2. On the panel on the left, click the > next to “STM32 C/C++ Application” and select the “platform-stm32l475-disco” entry
3. Check all five tabs are set up as in the screenshots below (They should be pre-set by the project)

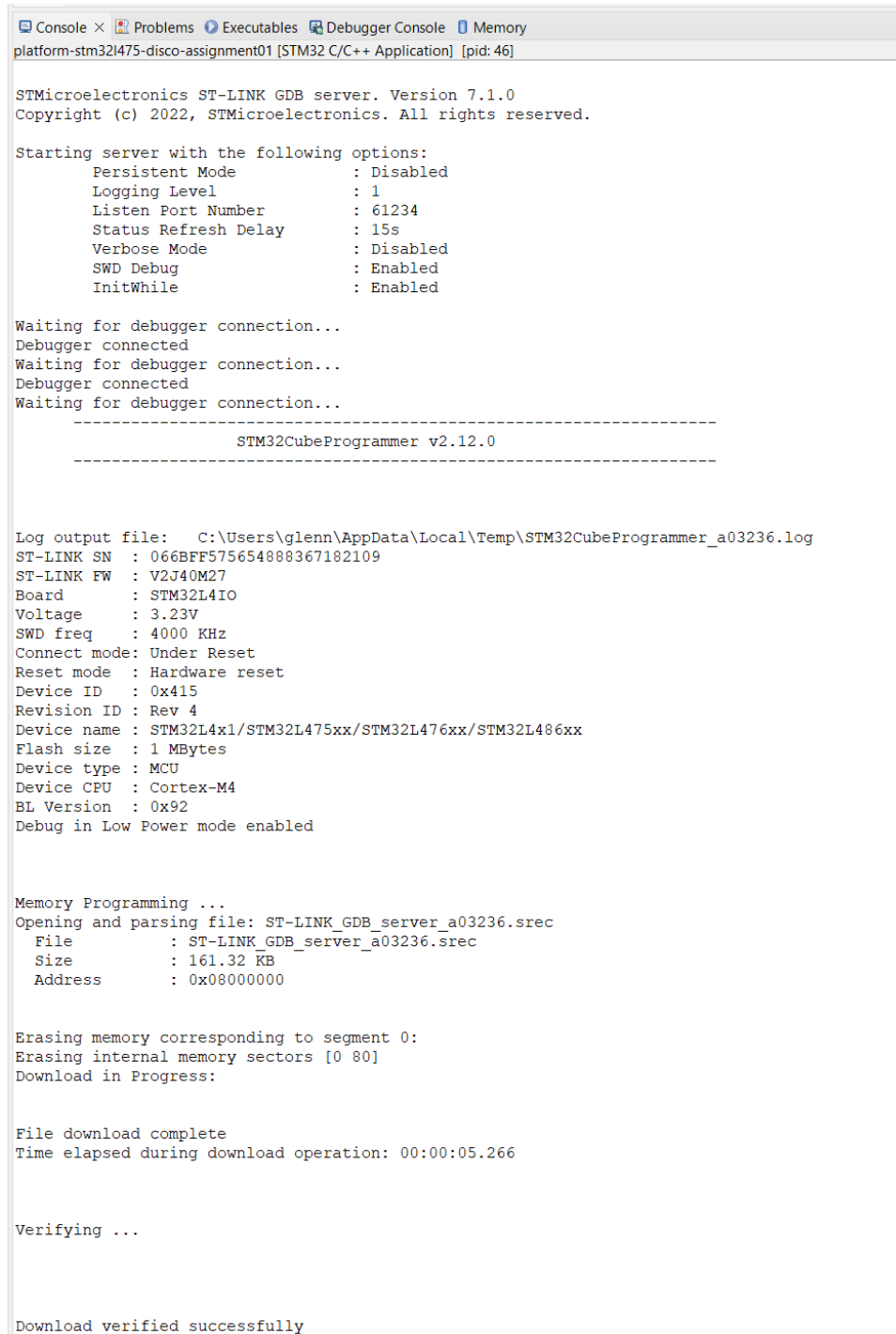






- Once the settings are confirmed, press the Debug button at the lower right of the dialog

5. The IDE will ask if you want to switch to the Debug perspective. It is recommended to say “yes” and to check the box to not ask again (this can be reversed in Preferences if desired)
6. The debug console will print output like this:



```
platform-stm32l475-disco-assignment01 [STM32 C/C++ Application] [pid: 46]

STMicroelectronics ST-LINK GDB server. Version 7.1.0
Copyright (c) 2022, STMicroelectronics. All rights reserved.

Starting server with the following options:
  Persistent Mode      : Disabled
  Logging Level       : 1
  Listen Port Number  : 61234
  Status Refresh Delay: 15s
  Verbose Mode        : Disabled
  SWD Debug           : Enabled
  InitWhile            : Enabled

Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...
Debugger connected
Waiting for debugger connection...

-----
                        STM32CubeProgrammer v2.12.0
-----

Log output file: C:\Users\glenn\AppData\Local\Temp\STM32CubeProgrammer_a03236.log
ST-LINK SN : 066BFF575654888367182109
ST-LINK FW : V2J40M27
Board      : STM32L4IO
Voltage    : 3.23V
SWD freq   : 4000 KHz
Connect mode: Under Reset
Reset mode : Hardware reset
Device ID  : 0x415
Revision ID: Rev 4
Device name: STM32L4x1/STM32L475xx/STM32L476xx/STM32L486xx
Flash size : 1 MBytes
Device type : MCU
Device CPU  : Cortex-M4
BL Version  : 0x92
Debug in Low Power mode enabled

Memory Programming ...
Opening and parsing file: ST-LINK_GDB_server_a03236.srec
  File      : ST-LINK_GDB_server_a03236.srec
  Size      : 161.32 KB
  Address   : 0x08000000

Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 80]
Download in Progress:

File download complete
Time elapsed during download operation: 00:00:05.266

Verifying ...

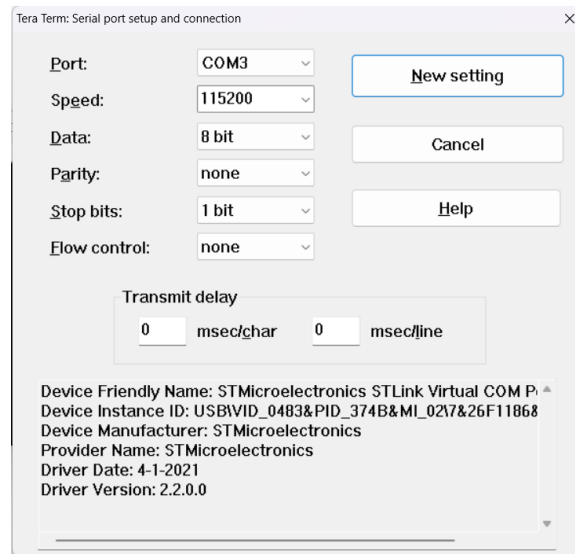
Download verified successfully
```

7. Download of the program and setting up the debugger should take under 30 seconds
8. The source code window should show main.cpp, with the debugger stopped at the first line of SystemClock_Config()

RUNNING THE DEBUGGER

Before running the program, we want to open a terminal window to display the output. We will use Tera Term with the same output settings as previously 115200 8-none-1

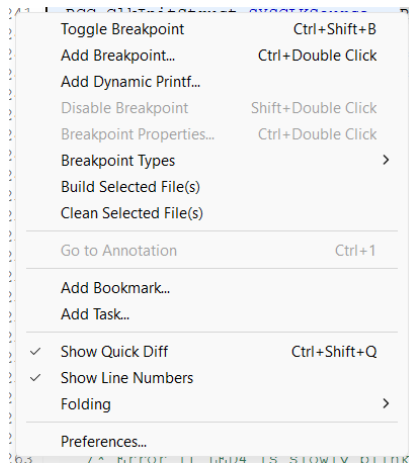
1. Open Tera Term and if not already selected, select the COM port with the name “STMicroelectronics STLink Virtual COM Port”
2. Go to the Setup->Serial port menu and open the setup dialog box.
3. Confirm the following settings (remembering the COM port number may be different)



4. The source code window should show main.cpp, with the debugger stopped at the first line of `SystemClock_Config()`. The Debug tab on the left shows the debugger halted in `main()` because the first thing `main()` does is call `SystemClock_Config()`.
5. The main debugging commands are available from the toolbar:



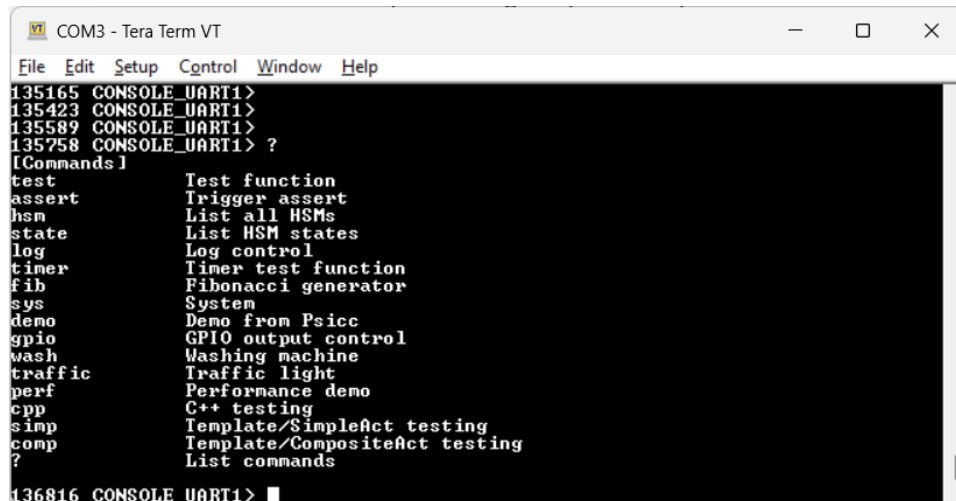
- a. To start the program running press the yellow/green triangle “Resume” button
 - b. To pause execution once running, press the “Suspend” button to the right of “Resume”
 - c. To terminate the debugger and go back to coding, press the red square “Terminate” button
 - d. The “Disconnect” button keeps the target running, but detaches it from the debugger
 - e. The three arrows step through instructions while suspended.
 - i. If the present instruction is a function call, “Step Into” will enter to the function being called
 - ii. If the present instruction is a function call, “Step Over” will completely run the called function and halt at the next instruction in the same function as the present instruction
 - iii. “Step Return” will exit the current function by running until the function returns and will halt at the next instruction in the calling function
6. There is a breakpoint menu accessible by right-clicking on the line numbers to the left of the source lines that allows you to add or remove breakpoints, disable breakpoints without removing them, and edit the breakpoint properties. You can toggle a breakpoint by double-clicking the line number



7. As with any debugger, variables can be watched, CPU registers and peripheral registers can be viewed as can the contents of memory
8. Press the Resume button to begin program execution. Tera Term should display large amount of startup information that finishes as shown in the screenshot:

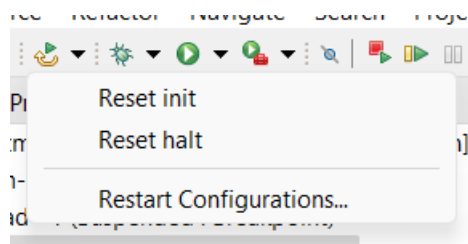
```
COM3 - Tera Term VT
File Edit Setup Control Window Help
225 SENSOR_PRESS(22): Starting DONE from SENSOR_PRESS(22) seq=0
225 SENSOR_PRESS(22): Starting EXIT
225 SENSOR_PRESS(22): Started ENTRY
226 SENSOR(18): Starting SENSOR_MAG_START_CFM from SENSOR_MAG(21) seq=2
226 SENSOR(18): Starting SENSOR_HUMID_TEMP_START_CFM from SENSOR_HUMID_TEMP(20)
seq=3
226 SENSOR(18): Starting SENSOR_PRESS_START_CFM from SENSOR_PRESS(22) seq=4
226 COMPOSITE_ACT(39): Starting EXIT
227 SENSOR(18): Starting SENSOR_ACCEL_GYRO_START_CFM from SENSOR_ACCEL_GYRO(19)
seq=1
227 SENSOR(18): Starting DONE from SENSOR(18) seq=5
227 SENSOR(18): Starting EXIT
227 SENSOR(18): Started ENTRY
227 SYSTEM(1): Starting2 SENSOR_START_CFM from SENSOR(18) seq=7
227 SYSTEM(1): Starting2 NEXT from SYSTEM(1) seq=8
228 SYSTEM(1): Starting2 EXIT
228 SYSTEM(1): Starting3 ENTRY
228 SYSTEM(1): Starting3 LEVEL_METER_START_CFM from LEVEL_METER(36) seq=9
228 SYSTEM(1): Starting DONE from SYSTEM(1) seq=10
228 SYSTEM(1): Starting3 EXIT
228 SYSTEM(1): Starting EXIT
228 SYSTEM(1): Started ENTRY
229 COMPOSITE_ACT(39): Started ENTRY
```

9. You can type a question mark (?) and press return to print a list of commands



```
COM3 - Tera Term VT
File Edit Setup Control Window Help
135165 CONSOLE_UART1>
135423 CONSOLE_UART1>
135589 CONSOLE_UART1>
135758 CONSOLE_UART1> ?
[Commands]
test          Test function
assert        Trigger assert
hsm           List all HSMs
state         List HSM states
log           Log control
timer         Timer test function
fib           Fibonacci generator
sys           System
demo          Demo from Psicc
gpio          GPIO output control
wash          Washing machine
traffic       Traffic light
perf          Performance demo
cpp           C++ testing
simp          Template/SimpleAct testing
comp          Template/CompositeAct testing
?             List commands
136816 CONSOLE_UART1>
```

10. Type fib and press enter to run the Fibonacci generator. Press enter again to quit the demo and return to the console.
11. To reset the application to the start of main without downloading the program and initializing OpenOCD again, press the “Reset Init” submenu from the reset button down-arrow:



12. To finish the debug session, hit the red square “Terminate” button. The IDE will revert from the Debugger perspective to the C/C++ perspective to facilitate coding.