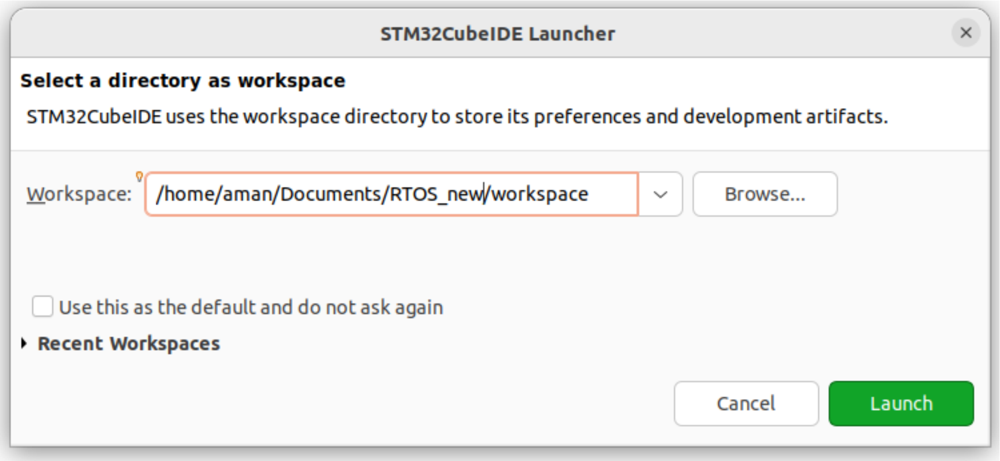
Steps for creating RTOS workspace in STM CubeMX IDE. -Aman Kanwar

Step 1. RTOS workspace creation for CubeMX



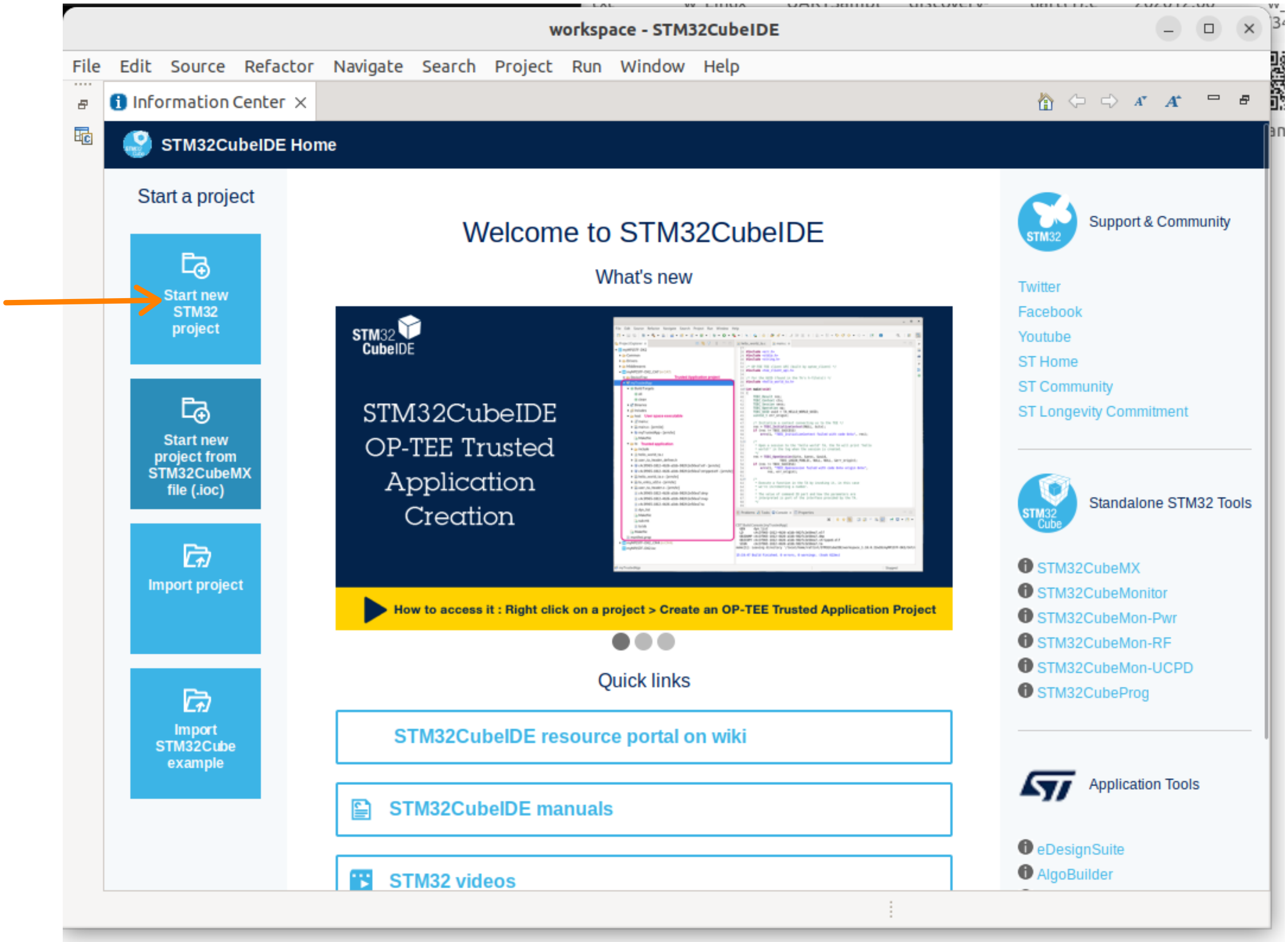
Select a directory as workspace

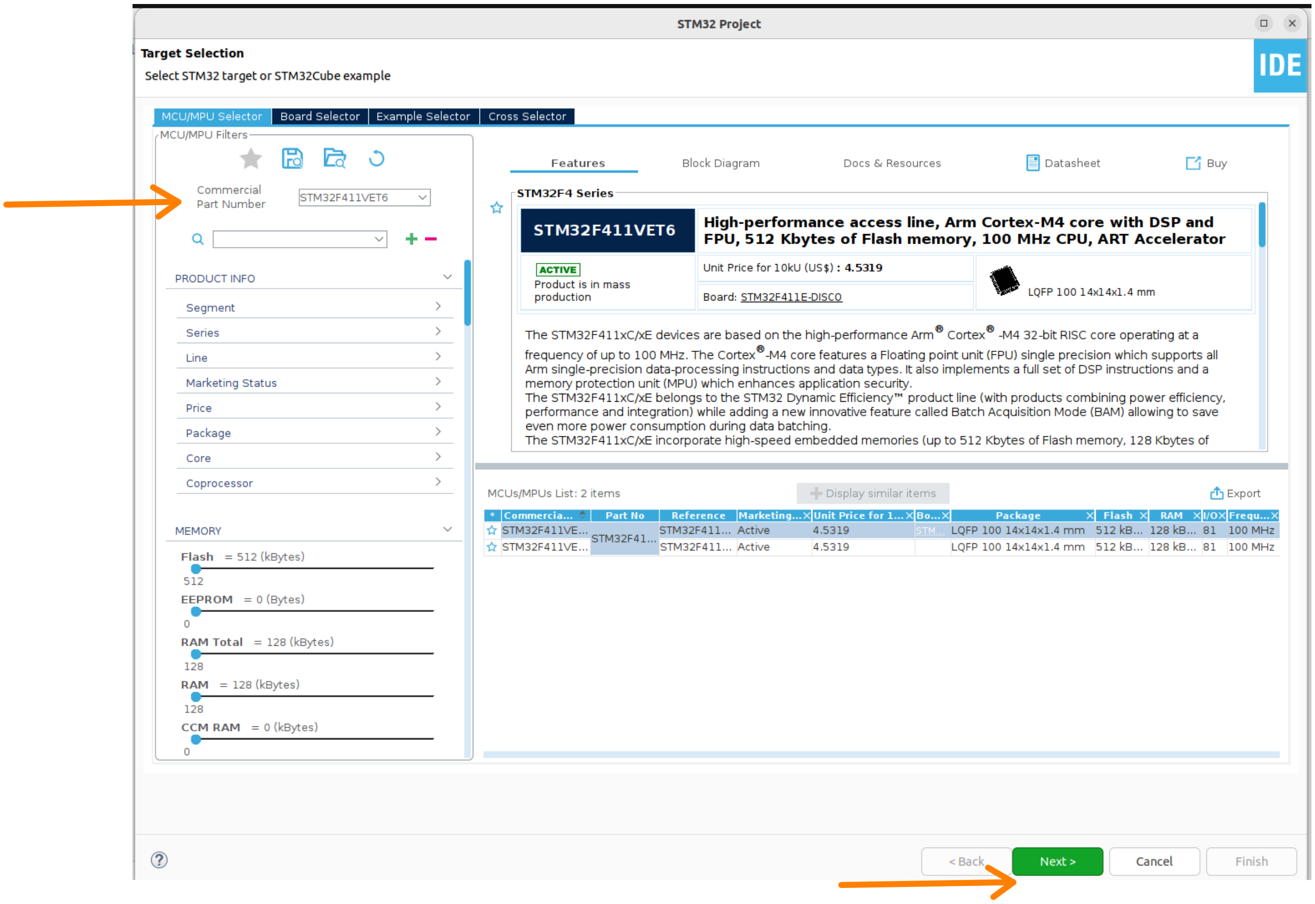
STM32CubeIDE uses the workspace directory to store its preferences and development artifacts.

Workspace: /home/aman/Documents/RTOS_new/workspace

☐ Use this as the default and do not ask again

Recent Workspaces





STM32 Project

Target Selection

Select STM32 target or STM32Cube example

Commercial Part Number: STM32F411VET6

Flash = 512 (kBytes)
512
EEPROM = 0 (Bytes)
0
RAM Total = 128 (kBytes)
128
RAM = 128 (kBytes)
128
CCM RAM = 0 (kBytes)
0

STM32F4 Series

STM32F411VET6

High-performance access line, Arm Cortex-M4 core with DSP and FPU, 512 Kbytes of Flash memory, 100 MHz CPU, ART Accelerator

ACTIVE
Product is in mass production

Unit Price for 10kU (US$) : **4.5319**

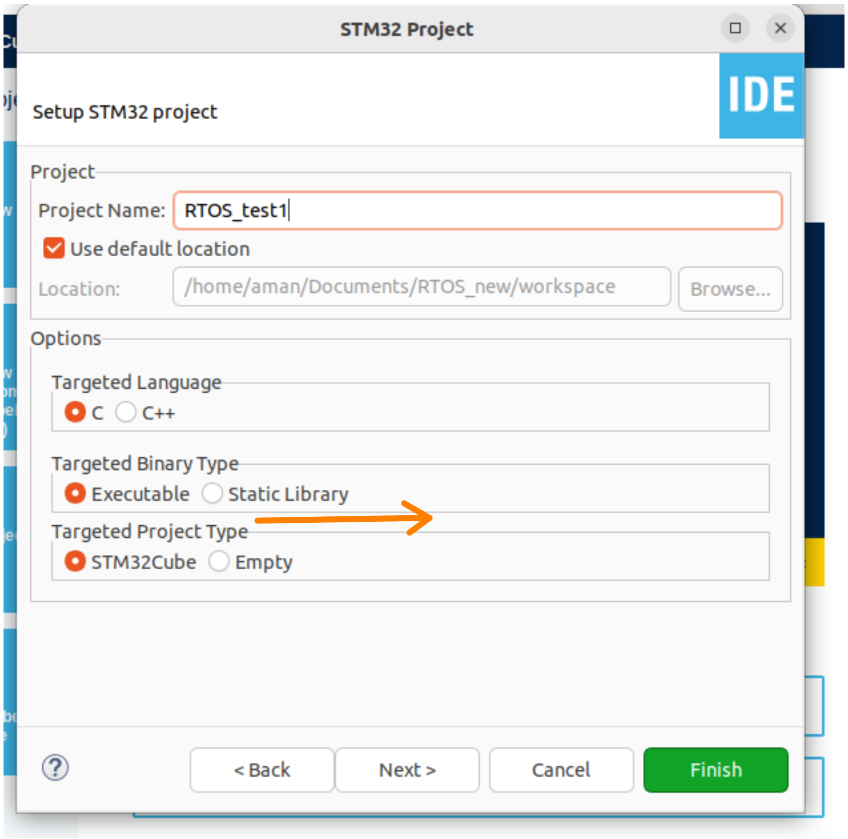
Board: STM32F411E-DISCO

LQFP 100 14x14x1.4 mm

The STM32F411xC/xE devices are based on the high-performance Arm® Cortex® -M4 32-bit RISC core operating at a frequency of up to 100 MHz. The Cortex®-M4 core features a Floating point unit (FPU) single precision which supports all Arm single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security.
The STM32F411xC/xE belongs to the STM32 Dynamic Efficiency™ product line (with products combining power efficiency, performance and integration) while adding a new innovative feature called Batch Acquisition Mode (BAM) allowing to save even more power consumption during data batching.
The STM32F411xC/xE incorporate high-speed embedded memories (up to 512 Kbytes of Flash memory, 128 Kbytes of

MCUs/MPUs List: 2 items

*	Commercia...	Part No	Reference	Marketing...	Unit Price for 1...	Bo...	Package	Flash	RAM	I/O	Frequ...
☆	STM32F411VE...	STM32F41...	STM32F411...	Active	4.5319	STM...	LQFP 100 14x14x1.4 mm	512 kB...	128 kB...	81	100 MHz
☆	STM32F411VE...		STM32F411...	Active	4.5319		LQFP 100 14x14x1.4 mm	512 kB...	128 kB...	81	100 MHz



STM32 Project

Setup STM32 project

Project

Project Name: RTOS_test1

☑ Use default location

Location: /home/aman/Documents/RTOS_new/workspace

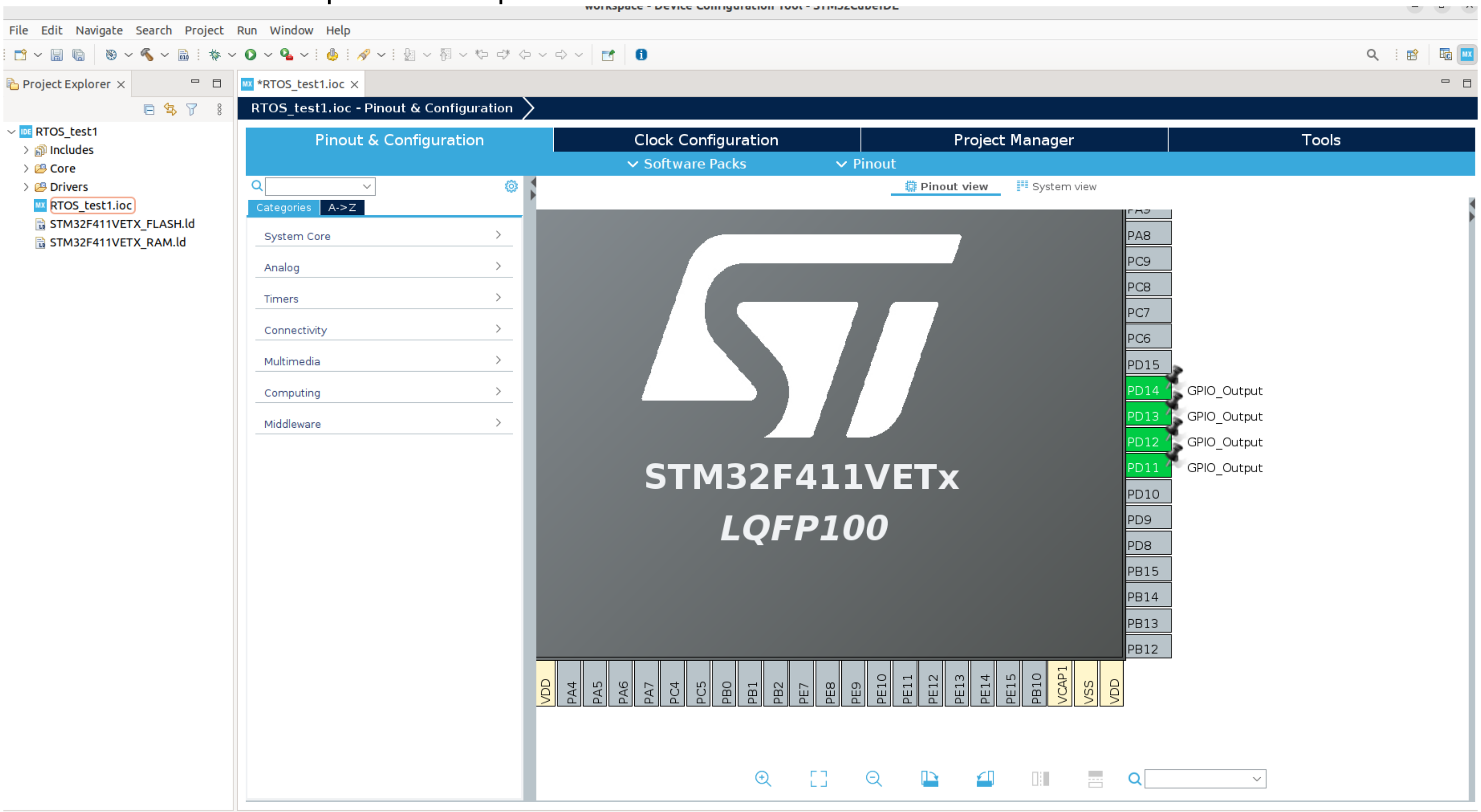
Options

Targeted Language
● C ○ C++

Targeted Binary Type
● Executable ○ Static Library

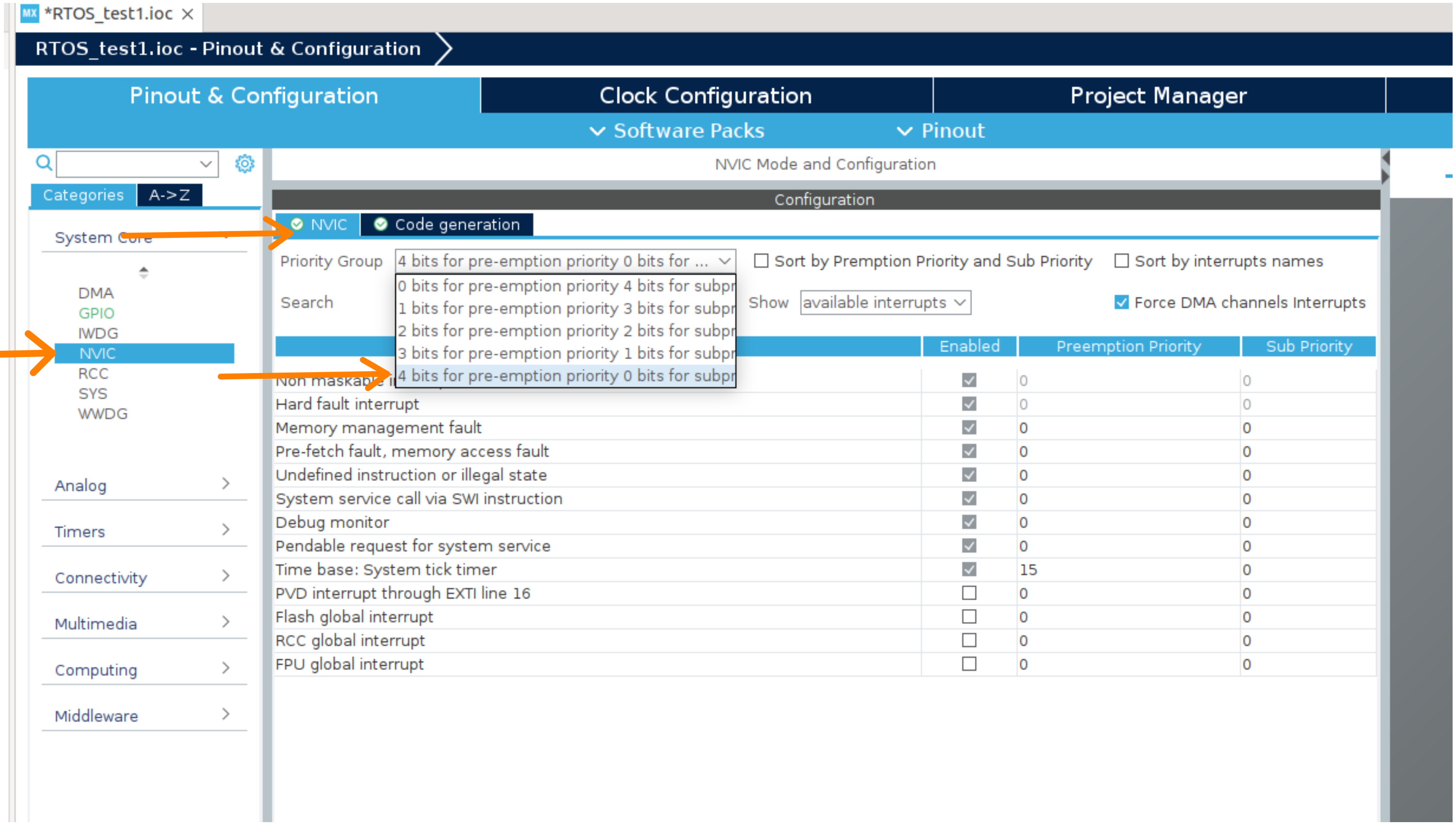
Targeted Project Type
● STM32Cube ○ Empty

Set the LED pins as outpuse based on the board used



Setting up the workspace for FreeRTOS will require setting up of NVIC and SysTick settings, since FreeRTOS needs a time base source for the SysTick tick counting.

NVIC priority group setting for 4 bit preemption and 0 bit for sub-priority (RTOS critical)



Priority Group: 4 bits for pre-emption priority 0 bits for ...

- 0 bits for pre-emption priority 4 bits for subpr
- 1 bits for pre-emption priority 3 bits for subpr
- 2 bits for pre-emption priority 2 bits for subpr
- 3 bits for pre-emption priority 1 bits for subpr
- 4 bits for pre-emption priority 0 bits for subpr

□ Sort by Premption Priority and Sub Priority
□ Sort by interrupts names

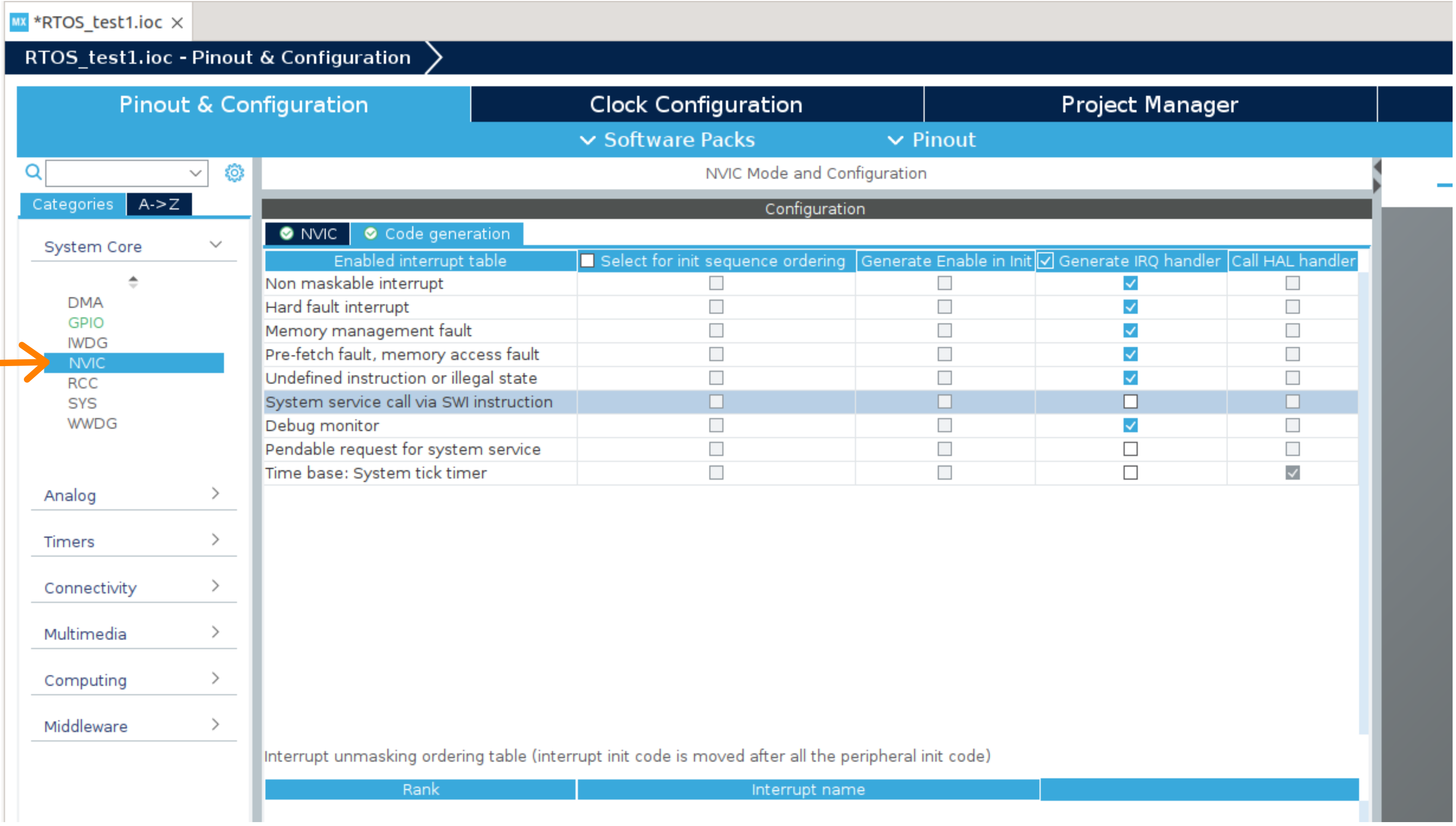
Search

Show: available interrupts

☑ Force DMA channels Interrupts

	Enabled	Preemption Priority	Sub Priority
Non maskable i	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	15	0
PVD interrupt through EXTI line 16	□	0	0
Flash global interrupt	□	0	0
RCC global interrupt	□	0	0
FPU global interrupt	□	0	0

Un-check of following handlers will be needed as highlighted. Since, same interrupt handler are defined by the FreeRTOS kernel and could cause duplication conflict while building.



NVIC Mode and Configuration

Configuration

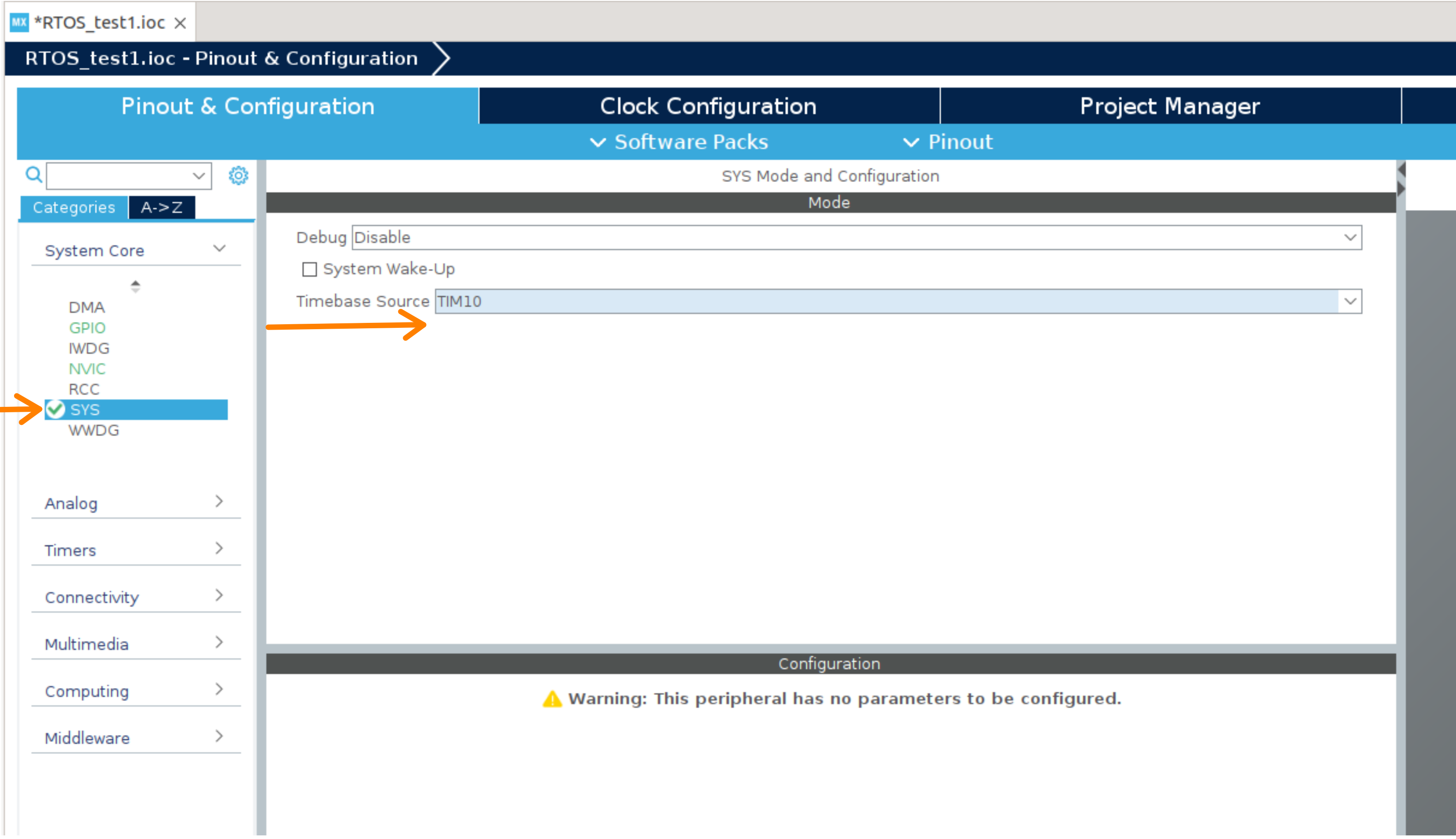
NVIC | Code generation

Enabled interrupt table	□ Select for init sequence ordering	Generate Enable in Init	☑ Generate IRQ handler	Call HAL handler
Non maskable interrupt	□	□	☑	□
Hard fault interrupt	□	□	☑	□
Memory management fault	□	□	☑	□
Pre-fetch fault, memory access fault	□	□	☑	□
Undefined instruction or illegal state	□	□	☑	□
System service call via SWI instruction	□	□	□	□
Debug monitor	□	□	☑	□
Pendable request for system service	□	□	□	□
Time base: System tick timer	□	□	□	☑

Interrupt unmasking ordering table (interrupt init code is moved after all the peripheral init code)

Rank	Interrupt name	

Now, the HAL driver configuration uses SysTick for its time base source. FreeRTOS also uses the same SysTick for the time base source. To resolve this conflict, we can move the time base source for HAL library as one of the timer.



SYS Mode and Configuration

Mode

Debug Disable

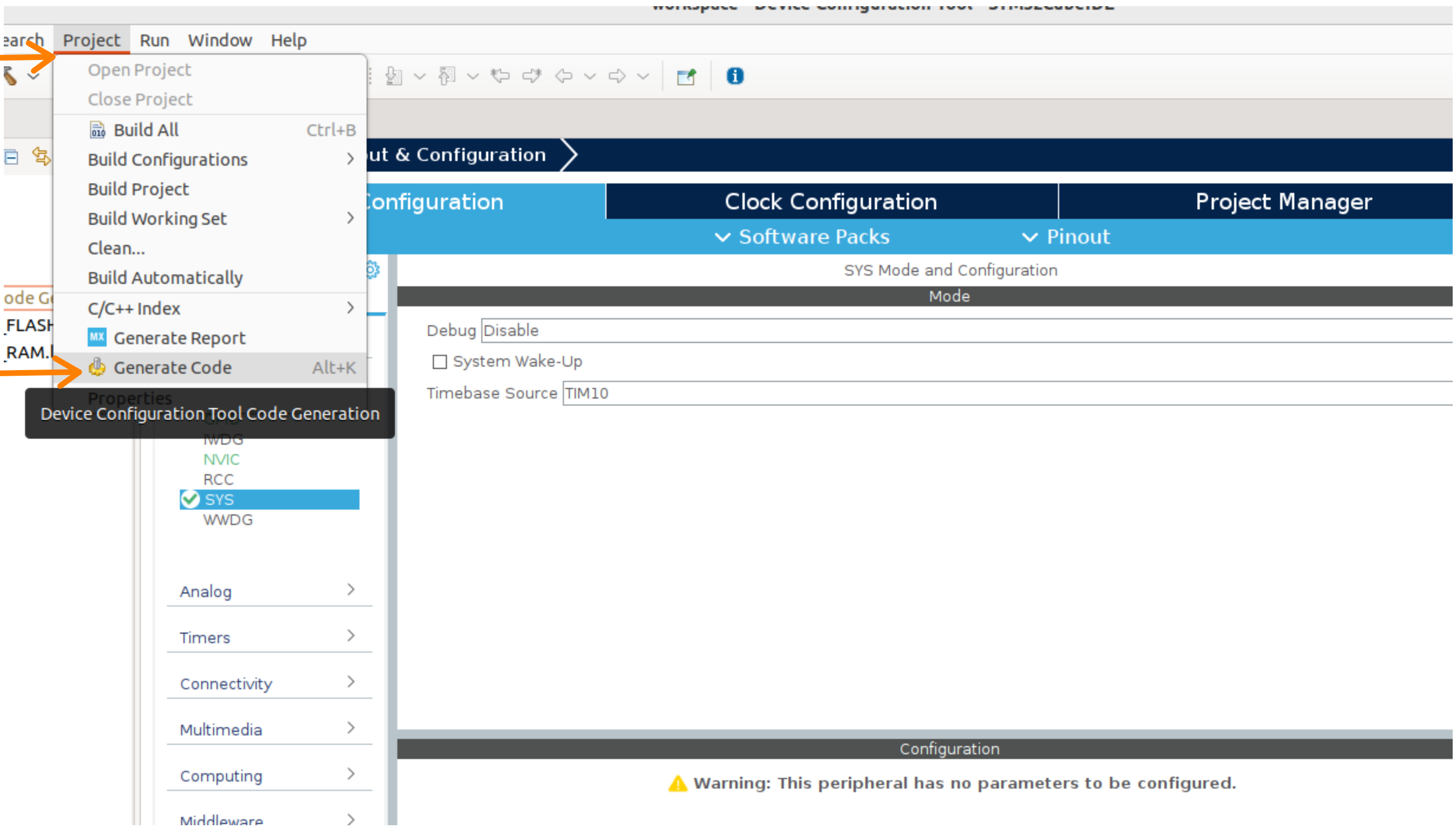
□ System Wake-Up

Timebase Source TIM10

Configuration

⚠ Warning: This peripheral has no parameters to be configured.

Once done, we can generate the Code for our configuration, same will open the C/C++ perspective.



SYS Mode and Configuration

Mode

Debug Disable

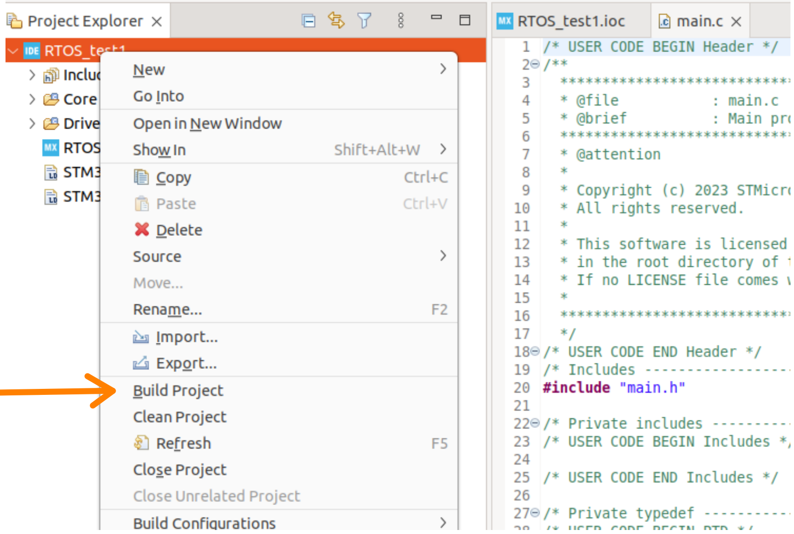
□ System Wake-Up

Timebase Source TIM10

Configuration

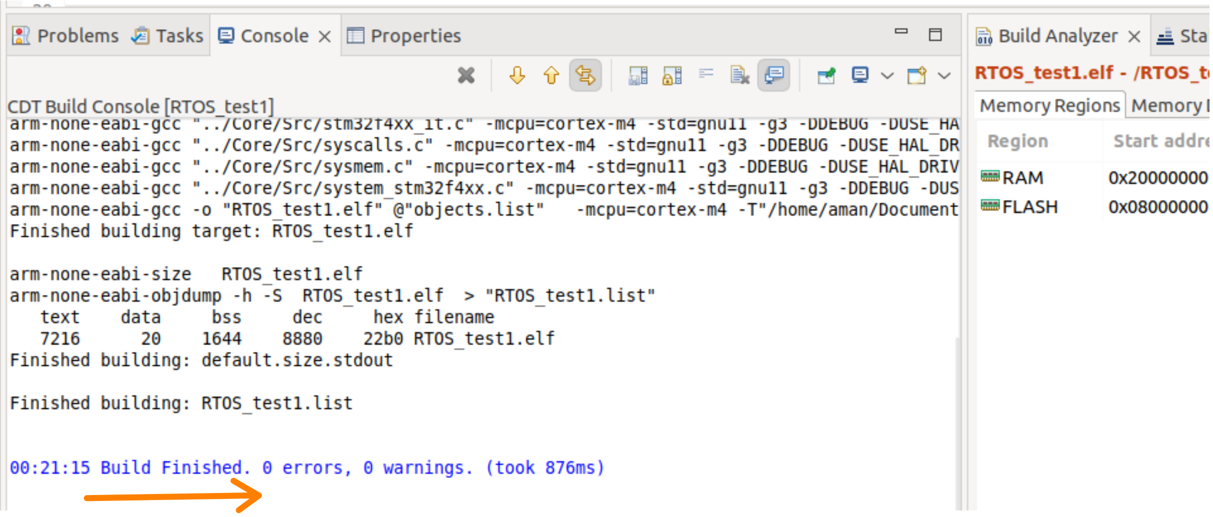
⚠ Warning: This peripheral has no parameters to be configured.

Build Project



```
/* USER CODE BEGIN Header */
/**
  ******************************
  * @file           : main.c
  * @brief          : Main pro
  ******************************
  * @attention
  *
  * Copyright (c) 2023 STMicro
  * All rights reserved.
  *
  * This software is licensed
  * in the root directory of t
  * If no LICENSE file comes w
  *
  ******************************
  */
/* USER CODE END Header */
/* Includes ------------------
#include "main.h"

/* Private includes ---------
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef ----------
```

0 Errors post build



CDT Build Console [RTOS_test1]

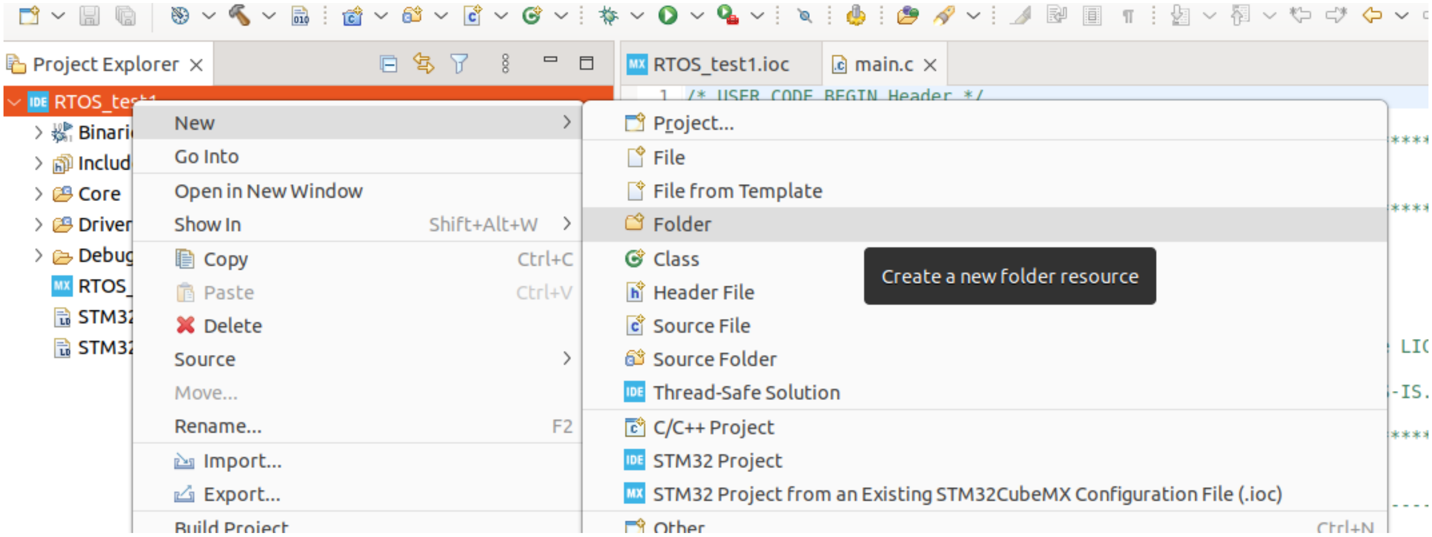
```
arm-none-eabi-gcc "../Core/Src/stm32f4xx_it.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE_HA
arm-none-eabi-gcc "../Core/Src/syscalls.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE_HAL_DR
arm-none-eabi-gcc "../Core/Src/sysmem.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE_HAL_DRIV
arm-none-eabi-gcc "../Core/Src/system_stm32f4xx.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUS
arm-none-eabi-gcc -o "RTOS_test1.elf" @"objects.list"   -mcpu=cortex-m4 -T"/home/aman/Document
Finished building target: RTOS_test1.elf

arm-none-eabi-size   RTOS_test1.elf
arm-none-eabi-objdump -h -S  RTOS_test1.elf  > "RTOS_test1.list"
   text    data     bss     dec     hex filename
   7216      20    1644    8880    22b0 RTOS_test1.elf
Finished building: default.size.stdout

Finished building: RTOS_test1.list


00:21:15 Build Finished. 0 errors, 0 warnings. (took 876ms)
```

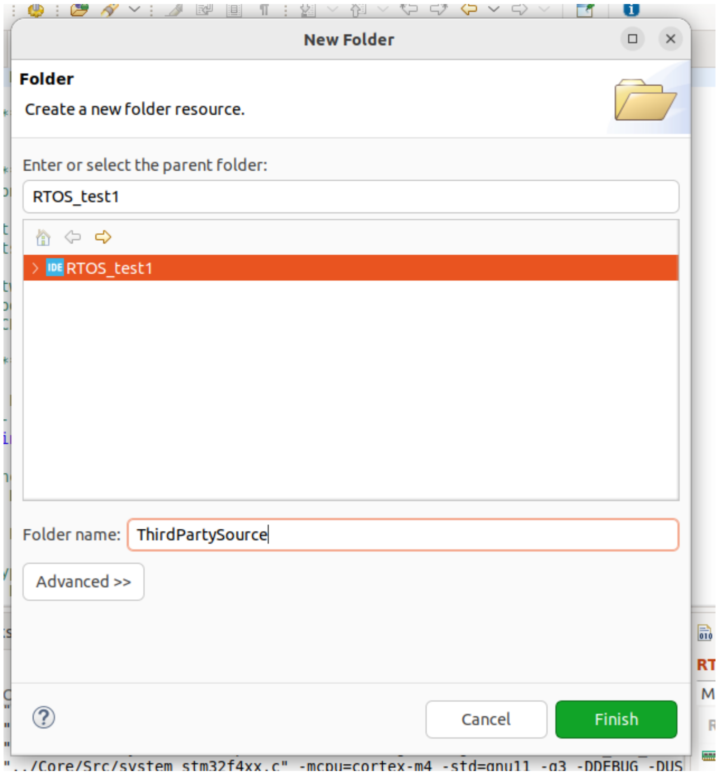
RTOS_test1.elf - /RTOS_t

Region	Start addre
RAM	0x20000000
FLASH	0x08000000

Need to import RTOS kernel source files to the project.

Create a folder "ThirdPartySource" under project





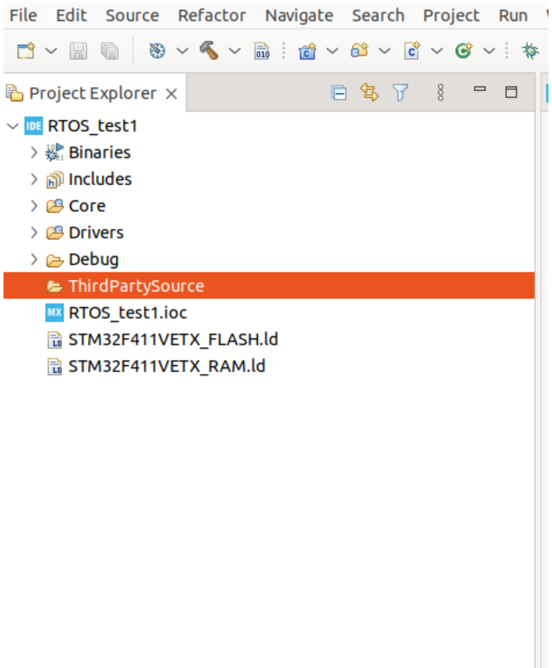
Folder

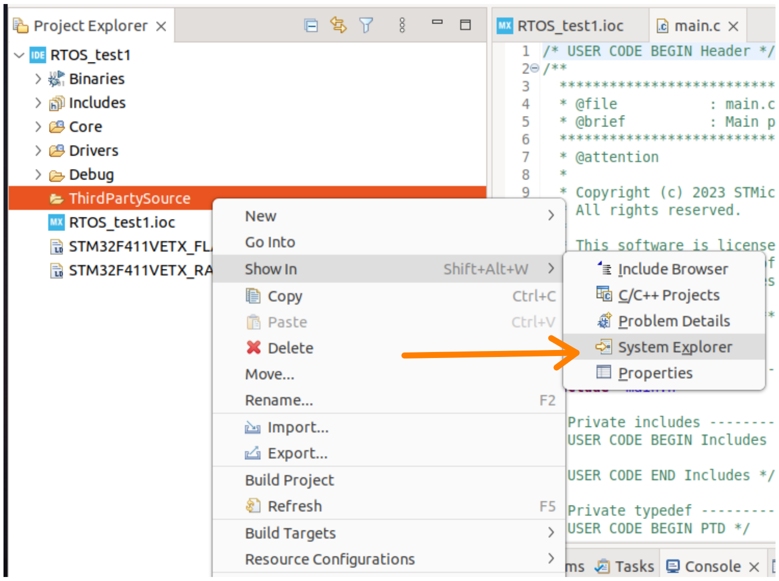
Create a new folder resource.

Enter or select the parent folder: RTOS_test1

Folder name: ThirdPartySource

```
"../Core/Src/system_stm32f4xx.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUS
```





```
/* USER CODE BEGIN Header */
/**
  *************************
  * @file           : main.c
  * @brief          : Main p
  *************************
  * @attention
  *
  * Copyright (c) 2023 STMic
  * All rights reserved.

  This software is license

  Private includes --------
  USER CODE BEGIN Includes

  USER CODE END Includes */

  Private typedef ---------
  USER CODE BEGIN PTD */
```

Open System Explorer for the folder ThirdPartySource.

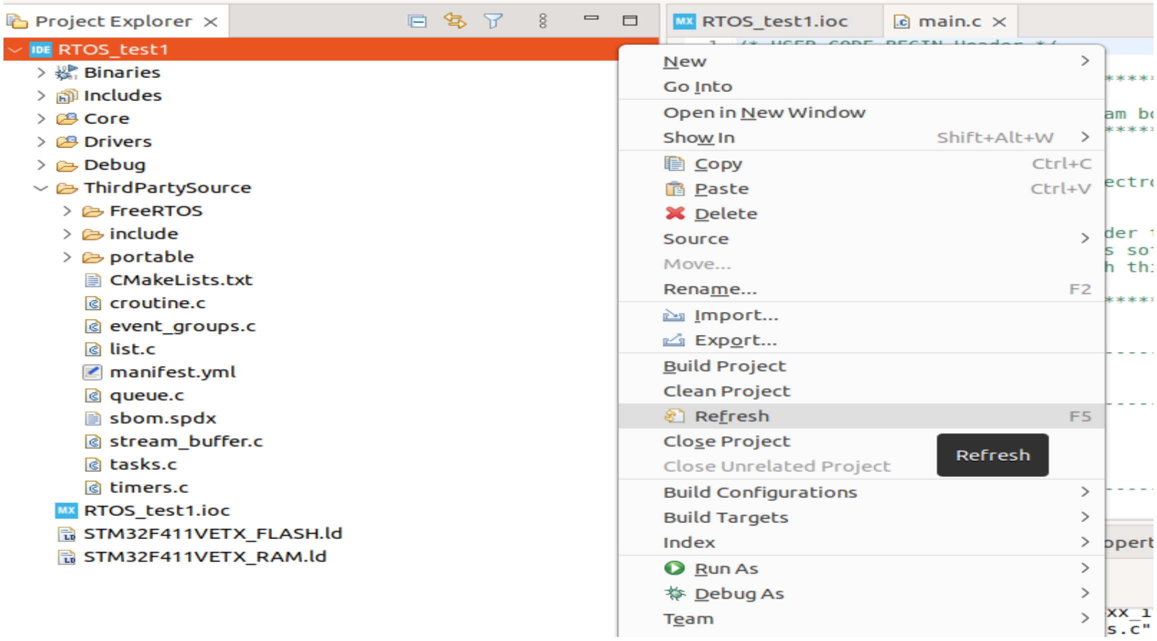
We need to copy RTOS kernel source file in this folder.

So the Kernel files can becoma a part of Build Project.

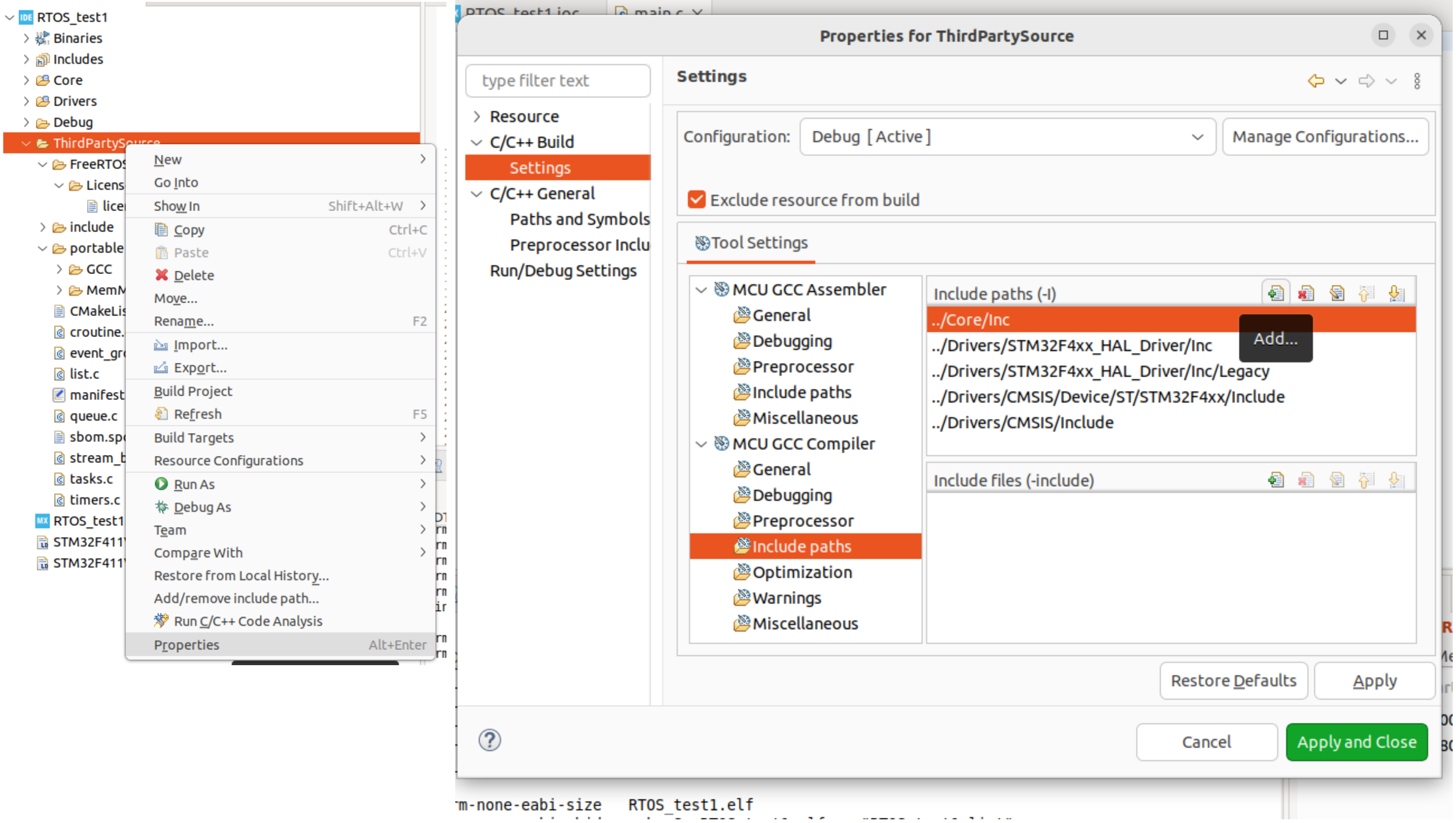
Now, need to Download the RTOS Kernel Source files from following location.

https://github.com/amankanwar/FreeRTOS/raw/main/FreeRTOS-KernelV10.5.1.zip

Add the mentioned directory to the project and refresh project.



Project Include Path settings for added files.



Configuration: Debug [Active]

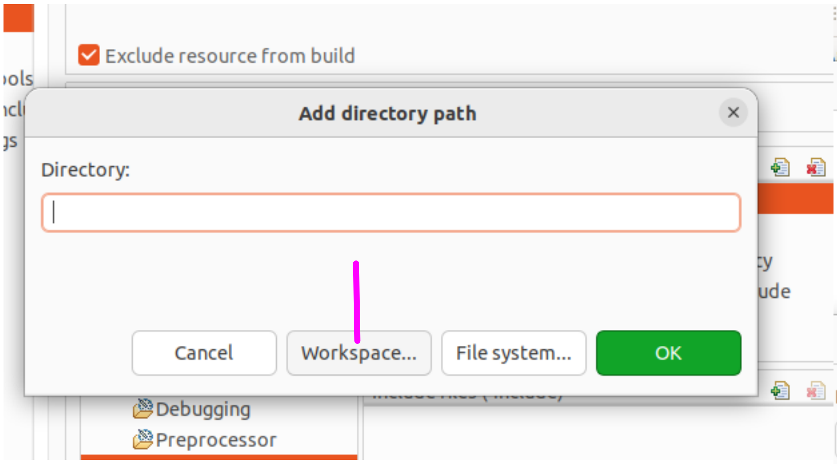
☑ Exclude resource from build

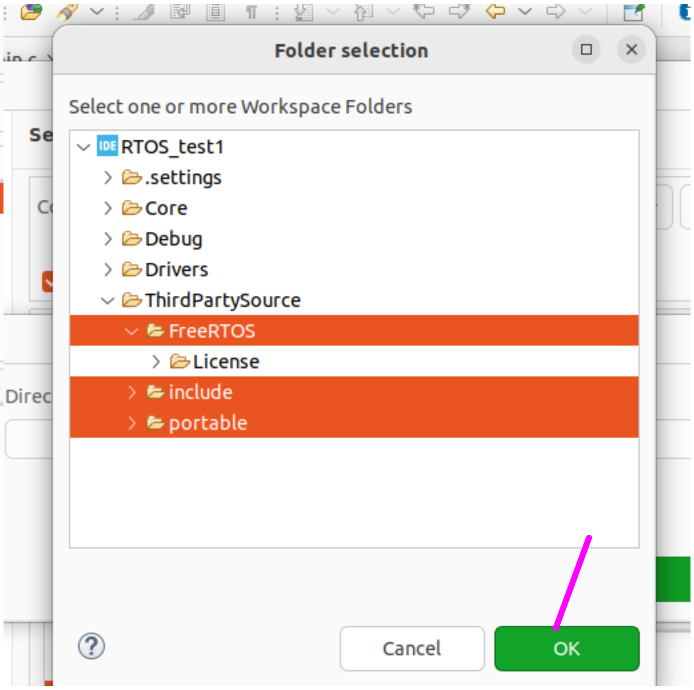
Include paths (-I)

- ../Core/Inc
- ../Drivers/STM32F4xx_HAL_Driver/Inc
- ../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy
- ../Drivers/CMSIS/Device/ST/STM32F4xx/Include
- ../Drivers/CMSIS/Include

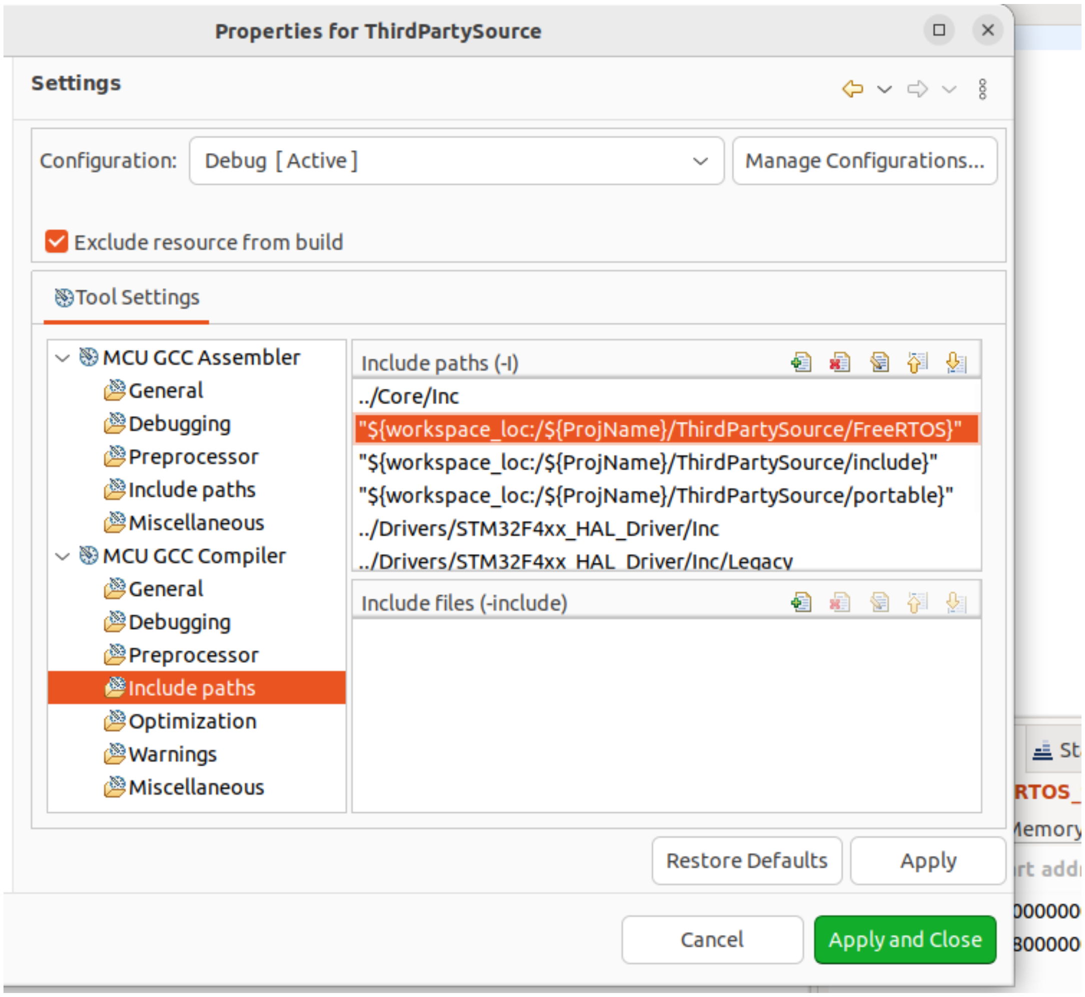
Include files (-include)

```
rm-none-eabi-size    RTOS_test1.elf
```



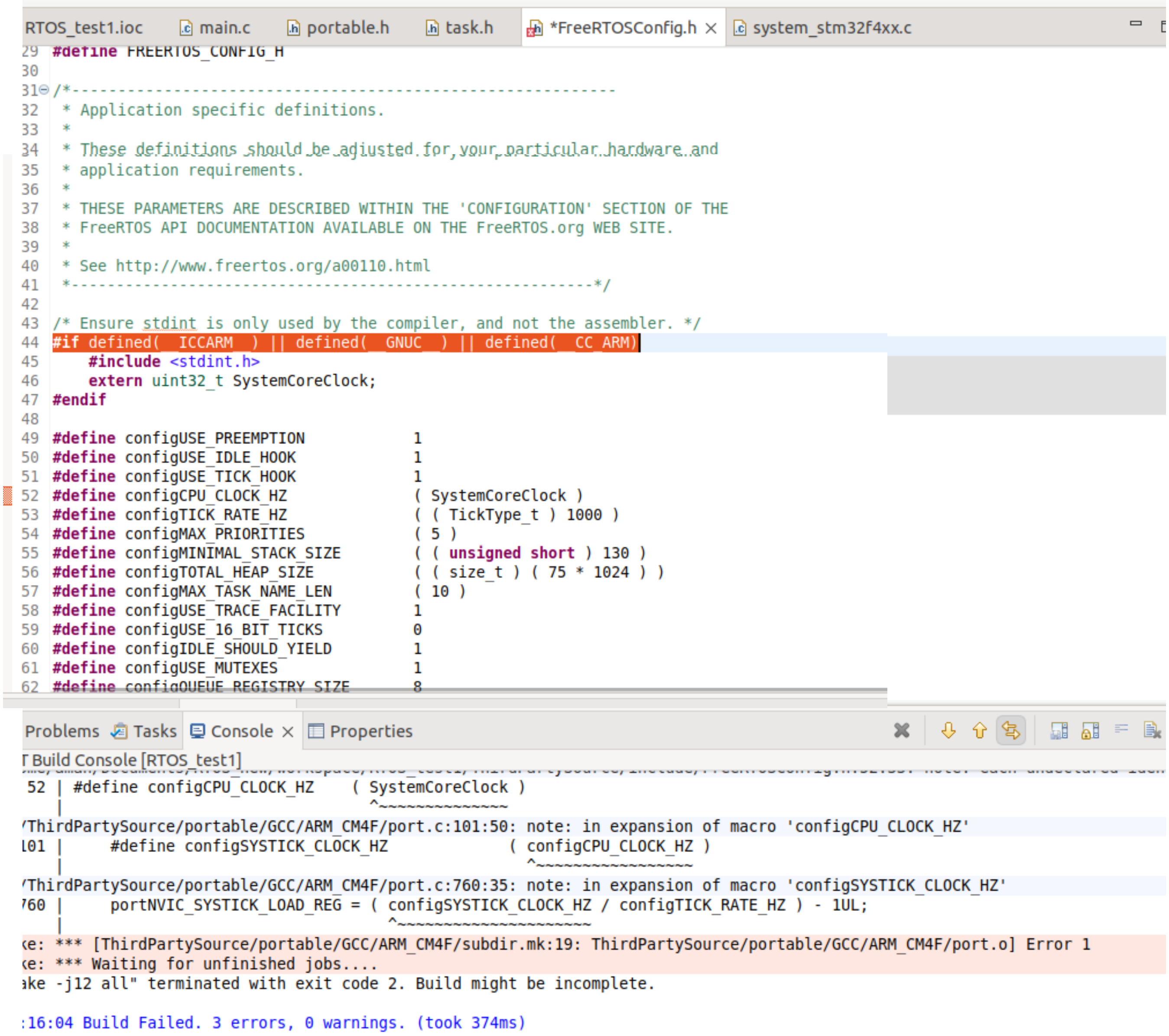


Apply and Close



```
../Core/Inc
"${workspace_loc:/${ProjName}/ThirdPartySource/FreeRTOS}"
"${workspace_loc:/${ProjName}/ThirdPartySource/include}"
"${workspace_loc:/${ProjName}/ThirdPartySource/portable}"
../Drivers/STM32F4xx_HAL_Driver/Inc
../Drivers/STM32F4xx_HAL_Driver/Inc/Legacy
```

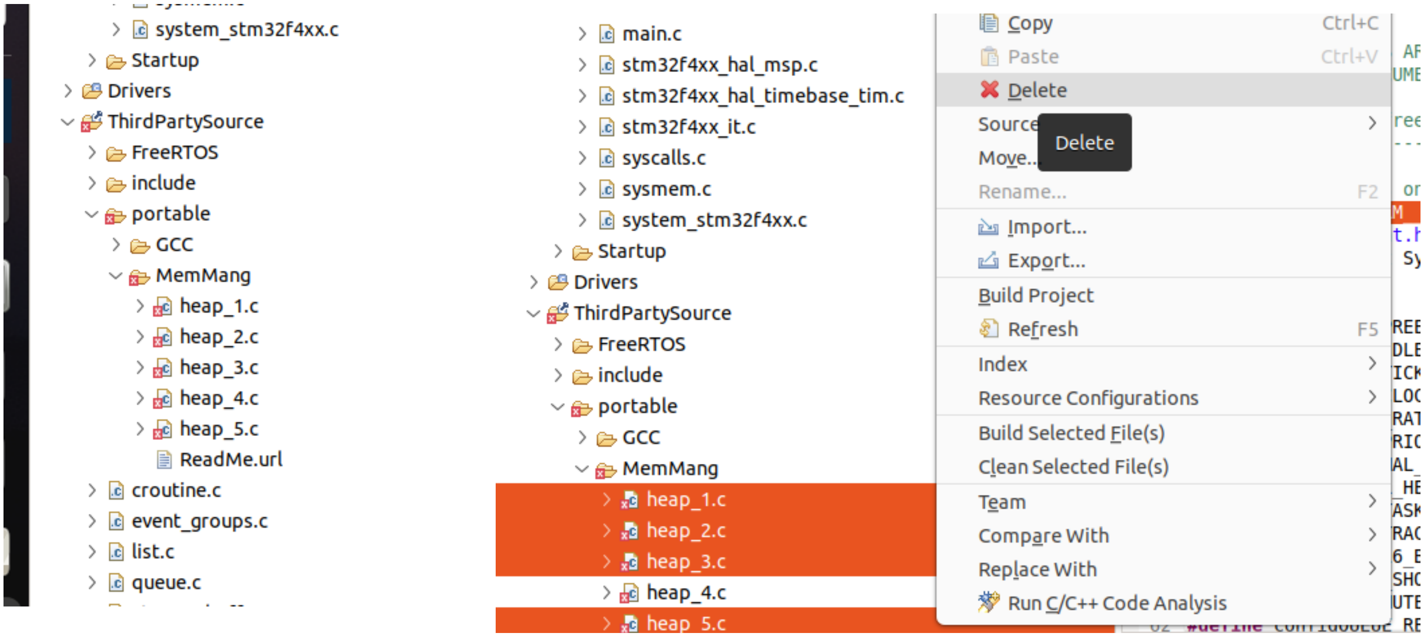
Add compiler information in FreeRTOSconfig.h from where the system core clock is getting exported out for the other driver files to be used.



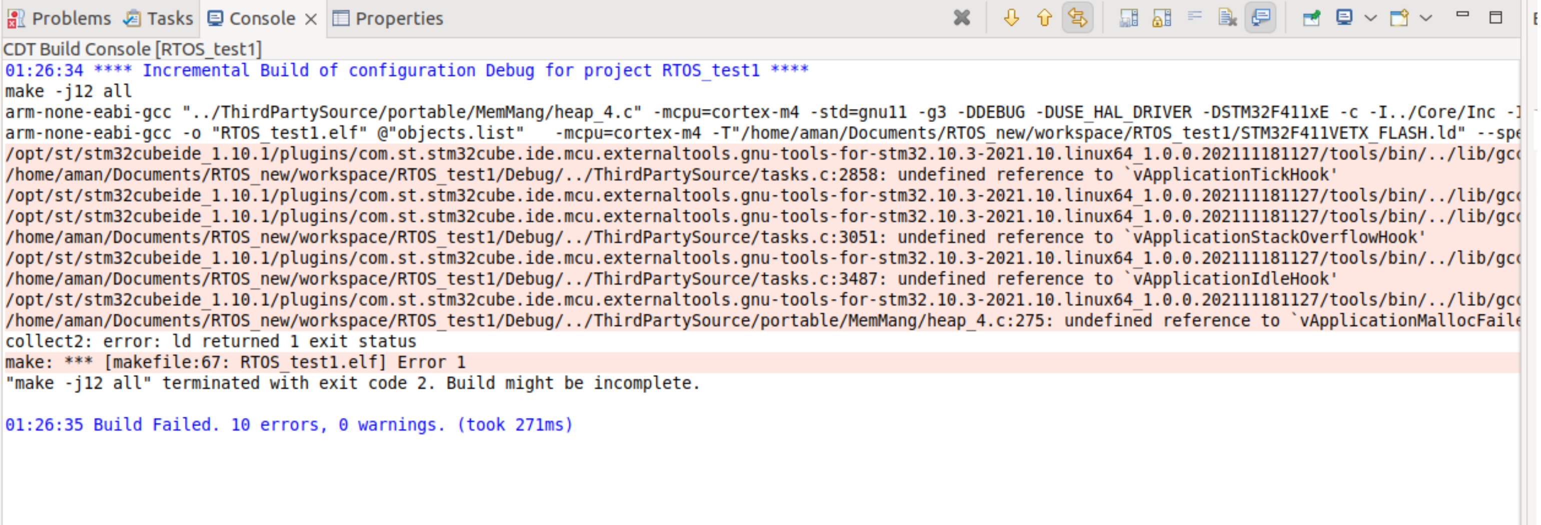
```
#define FREERTOS_CONFIG_H

/*-----------------------------------------------------------
 * Application specific definitions.
 *
 * These definitions should be adjusted for your particular hardware and
 * application requirements.
 *
 * THESE PARAMETERS ARE DESCRIBED WITHIN THE 'CONFIGURATION' SECTION OF THE
 * FreeRTOS API DOCUMENTATION AVAILABLE ON THE FreeRTOS.org WEB SITE.
 *
 * See http://www.freertos.org/a00110.html
 *----------------------------------------------------------*/

/* Ensure stdint is only used by the compiler, and not the assembler. */
#if defined(__ICCARM__) || defined(__GNUC__) || defined(__CC_ARM)
    #include <stdint.h>
    extern uint32_t SystemCoreClock;
#endif

#define configUSE_PREEMPTION            1
#define configUSE_IDLE_HOOK             1
#define configUSE_TICK_HOOK             1
#define configCPU_CLOCK_HZ              ( SystemCoreClock )
#define configTICK_RATE_HZ              ( ( TickType_t ) 1000 )
#define configMAX_PRIORITIES            ( 5 )
#define configMINIMAL_STACK_SIZE        ( ( unsigned short ) 130 )
#define configTOTAL_HEAP_SIZE           ( ( size_t ) ( 75 * 1024 ) )
#define configMAX_TASK_NAME_LEN         ( 10 )
#define configUSE_TRACE_FACILITY        1
#define configUSE_16_BIT_TICKS          0
#define configIDLE_SHOULD_YIELD         1
#define configUSE_MUTEXES               1
#define configQUEUE_REGISTRY_SIZE       8
```

```
 52 | #define configCPU_CLOCK_HZ    ( SystemCoreClock )
    |                                ^~~~~~~~~~~~~~~
/ThirdPartySource/portable/GCC/ARM_CM4F/port.c:101:50: note: in expansion of macro 'configCPU_CLOCK_HZ'
101 |     #define configSYSTICK_CLOCK_HZ            ( configCPU_CLOCK_HZ )
    |                                                  ^~~~~~~~~~~~~~~~~~
/ThirdPartySource/portable/GCC/ARM_CM4F/port.c:760:35: note: in expansion of macro 'configSYSTICK_CLOCK_HZ'
760 |     portNVIC_SYSTICK_LOAD_REG = ( configSYSTICK_CLOCK_HZ / configTICK_RATE_HZ ) - 1UL;
    |                                   ^~~~~~~~~~~~~~~~~~~~~~
ke: *** [ThirdPartySource/portable/GCC/ARM_CM4F/subdir.mk:19: ThirdPartySource/portable/GCC/ARM_CM4F/port.o] Error 1
ke: *** Waiting for unfinished jobs....
ake -j12 all" terminated with exit code 2. Build might be incomplete.

:16:04 Build Failed. 3 errors, 0 warnings. (took 374ms)
```

Next we need to select the Memory Management scheme that we're going to use. Hence, we will delete the Heap managers which we will not be using in the project code.



After this if we build we see the following errors.



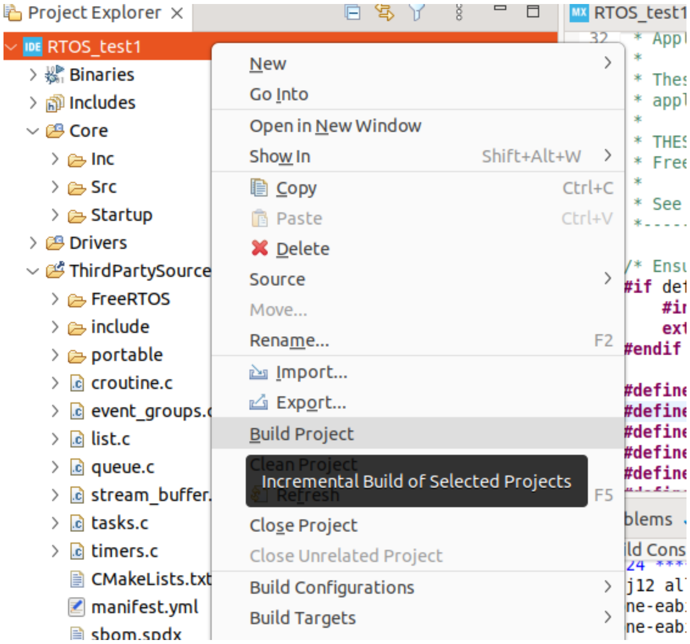
```
01:26:34 **** Incremental Build of configuration Debug for project RTOS_test1 ****
make -j12 all
arm-none-eabi-gcc "../ThirdPartySource/portable/MemMang/heap_4.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE_HAL_DRIVER -DSTM32F411xE -c -I../Core/Inc -I
arm-none-eabi-gcc -o "RTOS_test1.elf" @"objects.list"   -mcpu=cortex-m4 -T"/home/aman/Documents/RTOS_new/workspace/RTOS_test1/STM32F411VETX_FLASH.ld" --spe
/opt/st/stm32cubeide_1.10.1/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.10.3-2021.10.linux64_1.0.0.202111181127/tools/bin/../lib/gcc
/home/aman/Documents/RTOS_new/workspace/RTOS_test1/Debug/../ThirdPartySource/tasks.c:2858: undefined reference to `vApplicationTickHook'
/opt/st/stm32cubeide_1.10.1/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.10.3-2021.10.linux64_1.0.0.202111181127/tools/bin/../lib/gcc
/opt/st/stm32cubeide_1.10.1/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.10.3-2021.10.linux64_1.0.0.202111181127/tools/bin/../lib/gcc
/home/aman/Documents/RTOS_new/workspace/RTOS_test1/Debug/../ThirdPartySource/tasks.c:3051: undefined reference to `vApplicationStackOverflowHook'
/opt/st/stm32cubeide_1.10.1/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.10.3-2021.10.linux64_1.0.0.202111181127/tools/bin/../lib/gcc
/home/aman/Documents/RTOS_new/workspace/RTOS_test1/Debug/../ThirdPartySource/tasks.c:3487: undefined reference to `vApplicationIdleHook'
/opt/st/stm32cubeide_1.10.1/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.10.3-2021.10.linux64_1.0.0.202111181127/tools/bin/../lib/gcc
/home/aman/Documents/RTOS_new/workspace/RTOS_test1/Debug/../ThirdPartySource/portable/MemMang/heap_4.c:275: undefined reference to `vApplicationMallocFaile
collect2: error: ld returned 1 exit status
make: *** [makefile:67: RTOS_test1.elf] Error 1
"make -j12 all" terminated with exit code 2. Build might be incomplete.

01:26:35 Build Failed. 10 errors, 0 warnings. (took 271ms)
```

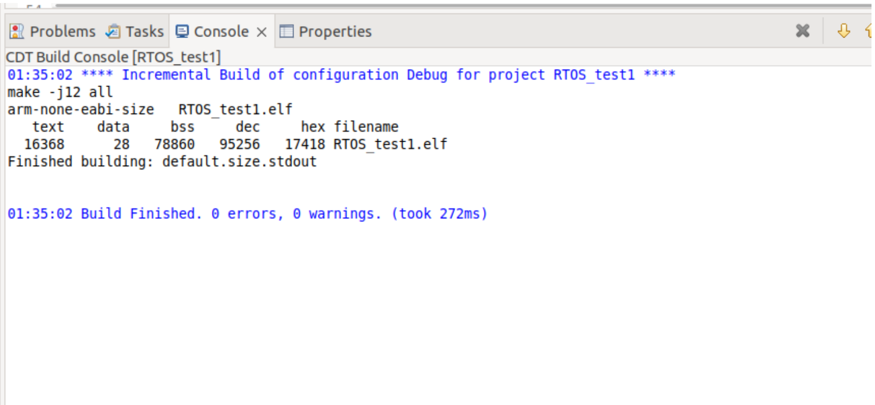
Here, if we notice the issue is coming because of the application hook macros, which are enabled by default but the build tool is not able to find the proper definitions for the same. Hence, either we need to provide the definition for these enabled hooks or we need to disable these hooks by setting to 0 in FreeRTOSconfig.h

```
-> #define configUSE_TICK_HOOK                0
-> #define configUSE_IDLE_HOOK                0
->#define configCHECK_FOR_STACK_OVERFLOW      0
->#define configUSE_MALLOC_FAILED_HOOK        0
```

Make sure to ask questions on what these are :)

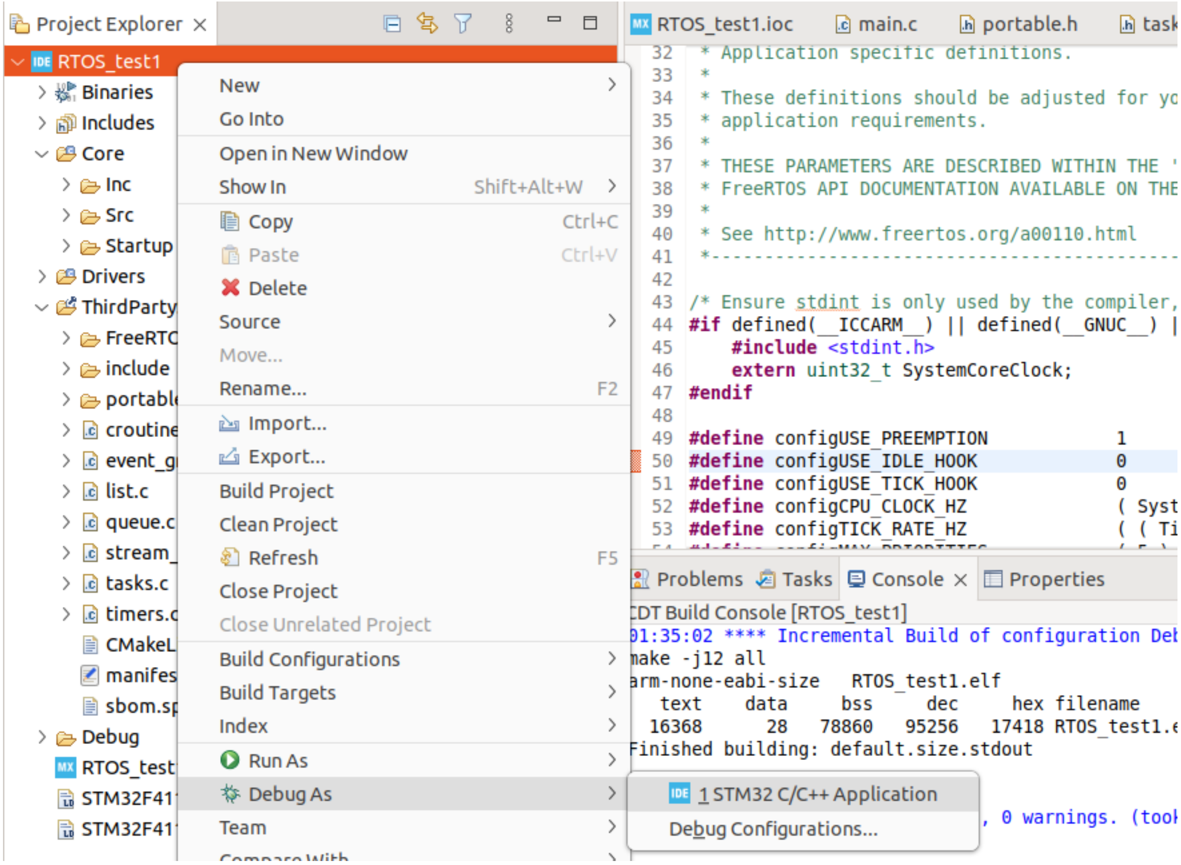
Initiating Build





```
01:35:02 **** Incremental Build of configuration Debug for project RTOS_test1 ****
make -j12 all
arm-none-eabi-size   RTOS_test1.elf
   text    data     bss     dec     hex filename
  16368      28   78860   95256   17418 RTOS_test1.elf
Finished building: default.size.stdout


01:35:02 Build Finished. 0 errors, 0 warnings. (took 272ms)
```

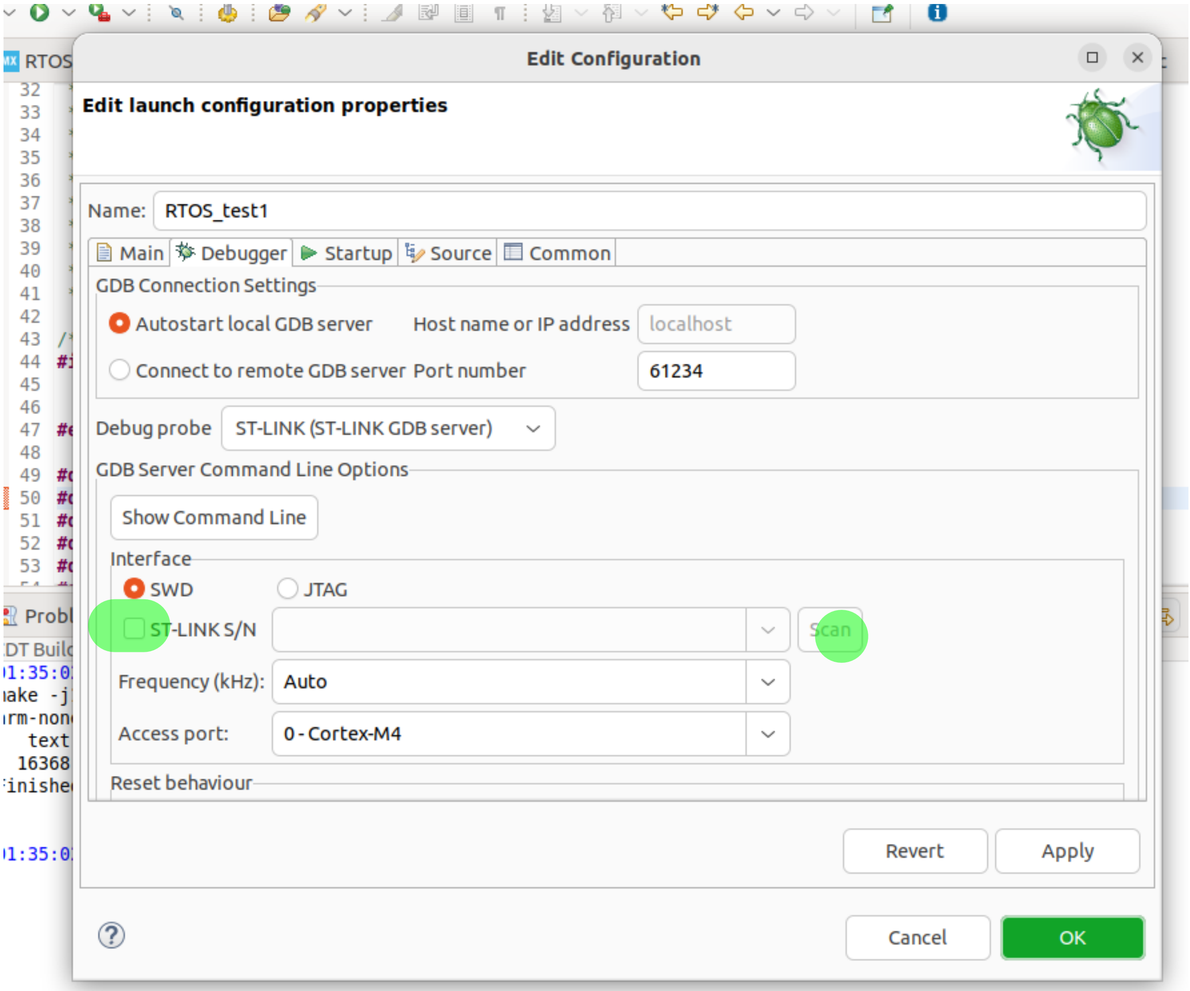
Now connect the STM32 board and then right click on the project and Debug as ->



```
 * Application specific definitions.
 *
 * These definitions should be adjusted for yo
 * application requirements.
 *
 * THESE PARAMETERS ARE DESCRIBED WITHIN THE '
 * FreeRTOS API DOCUMENTATION AVAILABLE ON THE
 *
 * See http://www.freertos.org/a00110.html
 *----------------------------------------------

/* Ensure stdint is only used by the compiler,
#if defined(__ICCARM__) || defined(__GNUC__) |
    #include <stdint.h>
    extern uint32_t SystemCoreClock;
#endif

#define configUSE_PREEMPTION          1
#define configUSE_IDLE_HOOK           0
#define configUSE_TICK_HOOK           0
#define configCPU_CLOCK_HZ            ( Syst
#define configTICK_RATE_HZ            ( ( Ti
```

```
CDT Build Console [RTOS_test1]
01:35:02 **** Incremental Build of configuration Deb
make -j12 all
arm-none-eabi-size   RTOS_test1.elf
   text    data     bss     dec     hex filename
  16368      28   78860   95256   17418 RTOS_test1.e
Finished building: default.size.stdout

, 0 warnings. (took
```

Under debugger select following check box and click scan



Edit launch configuration properties

Name: RTOS_test1

GDB Connection Settings

● Autostart local GDB server   Host name or IP address: localhost

○ Connect to remote GDB server   Port number: 61234

Debug probe: ST-LINK (ST-LINK GDB server)

GDB Server Command Line Options

Interface

● SWD   ○ JTAG

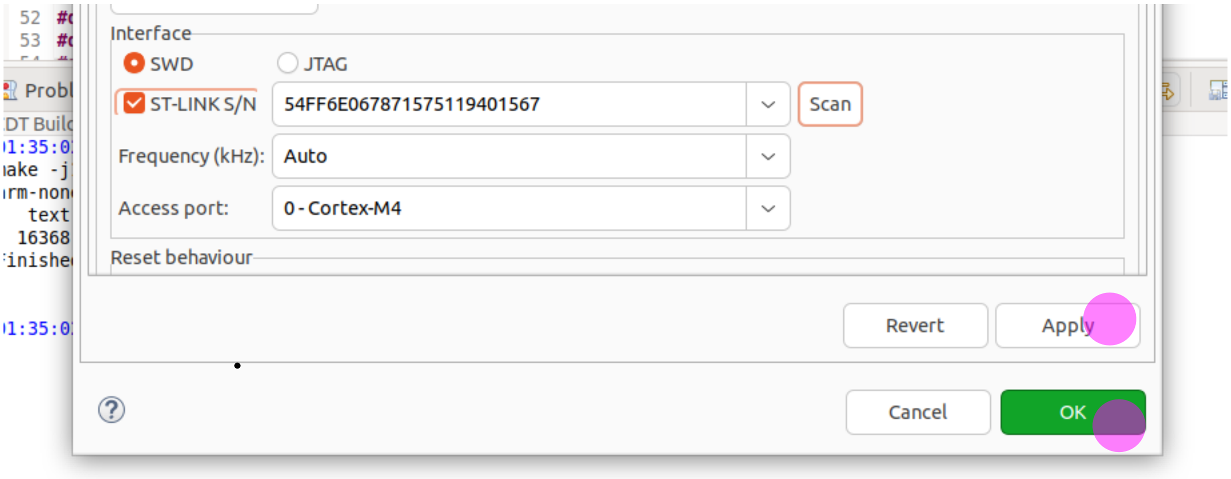
□ ST-LINK S/N

Frequency (kHz): Auto

Access port: 0 - Cortex-M4

Reset behaviour

It will populate the attached STLINK debugger and list the serial number of the same.
Click apply and ok



Interface

● SWD ○ JTAG

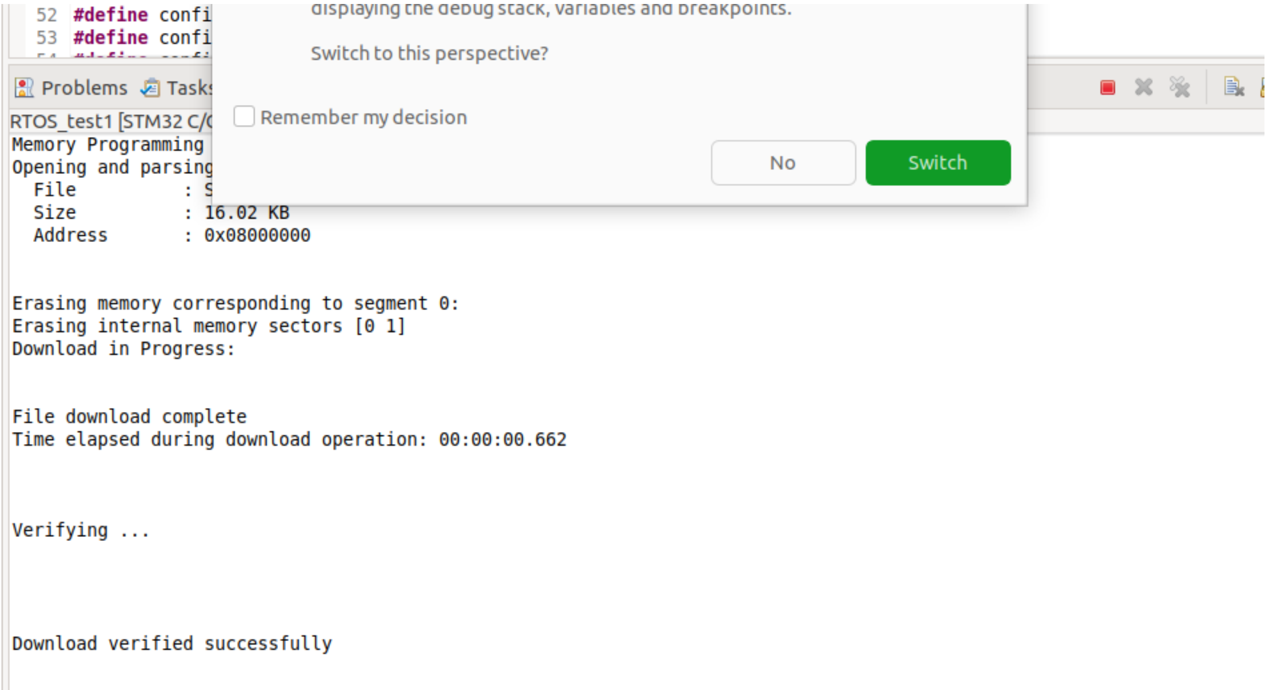
☑ ST-LINK S/N: 54FF6E067871575119401567

Frequency (kHz): Auto

Access port: 0 - Cortex-M4

Reset behaviour

It will show like following once, downloaded.



displaying the debug stack, variables and breakpoints.

Switch to this perspective?

□ Remember my decision

```
RTOS_test1 [STM32 C/C
Memory Programming
Opening and parsing
  File          : S
  Size          : 16.02 KB
  Address       : 0x08000000


Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 1]
Download in Progress:


File download complete
Time elapsed during download operation: 00:00:00.662



Verifying ...




Download verified successfully
```

DONE :)
-Aman Kanwar