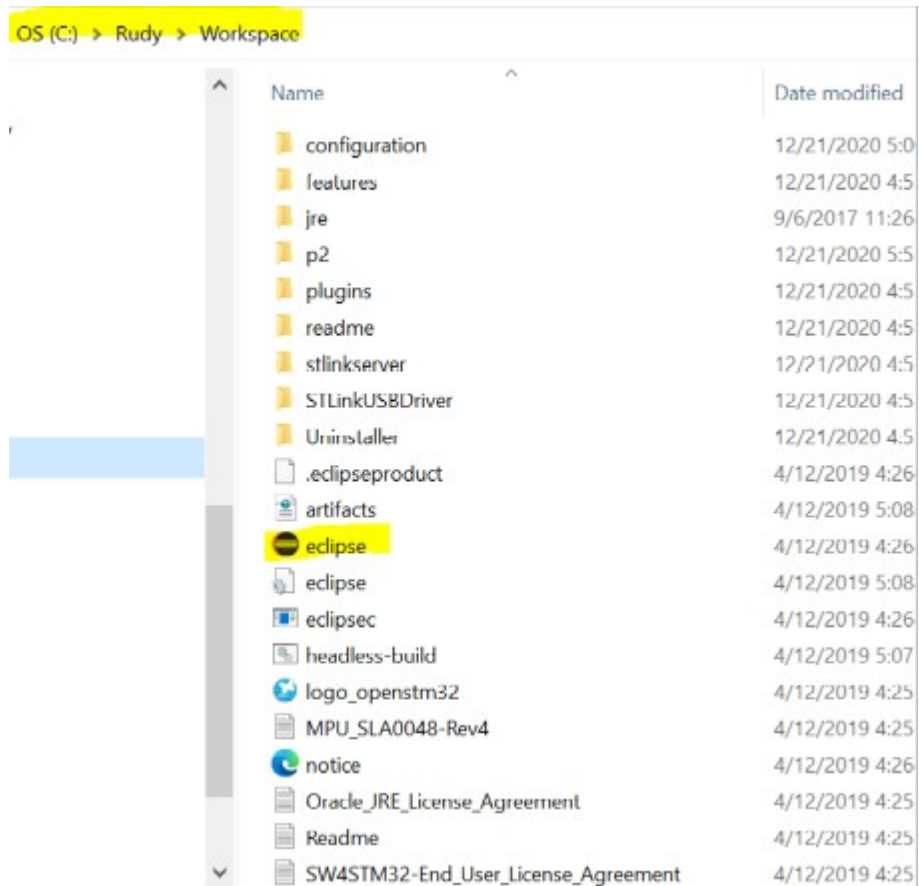



This method is using the STM32 System Workbench from ST.

1. Run the eclipse inside the directory where the STM32 System Workbench is installed. In my case, it was installed in C:\Rudy\Workspace

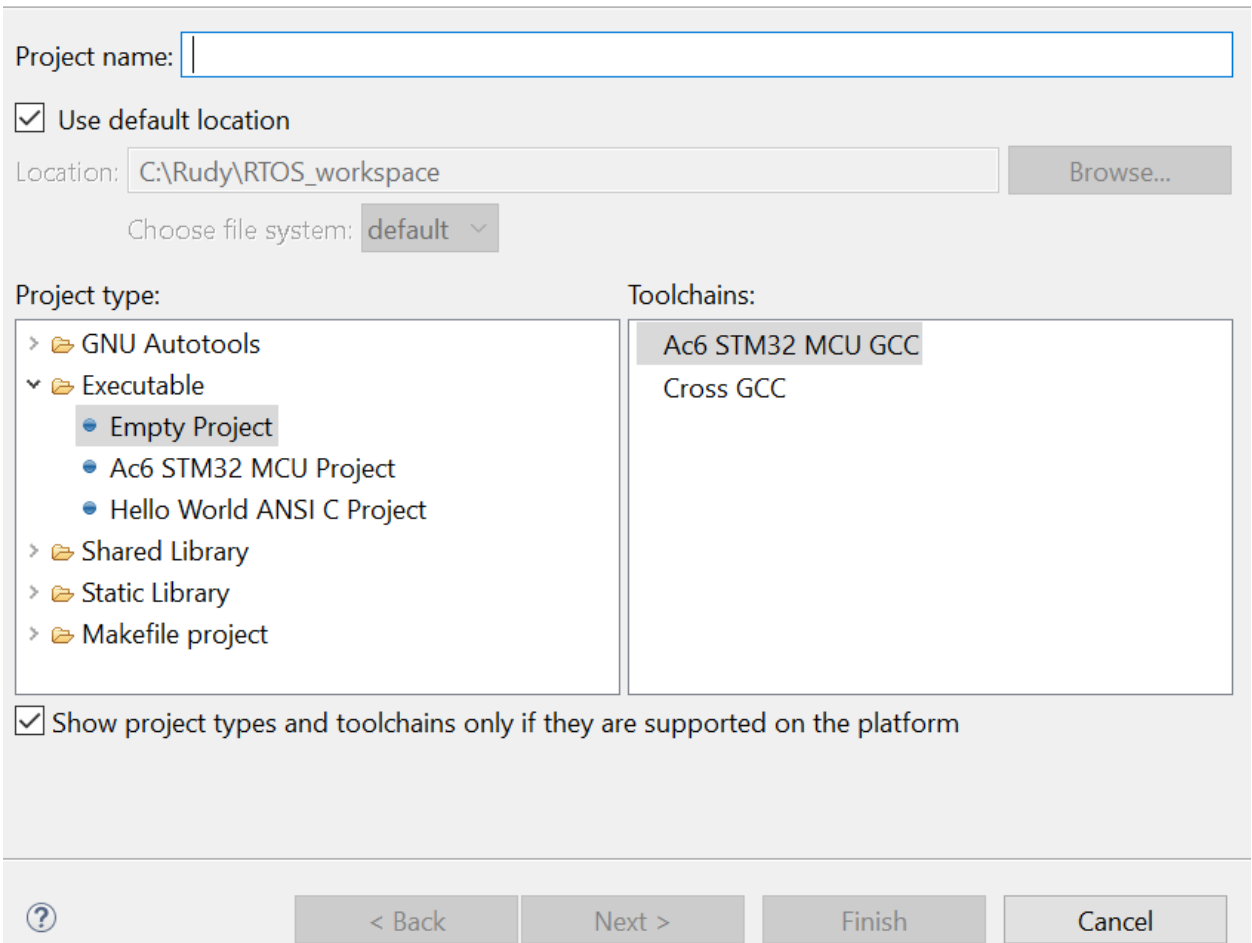


2. File -> New -> C Project

 C Project

C Project

Create C project of selected type



The dialog box for creating a C project. It includes a title bar with a close button, a subtitle 'C Project', and a description 'Create C project of selected type'. The main area contains a 'Project name' text box, a 'Use default location' checkbox, a 'Location' text box with the path 'C:\Rudy\RTOS_workspace' and a 'Browse...' button, a 'Choose file system' dropdown set to 'default', and two columns: 'Project type' and 'Toolchains'. The 'Project type' column shows a tree with 'GNU Autotools' expanded, containing 'Executable' (with sub-items 'Empty Project', 'Ac6 STM32 MCU Project', and 'Hello World ANSI C Project') and 'Shared Library', 'Static Library', and 'Makefile project'. The 'Toolchains' column lists 'Ac6 STM32 MCU GCC' and 'Cross GCC'. A checkbox at the bottom is checked, with the text 'Show project types and toolchains only if they are supported on the platform'. The footer has a help icon and four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

Project name:

☒ Use default location

Location:

Choose file system:

Project type:


- > GNU Autotools
 - ▼ Executable
 - Empty Project
 - Ac6 STM32 MCU Project
 - Hello World ANSI C Project
 - > Shared Library
 - > Static Library
 - > Makefile project


Toolchains:

- Ac6 STM32 MCU GCC
- Cross GCC

☒ Show project types and toolchains only if they are supported on the platform

3. Put "Project name", select "Location" and select "Executable" -> "Ac6 STM32 MCU Project".

 C Project — □ ×

C Project 

Create C project of selected type

Project name:

☒ Use default location

Location: Browse...

Choose file system: default ▾


Project type:	Toolchains:
<ul style="list-style-type: none">> GNU Autotools▼ Executable<ul style="list-style-type: none">• Empty Project• Ac6 STM32 MCU Project• Hello World ANSI C Project> Shared Library> Static Library> Makefile project	<div>Ac6 STM32 MCU GCC</div>

☒ Show project types and toolchains only if they are supported on the platform

? < Back Next > Finish Cancel

4. Click Next then one more Next

5. Select the board (e.g. STM32F4)

 C Project — □ ×

Target Configuration

Select either the mcu or the board target and configurations


Mcu Board

☐ Show ST Discovery boards ☐ Show ST EVAL boards
☒ Show ST NUCLEO boards ☐ Show custom boards


Series : STM32F4
Board : NUCLEO-F429ZI

Create a new custom board Remove this custom board

Mcu	STM32F429ZITx
Core	Arm Cortex-M4
Package	LQFP144
Memory 'RAM'	Size 0x30000 (@0x20000000)
Memory 'ROM'	Size 0x200000 (@0x8000000)


 < Back Next > Finish Cancel

6. Click Next and select “Standard Peripheral Library (StdPeriph)”
If Target firmware has not been found locally, simply click the “Download target firmware”
If there is an error, just finish the setup.

 C Project — □ ×

Project Firmware configuration


Select the project structure and firmware



☐ No firmware ☐ Don't generate startup files

☒ Standard Peripheral Library (StdPeriph)

☐ Hardware Abstraction Layer (Cube HAL)

 Target firmware has not been found locally, please install it !

[Download target firmware](#)

See '[Firmware Installation](#)' for settings related to firmware installation

☐ Extract all firmware in separate folder ⓘ

☐ Add low level drivers in the project

☒ As sources in the application project ⓘ

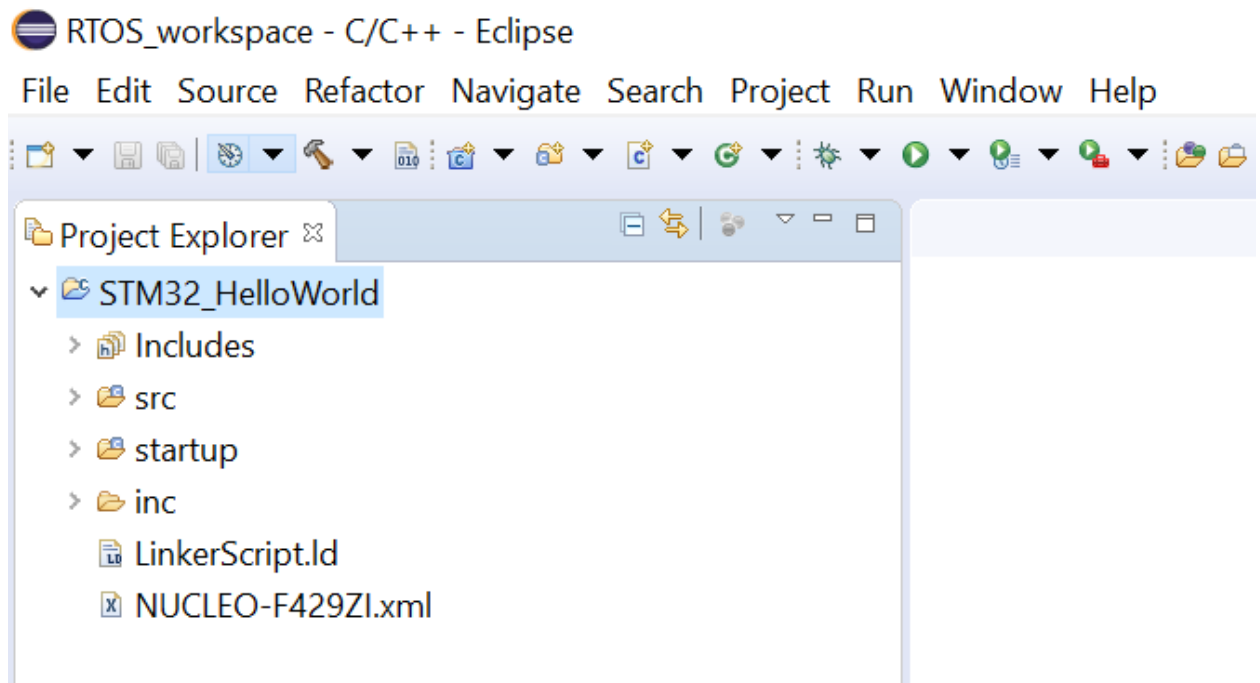
☐ As static external libraries ⓘ

Additional drivers

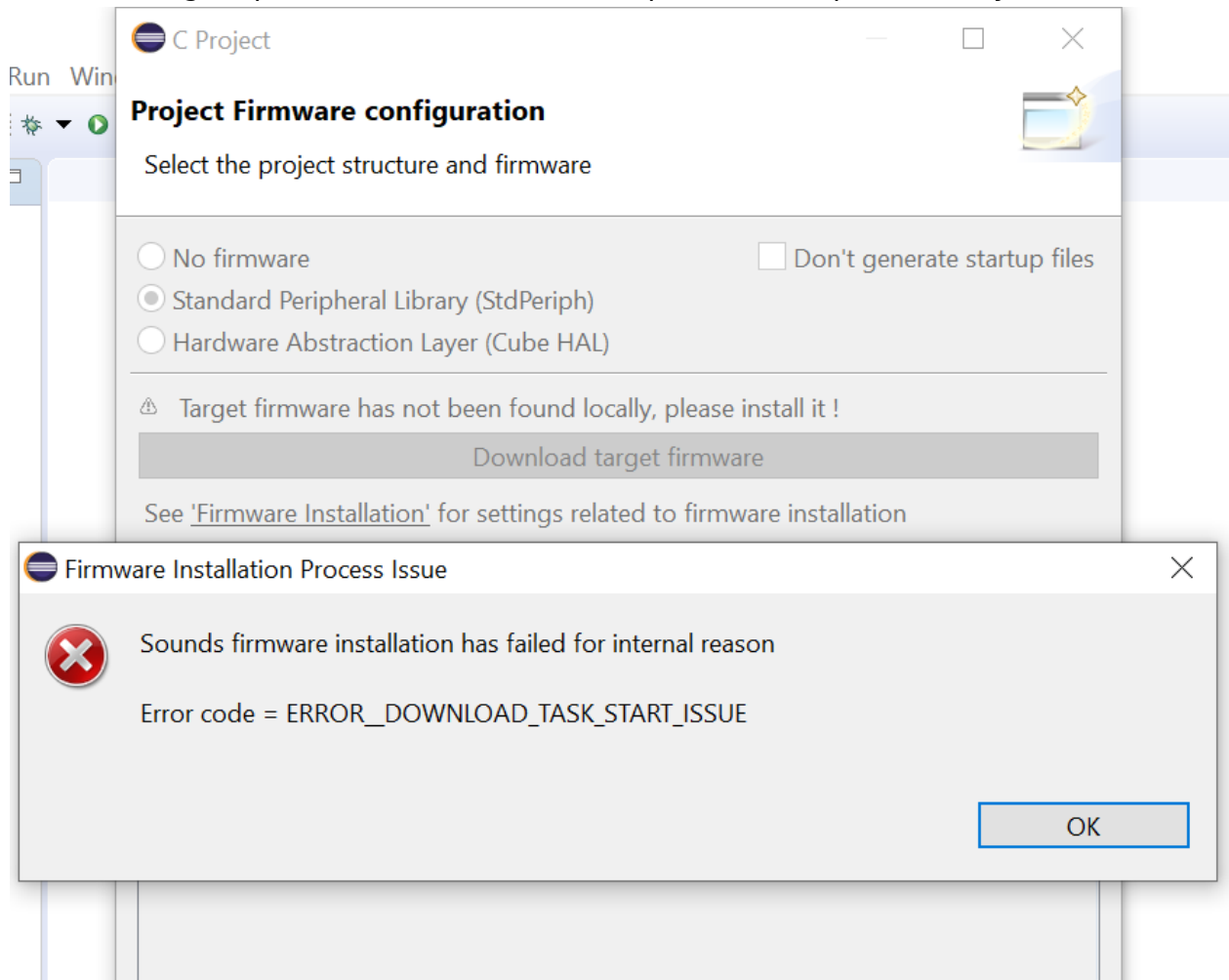
Additional utilities and third-party utilities:

⚠ You may have to make manual adjustments for third party utilities

7. Click Finish
The new C Project is now created.



8. If there is an error in step #5(as shown below), we need to integrate the StdPeriph library manually.
The following steps will show on how to setup the StdPeriph manually.

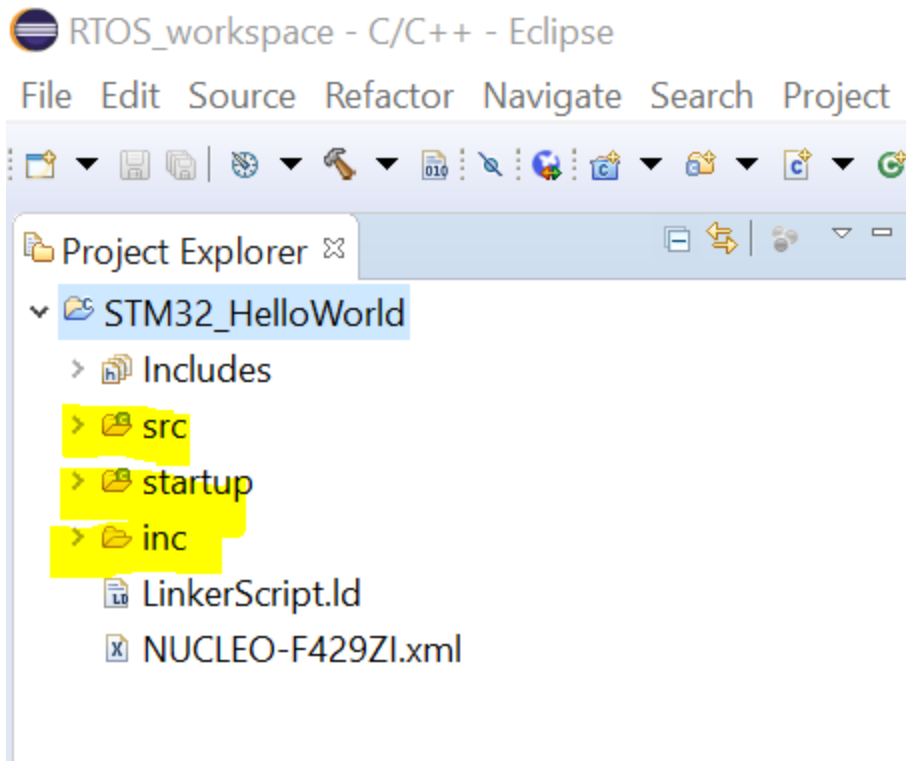


9. Use the attached standard peripheral library. The file is also available under STM32/Standard_Preripheral_Library folder.








StdPeri_fles.zip

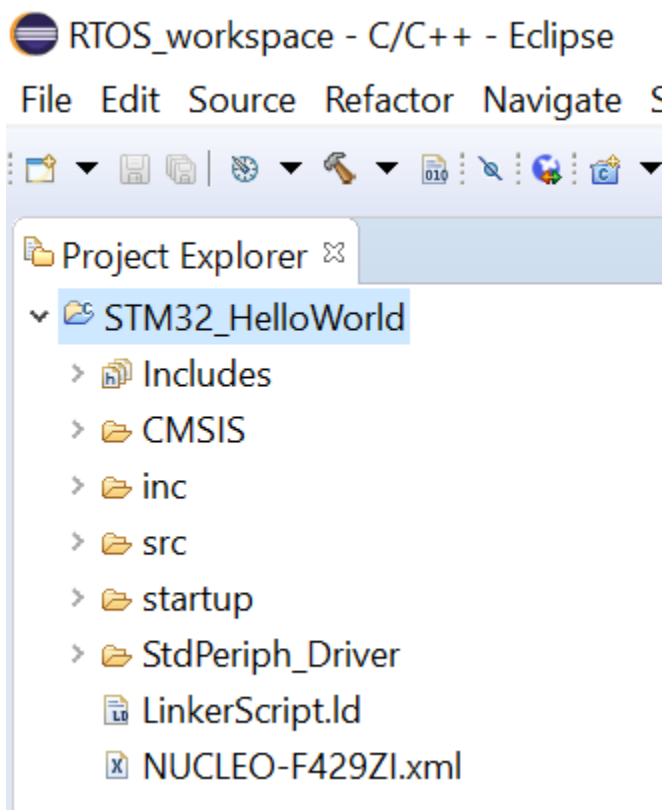
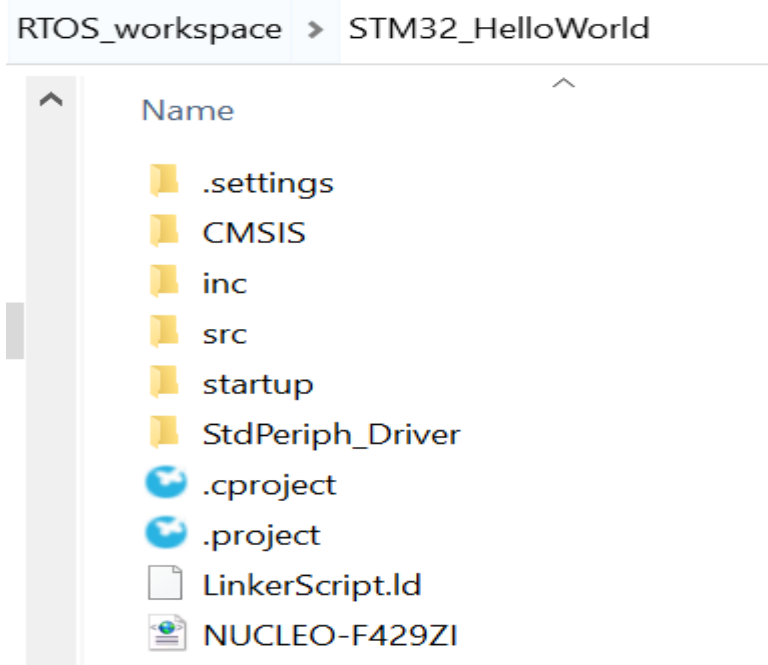
10. As we need to add the StdPeriph manually, we need to delete the folder “src”, “startup” and “inc” from our project. Right click and click “Delete”



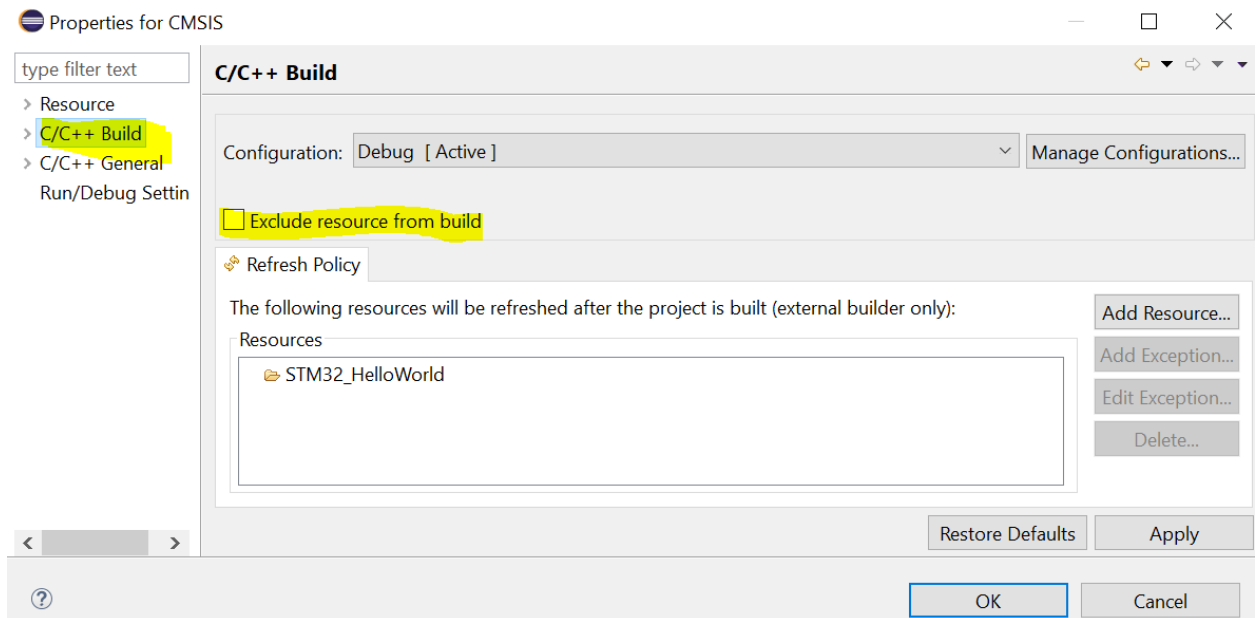
11. Extract the downloaded “StdPeri_files.zip” file. It contains the following files.

Name	Date modified
 CMSIS	12/22/2020 9:47 AM
 inc	12/22/2020 9:47 AM
 src	12/22/2020 9:47 AM
 startup	12/22/2020 9:47 AM
 StdPeriph_Driver	12/22/2020 9:47 AM

12. Select all the files shown in step #10 and copy into the project directory. You can also do drag and drop to copy the files. Go to the Eclipse and press F5 to refresh the project.

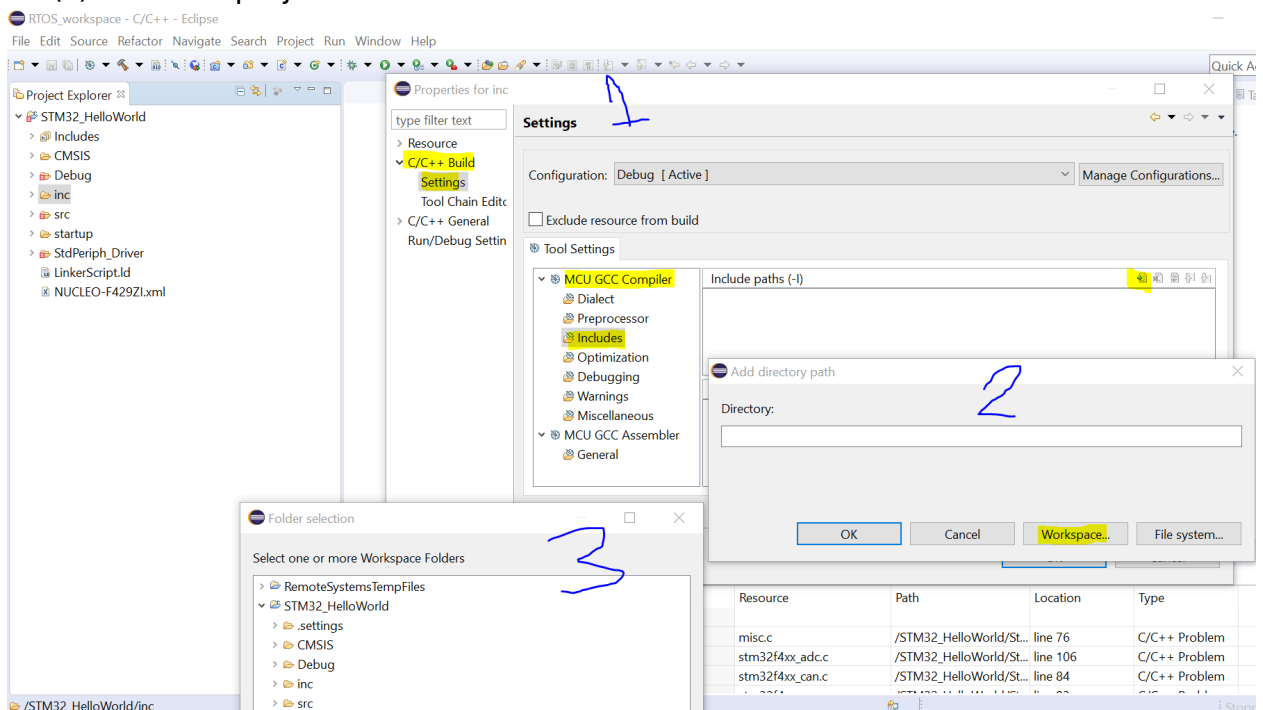


13. Right click each folders then “Properties” and make sure in the “C/C++ Build” -> “Exclude resource from build” is not checked.

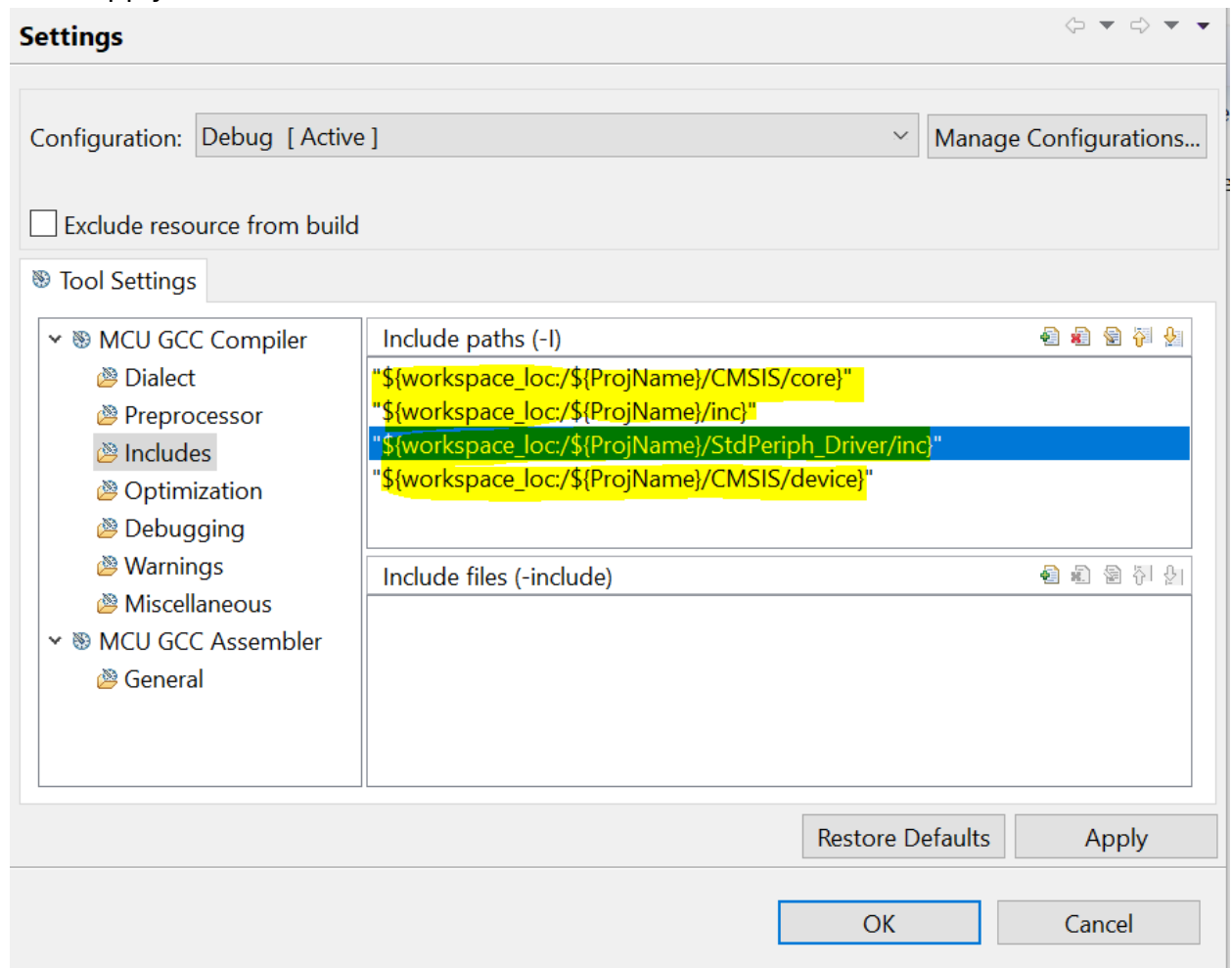


14. Then add the include files

Right Click on the Project -> Properties -> C/C++ Build -> Settings -> MCU GCC Compiler -> Includes -> Click the “+” icon to add the directory of the include files -> Click on Workspace -> Select the folder that contains include file(s) from the project.



Click Apply then OK.



15. Add the macro that suite to the microcontroller you are using. The definition are located in the following link: [CMSIS/device/stm32f4xx.h](#)

In my case, since I'm using STM32F49ZI, I'll select the macro

"STM32F429 439xx".

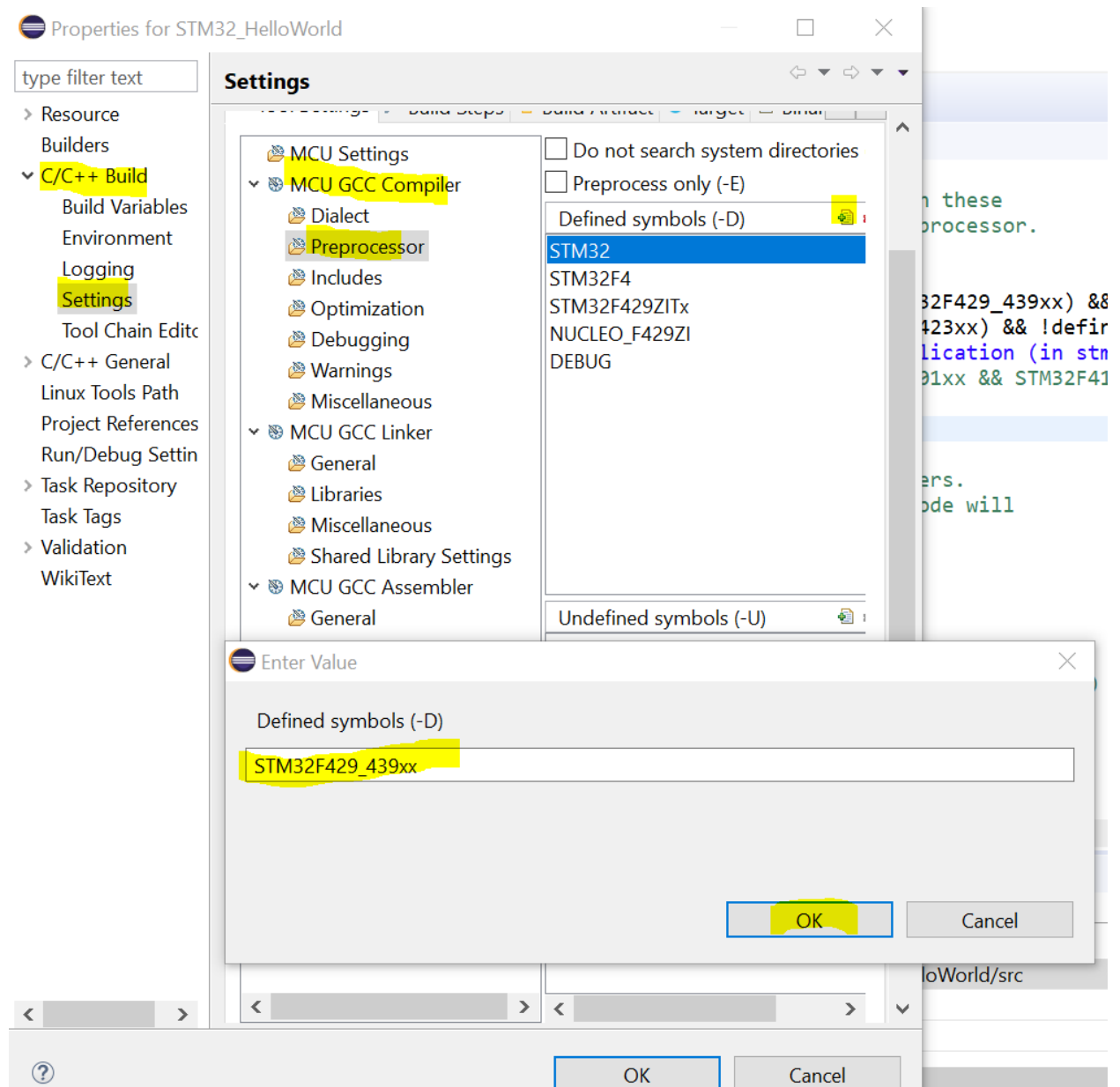
Also the macro "USE_STDPERIPH_DRIVER"

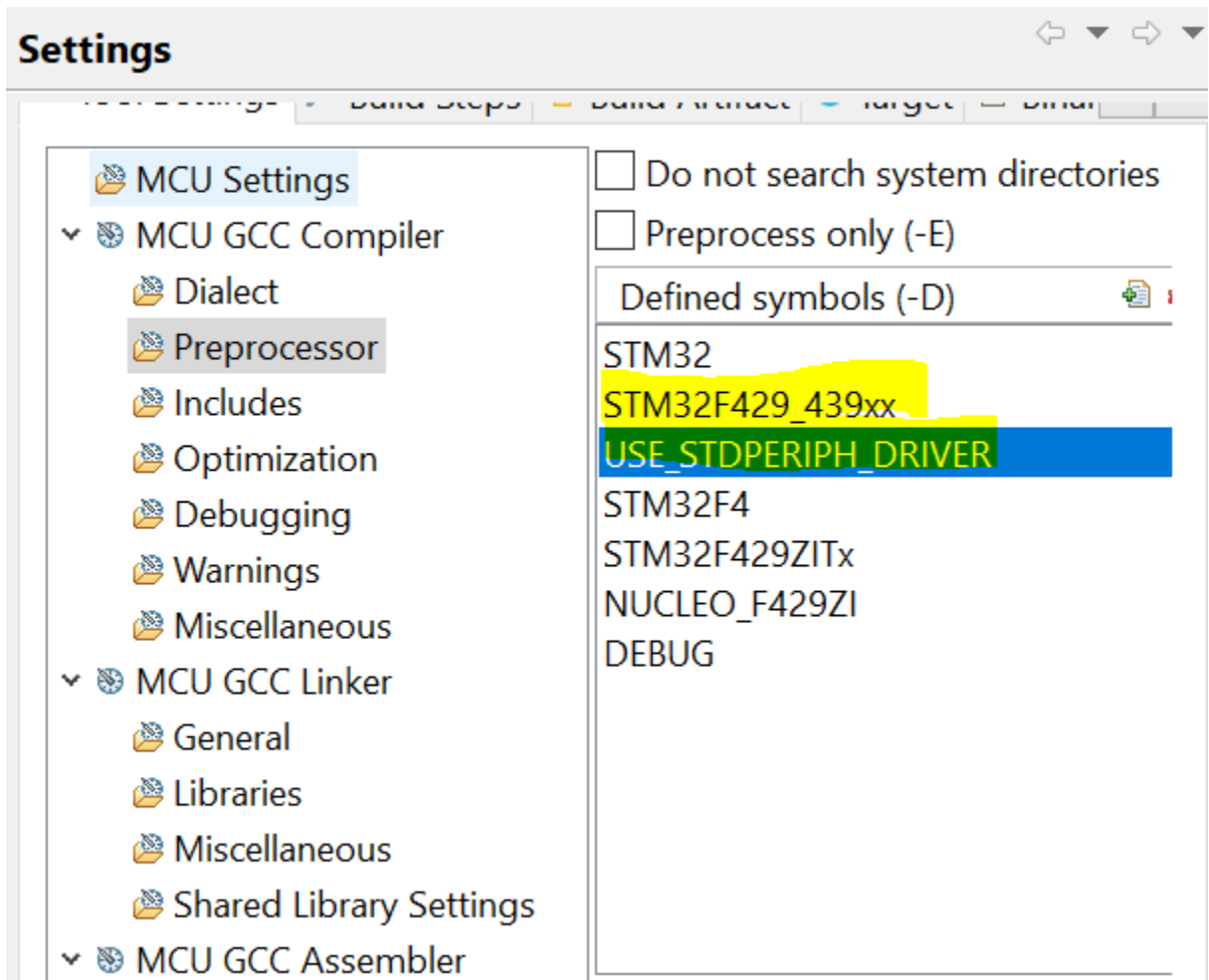
```

1 #if !defined(STM32F40_41xxx) && !defined(STM32F427_437xx) && !defined(STM32F429_439xx) && !defined(S
2 !defined(STM32F411xE) && !defined(STM32F412xG) && !defined(STM32F413_423xx) && !defined(STM32F4
3 /* #define STM32F40_41xxx */ /*!< STM32F405RG, STM32F405VG, STM32F405ZG, STM32F415RG, STM32F415V
4 STM32F407VG, STM32F407VE, STM32F407ZG, STM32F407ZE, STM32F407I
5 STM32F417VG, STM32F417VE, STM32F417ZG, STM32F417ZE, STM32F417I
6
7 /* #define STM32F427_437xx */ /*!< STM32F427VG, STM32F427VI, STM32F427ZG, STM32F427ZI, STM32F427I
8 STM32F437VG, STM32F437VI, STM32F437ZG, STM32F437ZI, STM32F437I
9
10 /* #define STM32F429_439xx */ /*!< STM32F429VG, STM32F429VI, STM32F429ZG, STM32F429ZI, STM32F429E
11 STM32F429NG, STM32F439NI, STM32F429IG, STM32F429II, STM32F439I
12 STM32F439ZG, STM32F439ZI, STM32F439BG, STM32F439BI, STM32F439I
13 STM32F439IG and STM32F439II Devices */
14
15 /* #define STM32F401xx */ /*!< STM32F401CB, STM32F401CC, STM32F401RB, STM32F401RC, STM32F401
16 STM32F401CD, STM32F401RD, STM32F401VD, STM32F401CExx, STM32F401

```

The screenshot shows the STM32CubeIDE Settings dialog, specifically the MCU GCC Compiler Preprocessor tab. The 'Defined symbols (-D)' list contains the following symbols: STM32, STM32F4, STM32F429ZITx, NUCLEO_F429ZI, and DEBUG. An 'Enter Value' dialog is open in the foreground, showing 'STM32F429_439xx' in the input field for the 'Defined symbols (-D)' list.





Now, the StdPeriph is successfully added.
Build the Project and you should be able to build it without any error. The binary file is also able to generate.

```

Problems Tasks Console Properties
CDT Build Console [STM32_HelloWorld]
Invoking: MCU GCC Linker
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -T"C:/Rudy/RTOS_workspace/STM32_HelloWorld/LinkerS
Finished building target: STM32_HelloWorld.elf

make --no-print-directory post-build
Generating binary and Printing size information:
arm-none-eabi-objcopy -O binary "STM32_HelloWorld.elf" "STM32_HelloWorld.bin"
arm-none-eabi-size "STM32_HelloWorld.elf"
text    data    bss     dec     hex filename
1400    1080    1092    3572    df4 STM32_HelloWorld.elf

10:22:18 Build Finished (took 1s.848ms)

```