AN5361

Application note

Getting started with projects based on dual-core STM32H7 microcontrollers in STM32CubeIDE

Introduction

This application note describes how to get started with projects based on STM32H7 Series dual-core microcontrollers in the STMicroelectronics STM32CubeIDE integrated development environment.

1 General information

STM32CubeIDE supports STM32 32-bit products based on the Arm® Cortex® processor.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 Prerequisites

The following tools are prerequisites for understanding the tutorial in this document and developing an application based on the STM32H7 Series:

- STM32CubeIDE 1.4.0 or newer
- STM32Cube_FW_H7_V1.7.0 or newer
- STM32CubeMX 6.0.0 or newer

Users are advised to keep updated with the documentation evolution of the STM32H7 Series at www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html.

1.2 The use cases in this document

In the STM32CubeIDE context, users have different ways to explore and get started with the development of projects based on the STM32H7 Series. From the list below, select the description that best fits the use case considered and refer to the corresponding section in this application note:

- I already have an SW4STM32 project with an `ioc` file:
  Refer to Section 2.2 Import an SW4STM32 project with an ioc file
- I already have an SW4STM32 project without an `ioc` file:
  Refer to Section 2.3 Import an SW4STM32 project without an ioc file
- I want to learn with and explore example projects:
  Refer to Section 2.5 Import a project from the STM32CubeH7 MCU Package
- I want to start a first STM32H7 project:
  – Empty project – No STM32CubeMX support for maximum flexibility.
    Refer to Section 2.4 Create an empty project based on the template in the STM32CubeH7 MCU Package
  – STM32CubeH7 project – STM32CubeMX-managed project.
    Refer to Section 2.1 Create a new STM32 project

1.3 Specific features of dual-core microcontrollers in the STM32H7 Series

The most obvious feature of a dual-core STM32H7 device is its significant performance boost that comes from the addition of an Arm® Cortex®-M4 core besides the Arm® Cortex®-M7 core. Dual-core STM32H7 devices are also very flexible. This application note is a simple getting-started guide to get up and running with a debug session when both cores are running. It does not cover such additional features as the flexibility of being able to boot on either core or the fact that each core can exist in an independent power domain to optimize energy consumption.

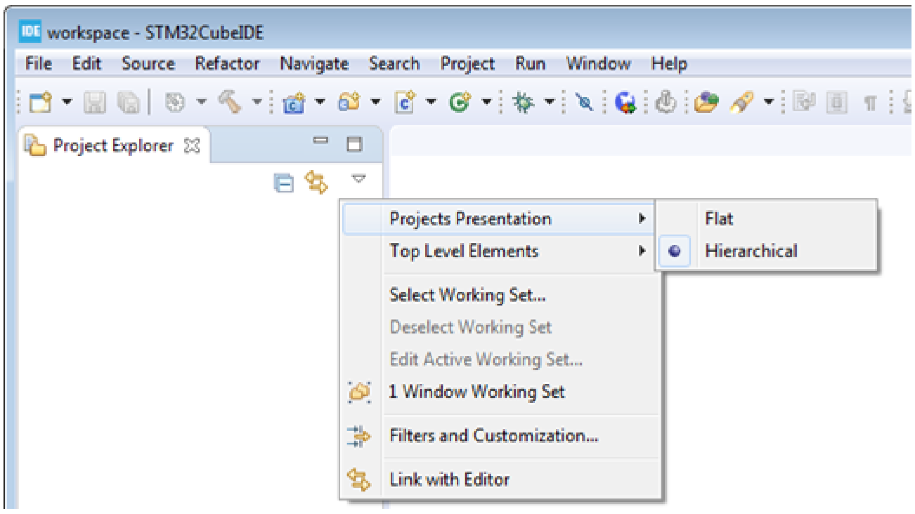
1.3.1 Dual-core STM32H7 project structure

When a dual-core STM32H7 project is created, its structure is automatically made hierarchical. The project structure for single-core projects is flat. On the contrary, in a multi-core project, the hierarchical project structure is used. When the user creates or imports a dual-core STM32H7 project, it consists of one root project together with sub-projects, referred to as MCU projects, for each core.

The MCU projects are real CDT™ projects that can contain both build and debug configurations. On the contrary, the root project is a simple container that allows sharing common code between the cores. The root project can contain neither build nor debug configurations.

If the project is not shown in a hierarchical structure, this can be changed as shown in Figure 1.

Figure 1. Setting the project hierarchical view



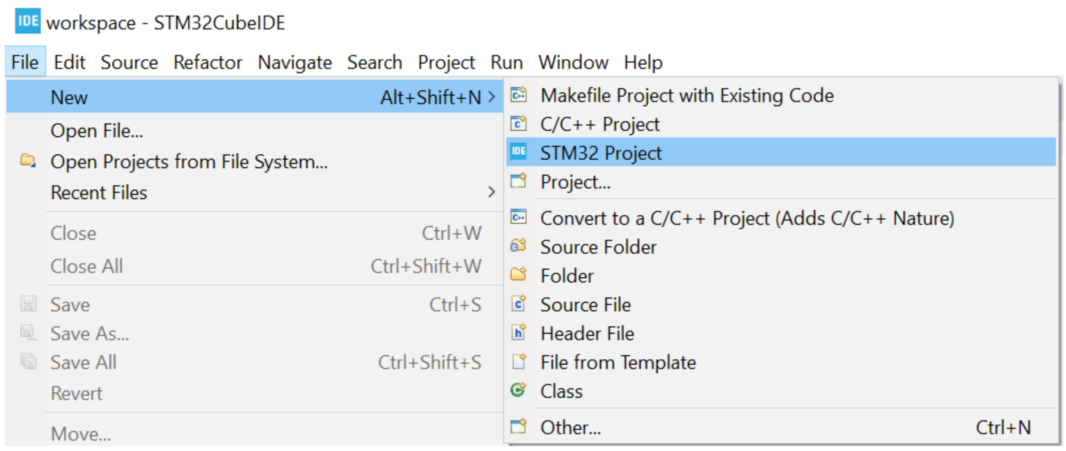
2 Create and import projects

This chapter describes how to create or import projects for dual-core microcontrollers in the STM32H7 Series.

2.1 Create a new STM32 project

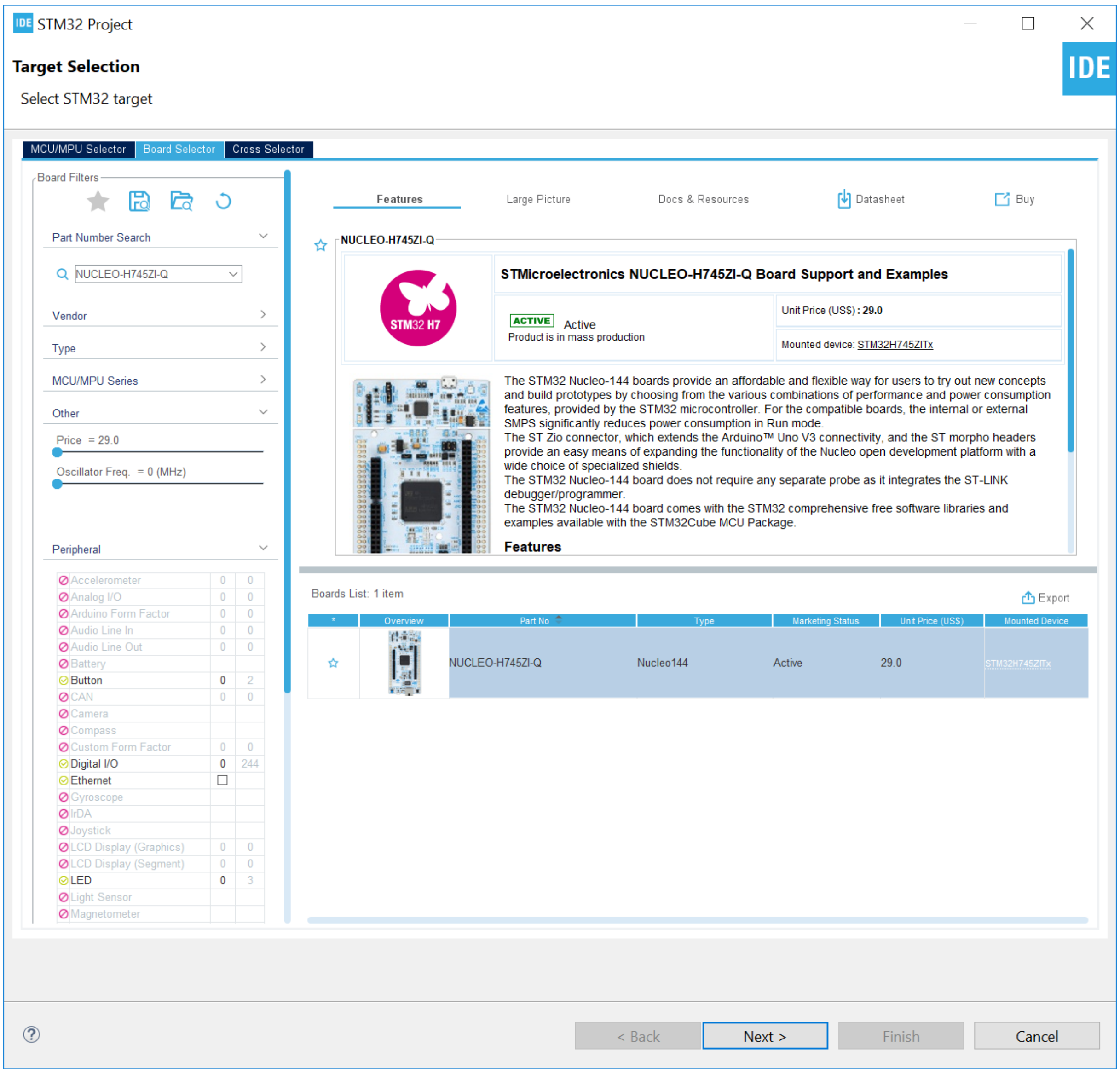
To start a new project, go to [**File**]>[**New**]>[**STM32 Project**] as shown in Figure 2.

Figure 2. **New STM32 project**



Select the desired MCU or board. In the example illustrated in Figure 3, the selected board is the NUCLEO-H745ZI-Q. Click on [**Next >**].

Figure 3. Target selection



STM32 Project

Target Selection

Select STM32 target

Part Number Search: NUCLEO-H745ZI-Q

Vendor

Type

MCU/MPU Series

Other

Price = 29.0

Oscillator Freq. = 0 (MHz)

Peripheral

Accelerometer	0	0
Analog I/O	0	0
Arduino Form Factor	0	0
Audio Line In	0	0
Audio Line Out	0	0
Battery		
Button	0	2
CAN	0	0
Camera		
Compass		
Custom Form Factor	0	0
Digital I/O	0	244
Ethernet	☐	
Gyroscope		
IrDA		
Joystick		
LCD Display (Graphics)	0	0
LCD Display (Segment)	0	0
LED	0	3
Light Sensor		
Magnetometer		

NUCLEO-H745ZI-Q

STMicroelectronics NUCLEO-H745ZI-Q Board Support and Examples

STM32 H7

ACTIVE Active
Product is in mass production

Unit Price (US$) : **29.0**

Mounted device: STM32H745ZITx

The STM32 Nucleo-144 boards provide an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.
The ST Zio connector, which extends the Arduino™ Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields.
The STM32 Nucleo-144 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.
The STM32 Nucleo-144 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

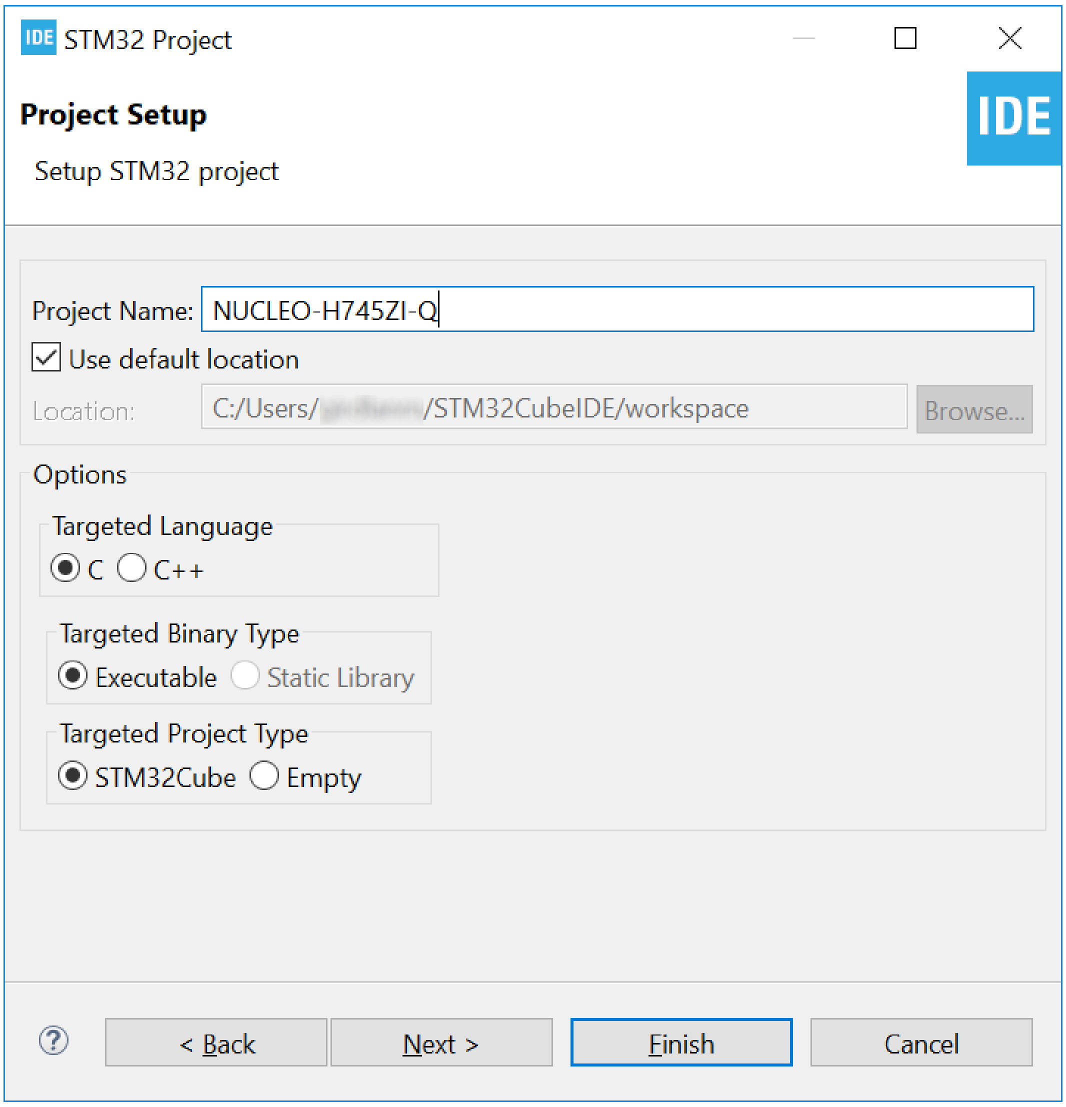
Features

Boards List: 1 item

*	Overview	Part No	Type	Marketing Status	Unit Price (US$)	Mounted Device
☆		NUCLEO-H745ZI-Q	Nucleo144	Active	29.0	STM32H745ZITx

After the target selection comes the project setup step shown in Figure 4. The *Targeted Project Type* setting determines whether the project gets generated by STM32CubeMX or not. An *Empty* project is a skeleton of a project that needs building upon while *STM32Cube* indicates an STM32CubeMX-managed project.

Figure 4. Projet setup



STM32 Project

Project Setup

Setup STM32 project

Project Name: NUCLEO-H745ZI-Q

[✓] Use default location

Location: C:/Users/[redacted]/STM32CubeIDE/workspace

Options

Targeted Language: [✓ C] C++

Targeted Binary Type: [✓ Executable] Static Library

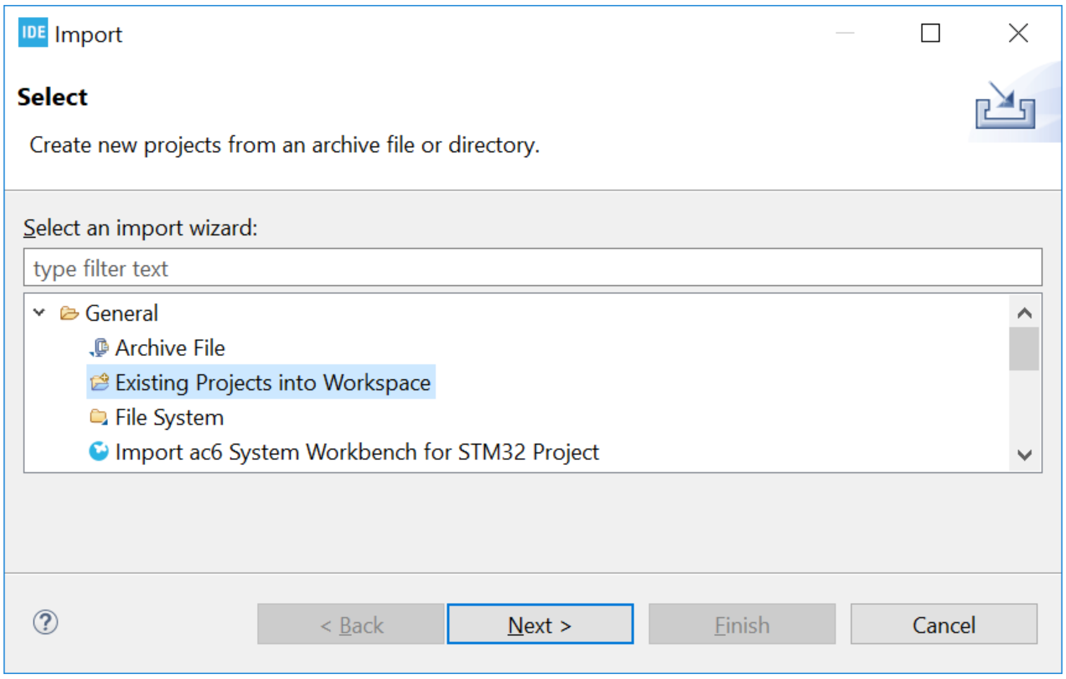
Targeted Project Type: [✓ STM32Cube] Empty

2.2 Import an SW4STM32 project with an `ioc` file

If the project already contains an `ioc` file, the easiest way to import the project into a working STM32CubeIDE environment is to copy it and open the copy through STM32CubeMX stand alone, then, in the *Project Manager*, change the *Toolchain / IDE* to STM32CubeIDE and regenerate the project.

After the project is regenerated, go to [**File**]>[**Import…**] and choose to import it as an *Existing projects into workspace* as shown in Figure 5.

Figure 5. Import an existing projet with an ioc file

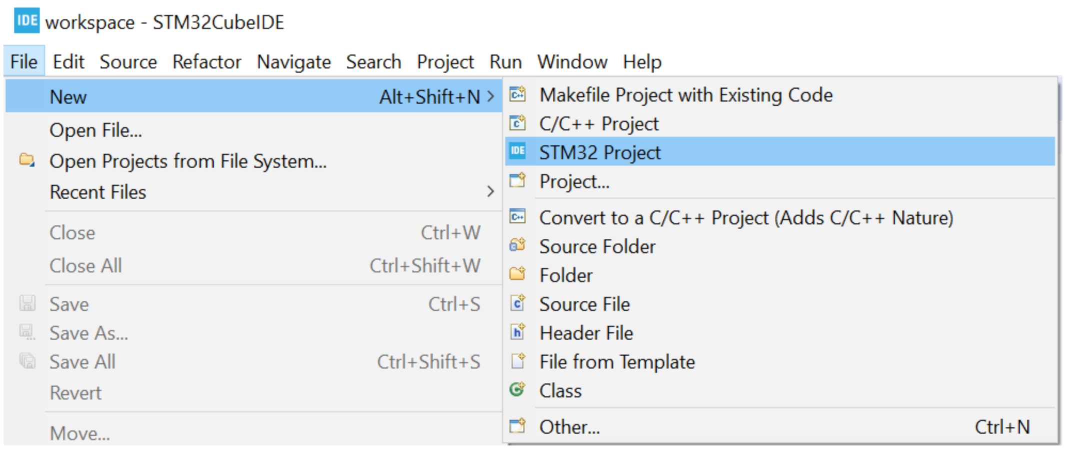

Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Archive File
Existing Projects into Workspace
File System
Import ac6 System Workbench for STM32 Project
< Back
Next >
Finish
Cancel

Then copy the code inside the different `/* USER CODE BEGIN */` blocks that exist in the project into the new STM32CubeIDE environment.

2.3 Import an SW4STM32 project without an `ioc` file

To make sure the project gets a hierarchical structure, the recommended way is to go to [**File**]>[**New**]>[**STM32 Project**] as shown in Figure 6.

Figure 6. New STM32 project



Select the device for the project being imported and click on [**Next >**].

When setting up the project as shown in , make sure the *Targeted Project Type* is set to *Empty* and click on [**Finish**].

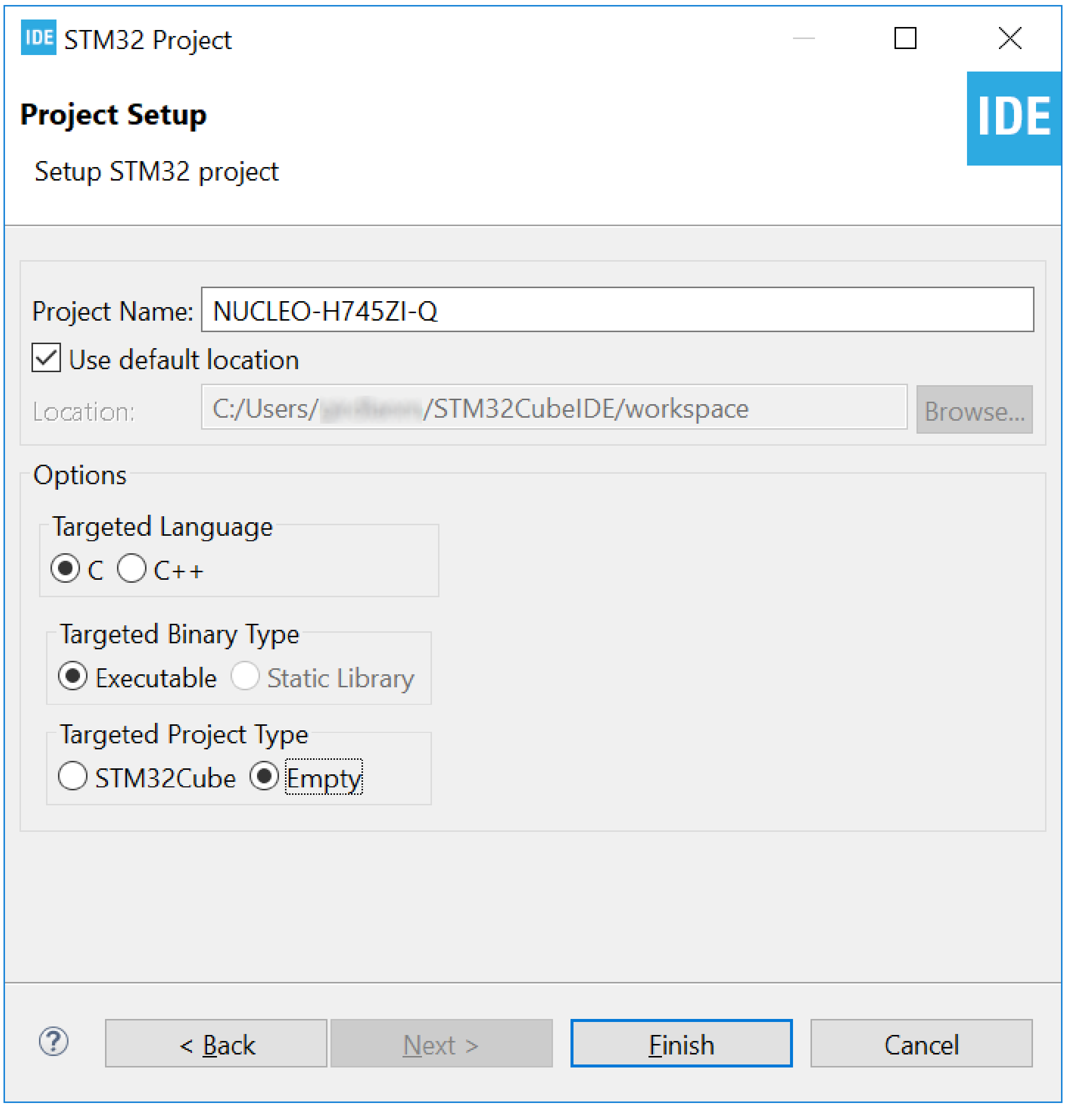


Figure 7. **Projet setup**

STM32 Project

Project Setup

Setup STM32 project

Project Name: NUCLEO-H745ZI-Q

[✓] Use default location

Location: C:/Users/[illegible]/STM32CubeIDE/workspace

Options

Targeted Language: [● C] ○ C++

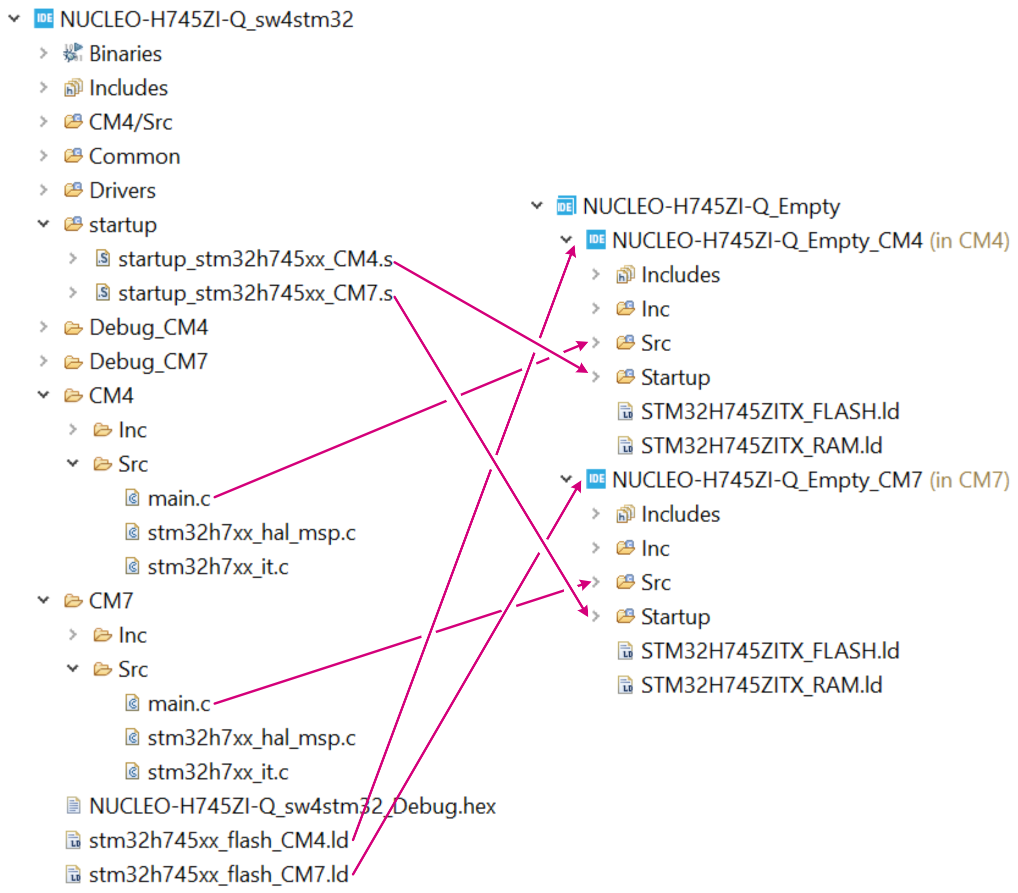
Targeted Binary Type: [● Executable] ○ Static Library

Targeted Project Type: ○ STM32Cube [● Empty]

After the empty hierarchical project is generated:

1. Go to **[File**]>**[Import…]**
2. Import the SW4STM32 project as *Import ac6 System Workbench for STM32 Project*
3. Copy and paste the project files and resources into the sub-project of the empty project as shown in Figure 8

Figure 8. Copy project content to empty sub-project



Note: *It is not recommended to import the* `.cproject`, `.project` *or* `.settings` *files.*

It is important to remember to configure also the same build settings used previously while the project was in the SW4STM32 environment. If the project contains linked resources, these need to be updated to point to the correct resource in the file system.

This process is necessary because, when importing without any special treatment a project from SW4STM32 that does not have an `ioc` file, it is imported into STM32CubeIDE with a flat project structure.

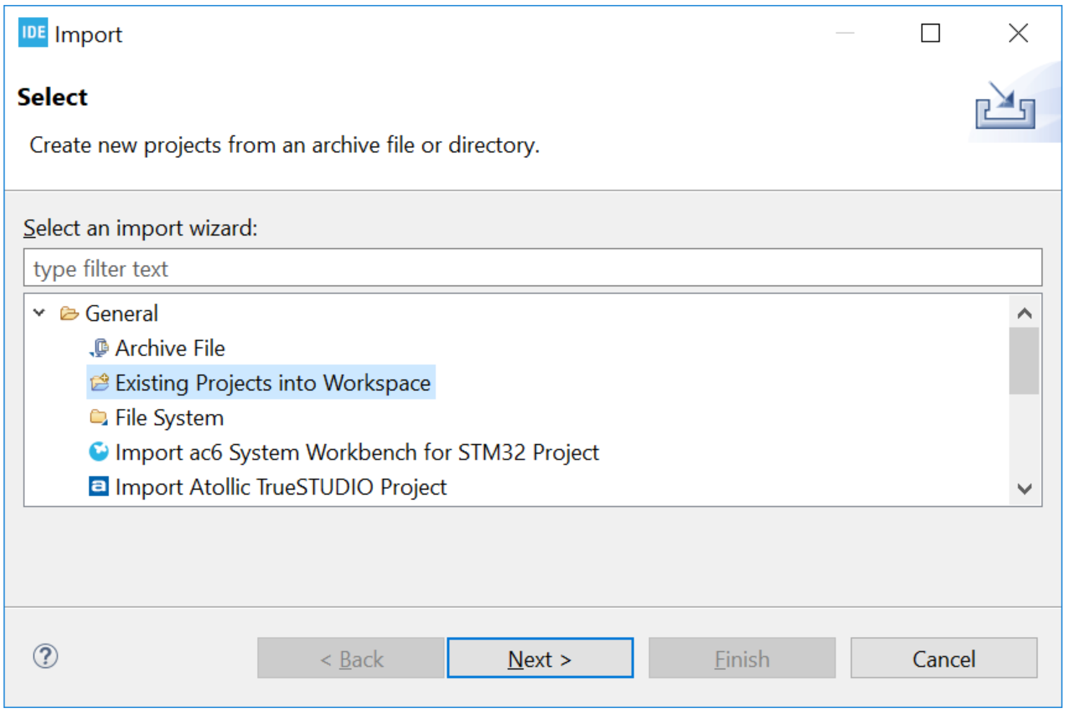
2.4 Create an empty project based on the template in the STM32CubeH7 MCU Package

Follow the same steps as in Section 2.3 Import an SW4STM32 project without an ioc file but use STM32Cube_FW_H7 firmware in the STM32CubeH7 MCU Package as input.

2.5 Import a project from the STM32CubeH7 MCU Package

In order to import the STM32Cube firmware project into STM32CubeIDE, go to [**File**]>[**Import**], select *Existing Projects into Workspace* as shown in Figure 9, and click on [**Next >**].

Figure 9. **Import of firmware project into STM32CubeIDE**



Select

Create new projects from an archive file or directory.

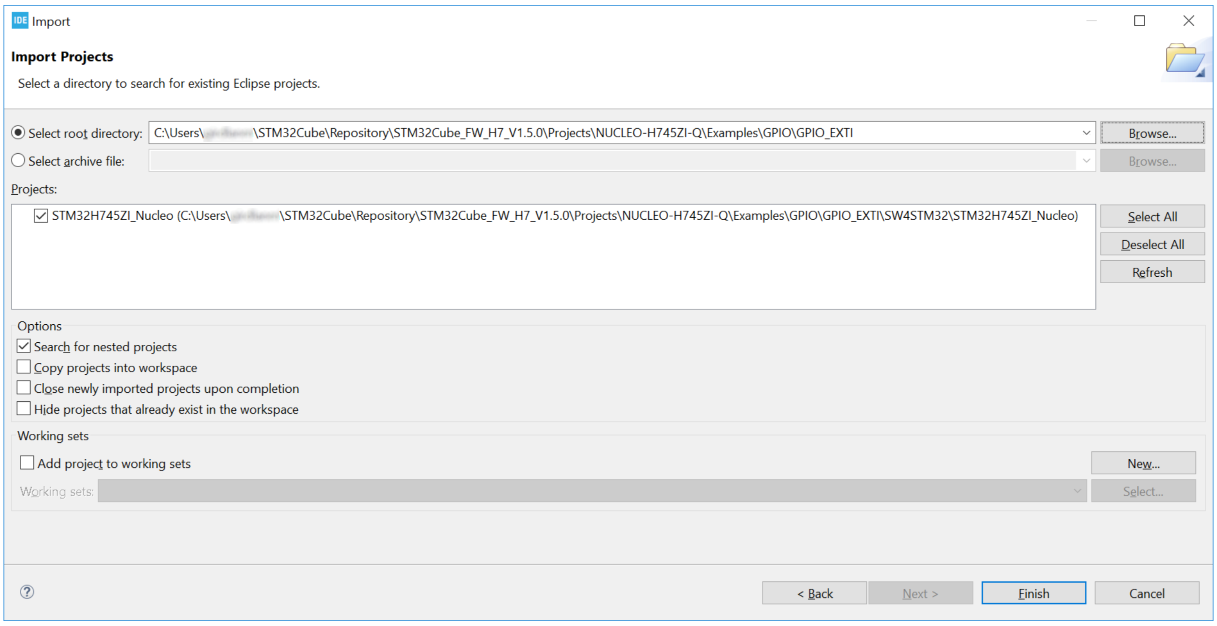
Select an import wizard:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Import ac6 System Workbench for STM32 Project
  - Import Atollic TrueSTUDIO Project

Then select the correct project. A project example is by default located at `$HOME\STM32Cube\Repository\STM32Cube_FW_H7_VX.X.X\Projects\NUCLEO-H745ZI-Q\Examples\GPIO\GPIO_EXTI`.

Figure 10. **Firmware project selection**



Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\Users\[redacted]\STM32Cube\Repository\STM32Cube_FW_H7_V1.5.0\Projects\NUCLEO-H745ZI-Q\Examples\GPIO\GPIO_EXTI

( ) Select archive file:

Projects:

- [x] STM32H745ZI_Nucleo (C:\Users\[redacted]\STM32Cube\Repository\STM32Cube_FW_H7_V1.5.0\Projects\NUCLEO-H745ZI-Q\Examples\GPIO\GPIO_EXTI\SW4STM32\STM32H745ZI_Nucleo)

Options

- [x] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Close newly imported projects upon completion
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

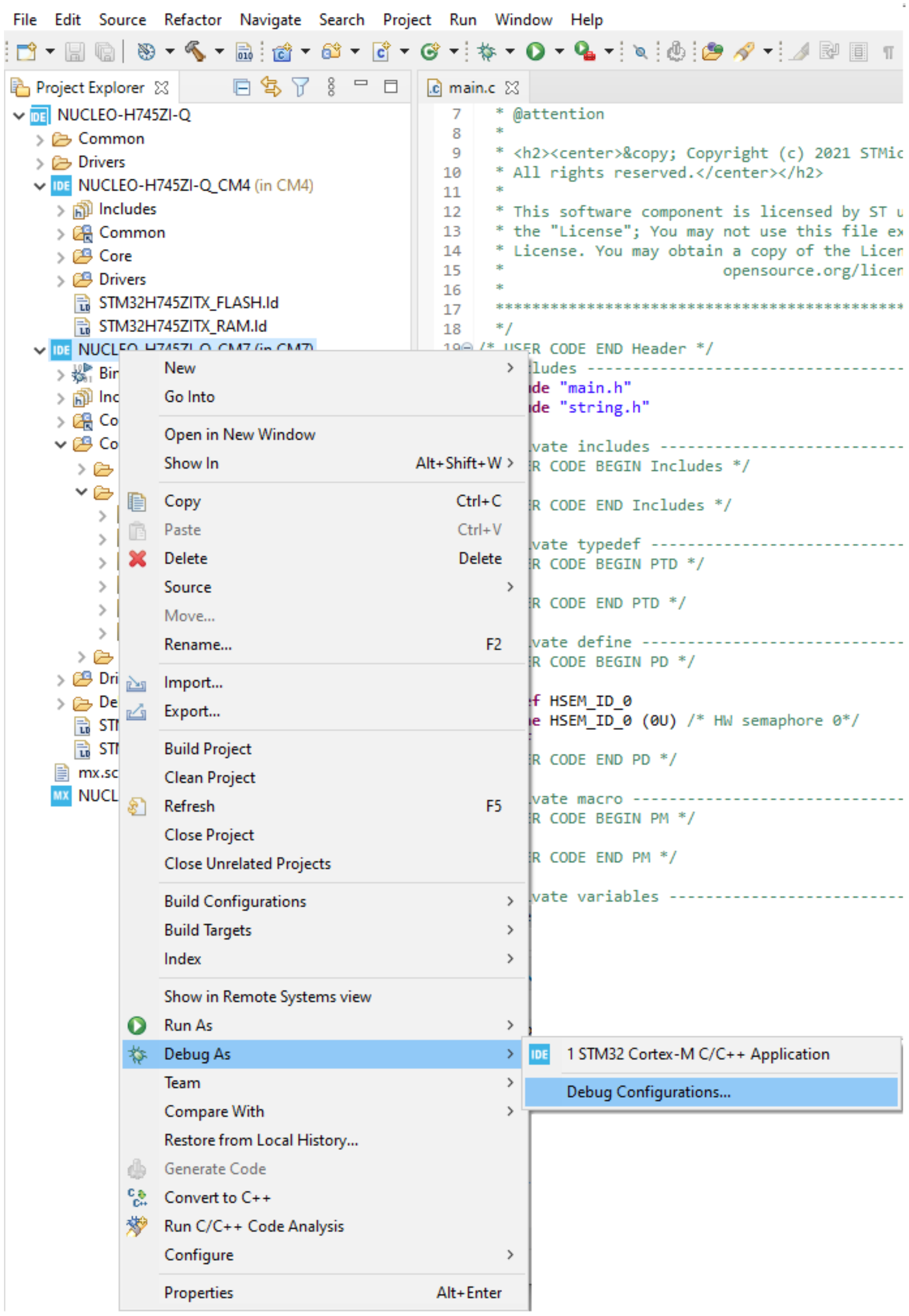
After selecting the project, click on [**Finish**] to import and build the project.

3 Debugging

This chapter highlights some of the points to bear in mind while debugging a device in the STM32H7 Series. In the next two sections, this application note covers the configurations needed to start debug sessions with ST-LINK GDB server and OpenOCD.

To start configuring the launch of the debug session, right-click the sub-project (in this example, the Cortex®-M7) and select [**Debug As**]>[**Debug Configurations...**] as shown in Figure 11.

Figure 11. ST-LINK GDB server debug configuration (1 of 6) and OpenOCD debug configuration (1 of 3)



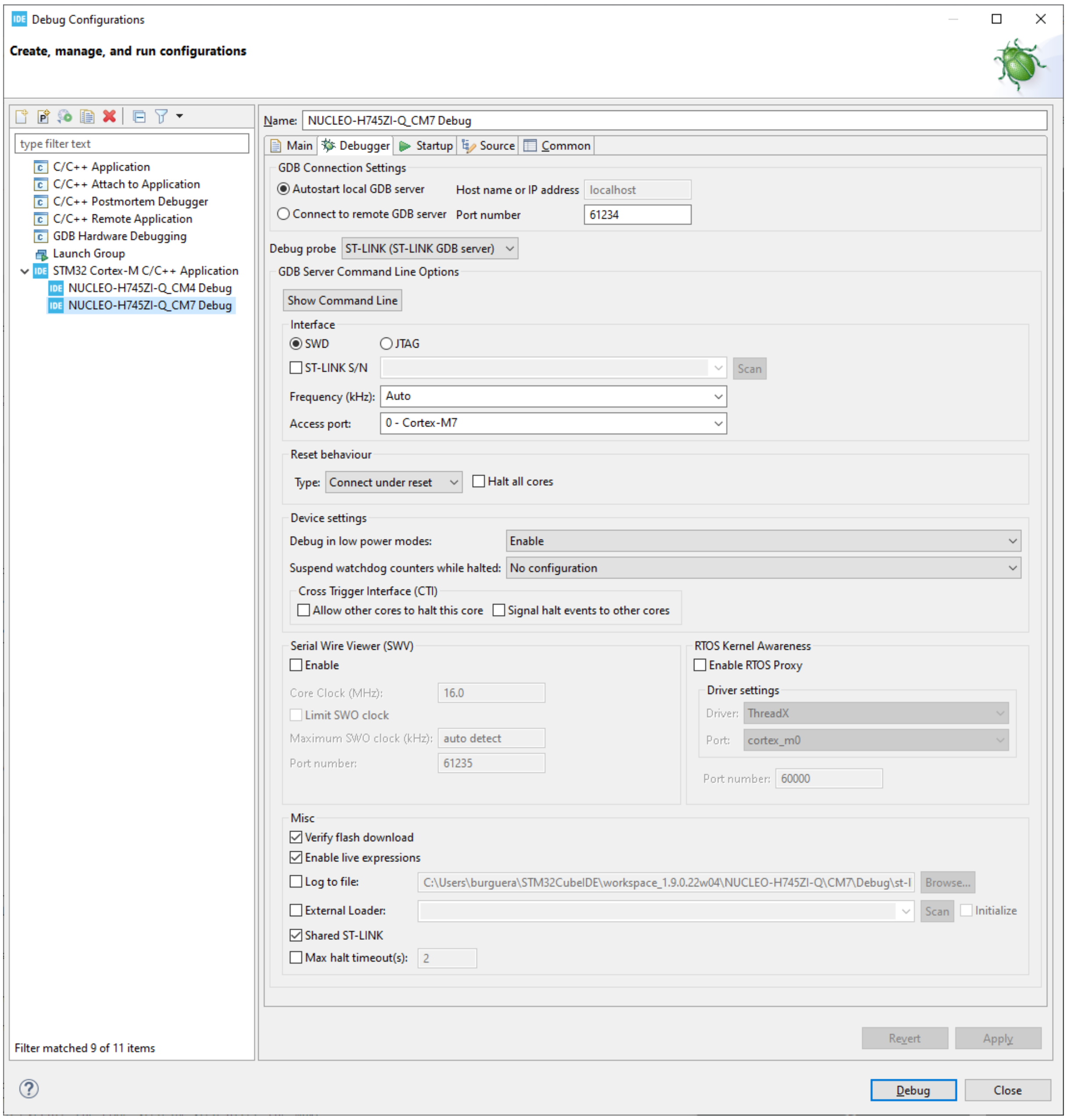
```
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2021 STMic
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST u
 * the "License"; You may not use this file ex
 * License. You may obtain a copy of the Licer
 *                        opensource.org/licer
 *
 ******************************************
 */
/* USER CODE END Header */
```

Note: For the rest of this section, examples are based on the NUCLEO-H755ZI-Q board.

3.1 Setting up with ST-LINK GDB server

Since the STM32CubeMX default project puts the Cortex®-M4 in the Deepsleep mode, the [**Reset behaviour**] field must be set to *Connect under reset* to be able to communicate with the CPU (refer to Figure 12). The *Halt all cores* option must not be set. Setting it prevents the Cortex®-M4 from starting to execute the code. The Cortex®-M4 must execute the code and go to Stop mode before the Cortex®-M7 can start executing the code and wake up the Cortex®-M4 by releasing a hardware semaphore. In the debug configuration, select *Enable shared ST-LINK*, switch to the *Startup* tab and select [**Add…**].

Figure 12. ST-LINK GDB server debug configuration (2 of 6)


Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- STM32 Cortex-M C/C++ Application
  - NUCLEO-H745ZI-Q_CM4 Debug
  - NUCLEO-H745ZI-Q_CM7 Debug

Filter matched 9 of 11 items

Name: NUCLEO-H745ZI-Q_CM7 Debug

Main | Debugger | Startup | Source | Common

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
Show Command Line

Interface
[✓ SWD] JTAG
☐ ST-LINK S/N Scan
Frequency (kHz): Auto
Access port: 0 - Cortex-M7

Reset behaviour
Type: Connect under reset ☐ Halt all cores

Device settings
Debug in low power modes: Enable
Suspend watchdog counters while halted: No configuration
Cross Trigger Interface (CTI)
☐ Allow other cores to halt this core ☐ Signal halt events to other cores

Serial Wire Viewer (SWV)
☐ Enable
Core Clock (MHz): 16.0
☐ Limit SWO clock
Maximum SWO clock (kHz): auto detect
Port number: 61235

RTOS Kernel Awareness
☐ Enable RTOS Proxy
Driver settings
Driver: ThreadX
Port: cortex_m0
Port number: 60000

Misc
☑ Verify flash download
☑ Enable live expressions
☐ Log to file: C:\Users\burguera\STM32CubeIDE\workspace_1.9.0.22w04\NUCLEO-H745ZI-Q\CM7\Debug\st-l Browse...
☐ External Loader: Scan ☐ Initialize
☑ Shared ST-LINK
☐ Max halt timeout(s): 2

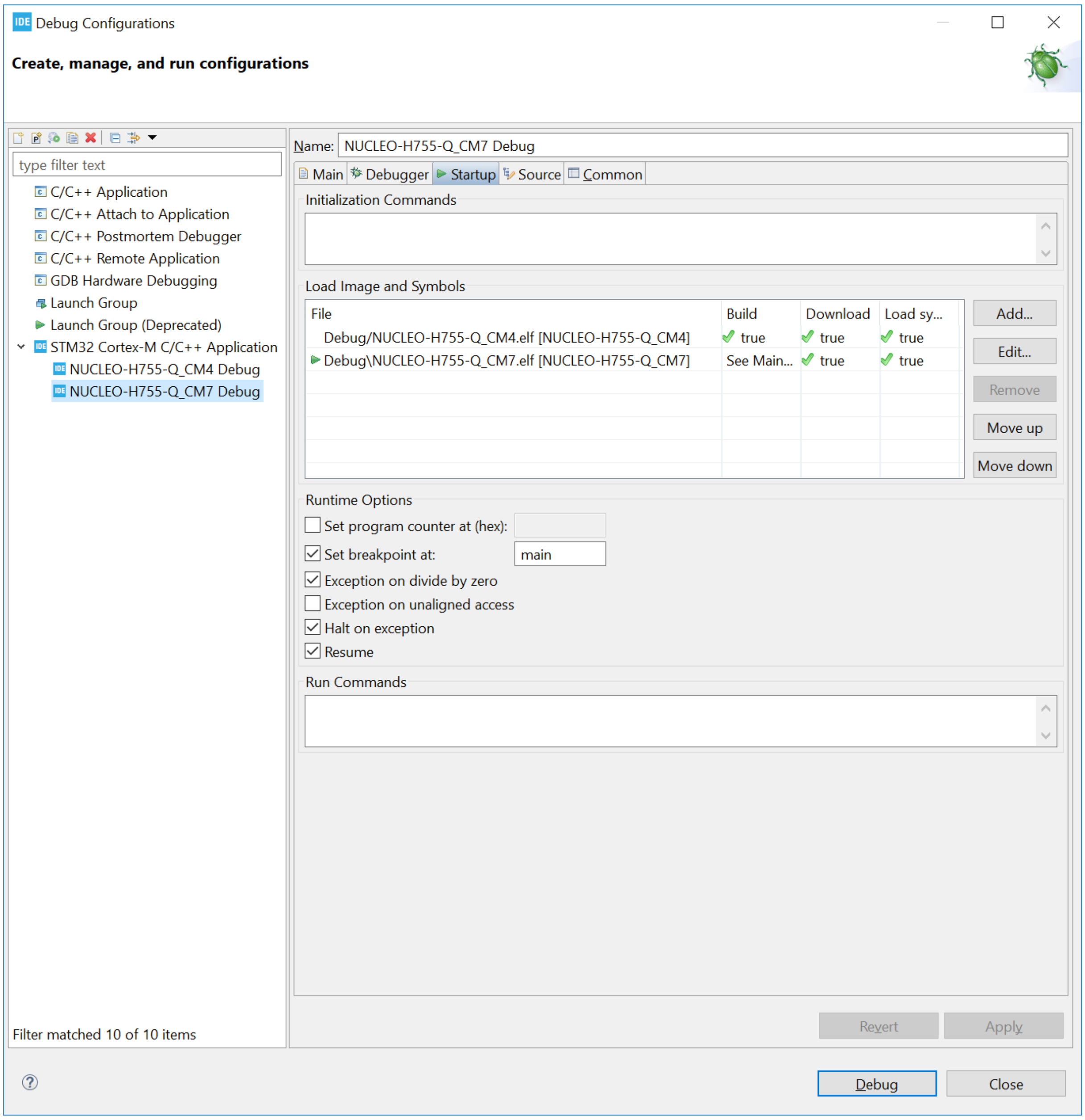
Revert Apply

Debug Close

Startup tab - Cortex®-M7

The Cortex®-M7 debug configuration is responsible for loading both the Cortex®-M7 and Cortex®-M4 images. Go to the *Startup* tab to do this as shown in Figure 13:

Figure 13. ST-LINK GDB server debug configuration (3 of 6)



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)
- STM32 Cortex-M C/C++ Application
  - NUCLEO-H755-Q_CM4 Debug
  - NUCLEO-H755-Q_CM7 Debug

Filter matched 10 of 10 items

Name: NUCLEO-H755-Q_CM7 Debug

Main | Debugger | Startup | Source | Common

Initialization Commands

Load Image and Symbols

File	Build	Download	Load sy...
Debug/NUCLEO-H755-Q_CM4.elf [NUCLEO-H755-Q_CM4]	✓ true	✓ true	✓ true
Debug\NUCLEO-H755-Q_CM7.elf [NUCLEO-H755-Q_CM7]	See Main...	✓ true	✓ true

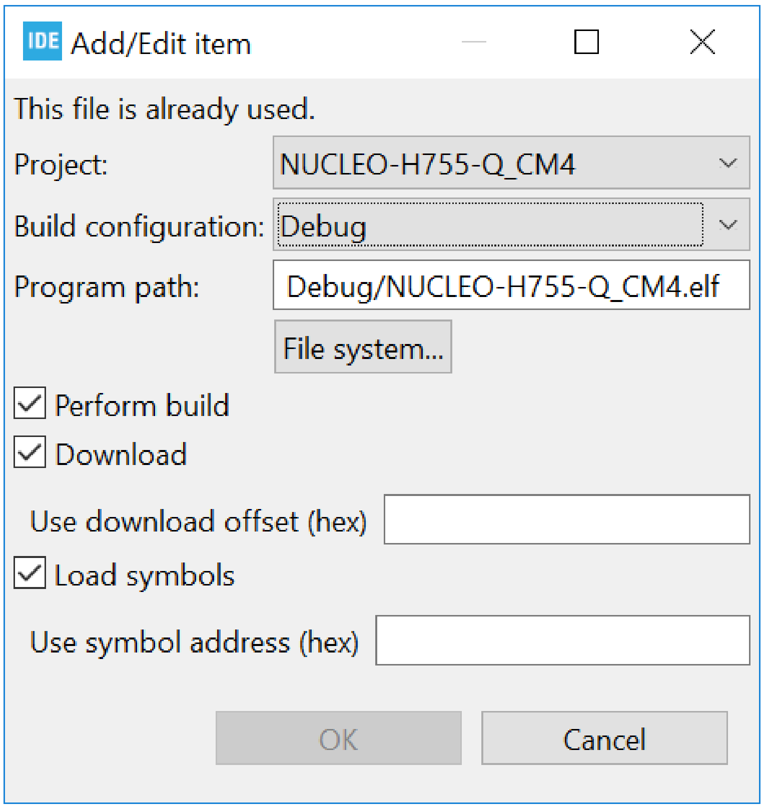
Runtime Options

- [ ] Set program counter at (hex):
- [x] Set breakpoint at: main
- [x] Exception on divide by zero
- [ ] Exception on unaligned access
- [x] Halt on exception
- [x] Resume

Run Commands

To download also the Cortex®-M4 image, click [**Add...**], browse the correct project, and build the configuration. The result is shown in Figure 14.

Figure 14. ST-LINK GDB server debug configuration (4 of 6)



Add/Edit item

This file is already used.

Project:	NUCLEO-H755-Q_CM4
Build configuration:	Debug
Program path:	Debug/NUCLEO-H755-Q_CM4.elf

☑ Perform build
☑ Download
Use download offset (hex)
☑ Load symbols
Use symbol address (hex)

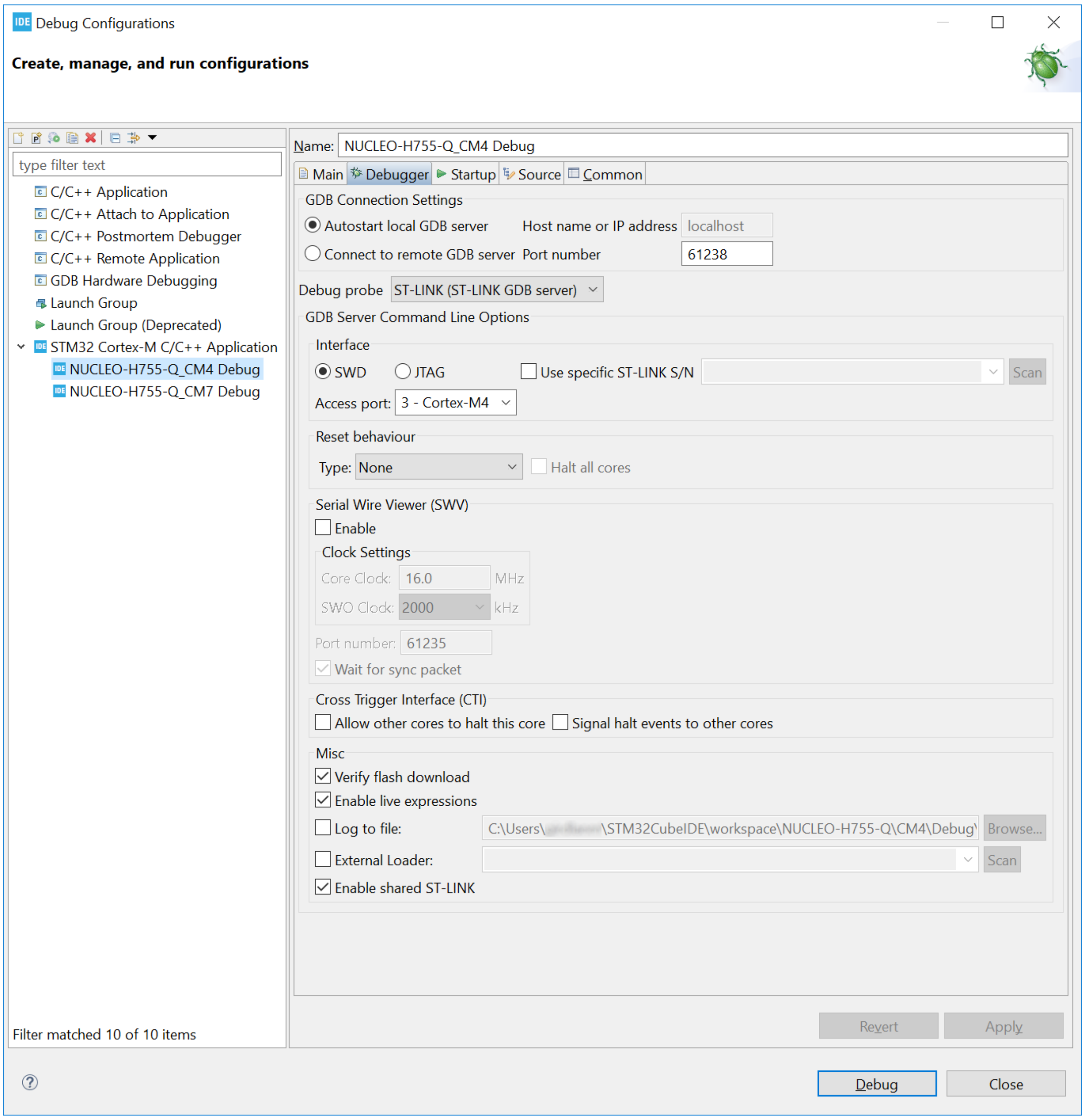
The order in the load list is very important. The last entry in the list, marked by a green arrow (refer to Figure 13, is the image debugged with this debug configuration. Consequently, the debugger fetches the program counter value (PC) from this image.

Startup tab - Cortex®-M4

As shown in Figure 15:

- Make sure that the *Port number* exceeds the value of the previous debug configuration by at least 3 (61238 in this example)
- Select *3 – Cortex-M4* for [**Access port**]
- Select *None* for [**Reset behaviour**]
- Select *Enable shared ST-LINK*

Figure 15. **ST-LINK GDB server debug configuration (5 of 6)**



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)
- STM32 Cortex-M C/C++ Application
  - NUCLEO-H755-Q_CM4 Debug
  - NUCLEO-H755-Q_CM7 Debug

Filter matched 10 of 10 items

Name: NUCLEO-H755-Q_CM4 Debug

Main | Debugger | Startup | Source | Common

GDB Connection Settings
- (●) Autostart local GDB server — Host name or IP address: localhost
- ( ) Connect to remote GDB server — Port number: 61238

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

GDB Server Command Line Options

Interface
- (●) SWD ( ) JTAG ☐ Use specific ST-LINK S/N — Scan
- Access port: 3 - Cortex-M4

Reset behaviour
- Type: None ☐ Halt all cores

Serial Wire Viewer (SWV)
- ☐ Enable
- Clock Settings
  - Core Clock: 16.0 MHz
  - SWO Clock: 2000 kHz
- Port number: 61235
- ☑ Wait for sync packet

Cross Trigger Interface (CTI)
- ☐ Allow other cores to halt this core ☐ Signal halt events to other cores

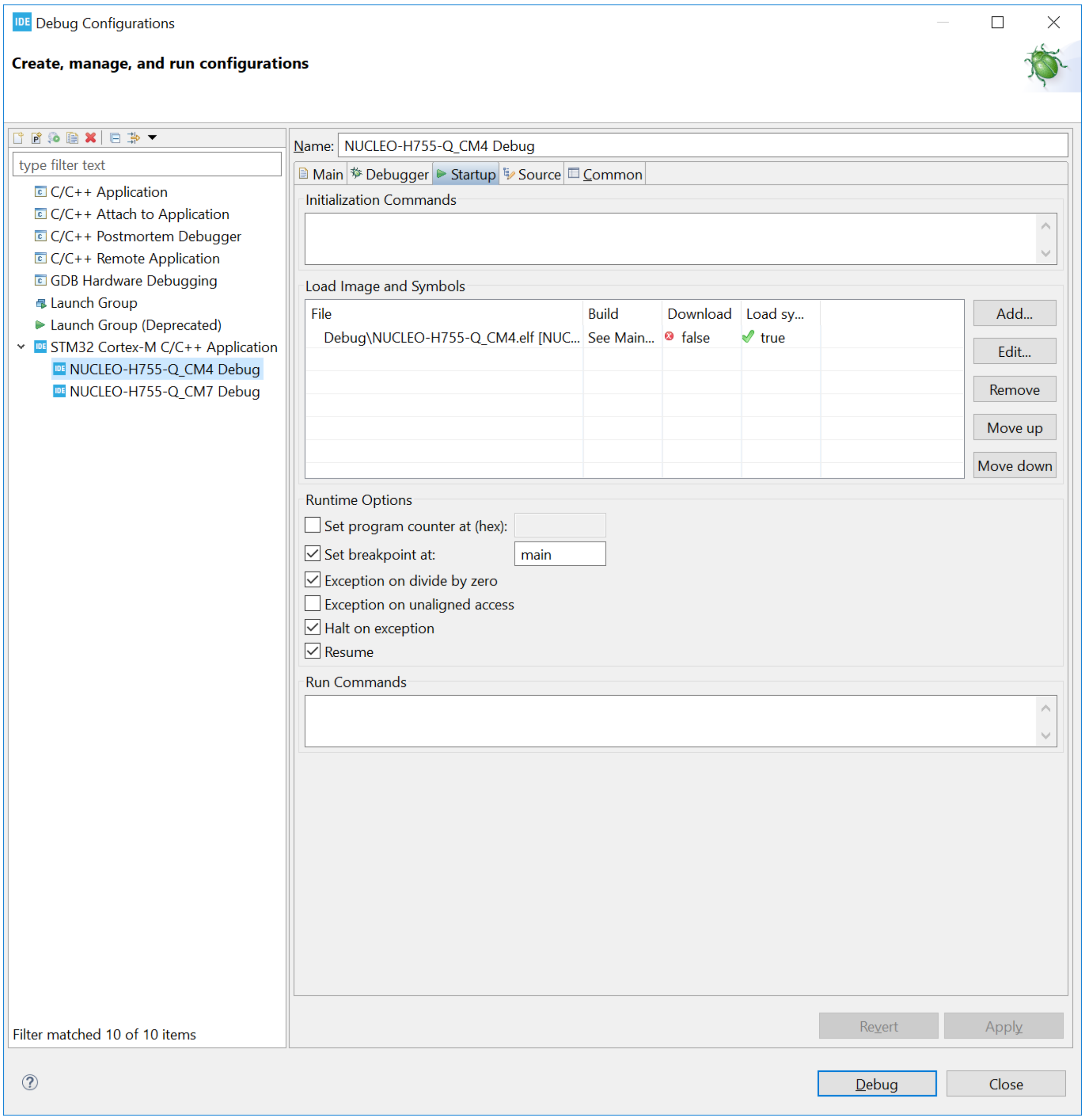
Misc
- ☑ Verify flash download
- ☑ Enable live expressions
- ☐ Log to file: C:\Users\[redacted]\STM32CubeIDE\workspace\NUCLEO-H755-Q\CM4\Debug\ — Browse...
- ☐ External Loader: — Scan
- ☑ Enable shared ST-LINK

Revert | Apply

Debug | Close

Go to the *Startup* tab and select [**Edit…**]>[**Disable Download**]. This is required since the download is already performed by the Cortex®-M7 configuration (refer to Figure 16).

Figure 16. ST-LINK GDB server debug configuration (6 of 6)



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)
- STM32 Cortex-M C/C++ Application
  - NUCLEO-H755-Q_CM4 Debug
  - NUCLEO-H755-Q_CM7 Debug

Filter matched 10 of 10 items

Name: NUCLEO-H755-Q_CM4 Debug

Main | Debugger | Startup | Source | Common

Initialization Commands

Load Image and Symbols

File	Build	Download	Load sy...
Debug\NUCLEO-H755-Q_CM4.elf [NUC...	See Main...	✗ false	✓ true

Add... | Edit... | Remove | Move up | Move down

Runtime Options

- [ ] Set program counter at (hex):
- [x] Set breakpoint at: main
- [x] Exception on divide by zero
- [ ] Exception on unaligned access
- [x] Halt on exception
- [x] Resume

Run Commands

Revert | Apply

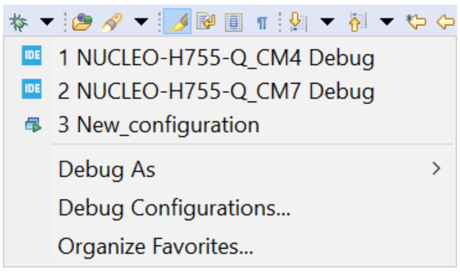
Debug | Close

The configuration is complete.

3.1.1 Launching the configurations

1. Launch the Cortex®-M7 configuration to download both the Cortex®-M4 and Cortex®-M7 images
2. Set the Cortex®-M7 core running so that the HSEM semaphore is released and Cortex®-M4 wakes up from Stop mode
3. Launch the Cortex®-M4 configuration using the arrow next to the debug icon. It is in the running mode and the user can halt it.

Figure 17. **ST-LINK GDB server debug configuration launch**

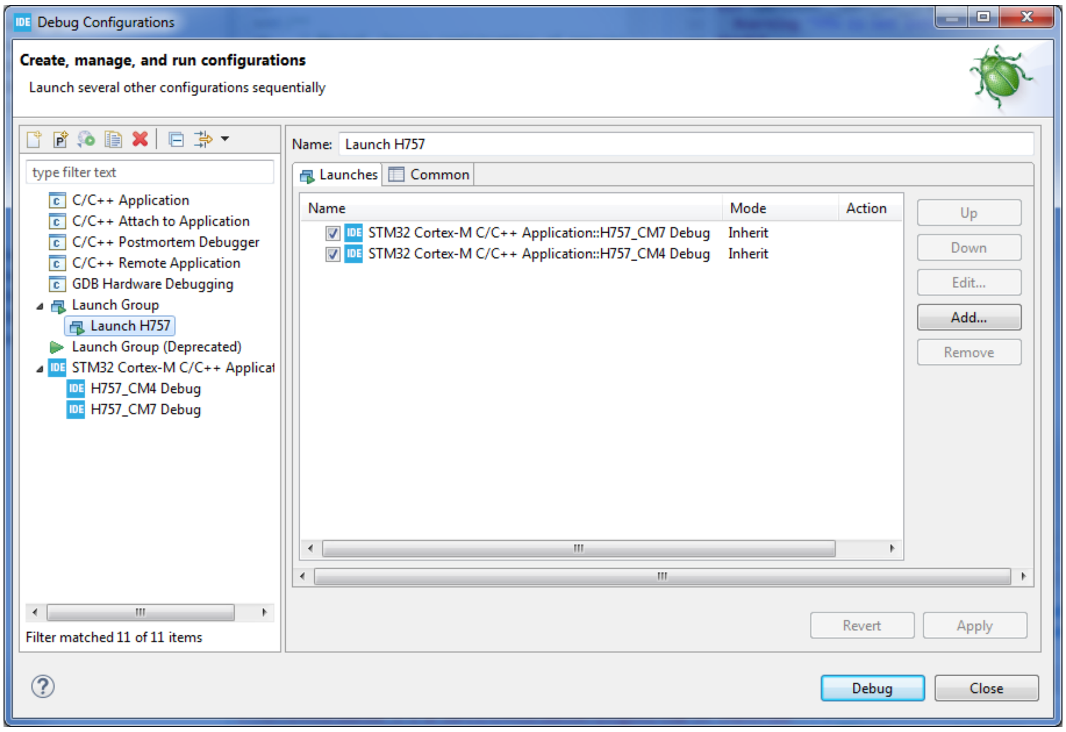


Note: *After creating the debug configurations for both cores, they are not shown in the scroll-down menu if they have never been launched before. This is because the arrow provides access to the history of latest launches, with a grayed "no history" message if there are none. First-time degug launch must be done through the debug configuration wizard.*

When debugging an STM32H7 device with the ST-LINK GDB server, it is possible to create a launch group, which offers the following advantages:

- The user can launch the debug session on both cores with only one launch configuration as shown in Figure 18.

Figure 18. **ST-LINK GDB server launch group**


Debug Configurations
Create, manage, and run configurations
Launch several other configurations sequentially
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
Launch Group
Launch H757
Launch Group (Deprecated)
STM32 Cortex-M C/C++ Applicat
H757_CM4 Debug
H757_CM7 Debug
Filter matched 11 of 11 items
Launches
Common
Up
Down
Edit...
Add...
Remove
Revert
Apply
Debug
Close

Name: Launch H757

Name	Mode	Action
✓ STM32 Cortex-M C/C++ Application::H757_CM7 Debug	Inherit	
✓ STM32 Cortex-M C/C++ Application::H757_CM4 Debug	Inherit	

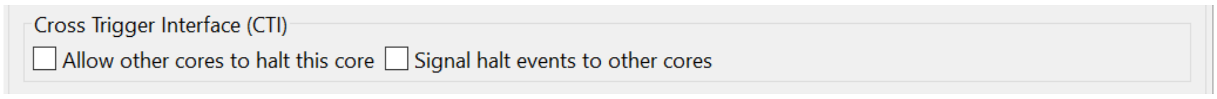
- The user can stop both cores using the launch group. This avoids having to terminate both debug sessions individually.

3.1.2 Cross-trigger Interface

The cross-trigger interface is used to send halt signals from one core to the other. To enable the Cortex®-M4 to halt the Cortex®-M7, apply the following configuration:

- In the Cortex®-M4 debug configuration: select *Signal halt events to other cores*
- In the Cortex®-M7 debug configuration: select *Allow other cores to halt this core*

Figure 19. ST-LINK GDB server debug cross-trigger interface



Cross Trigger Interface (CTI)
☐ Allow other cores to halt this core ☐ Signal halt events to other cores

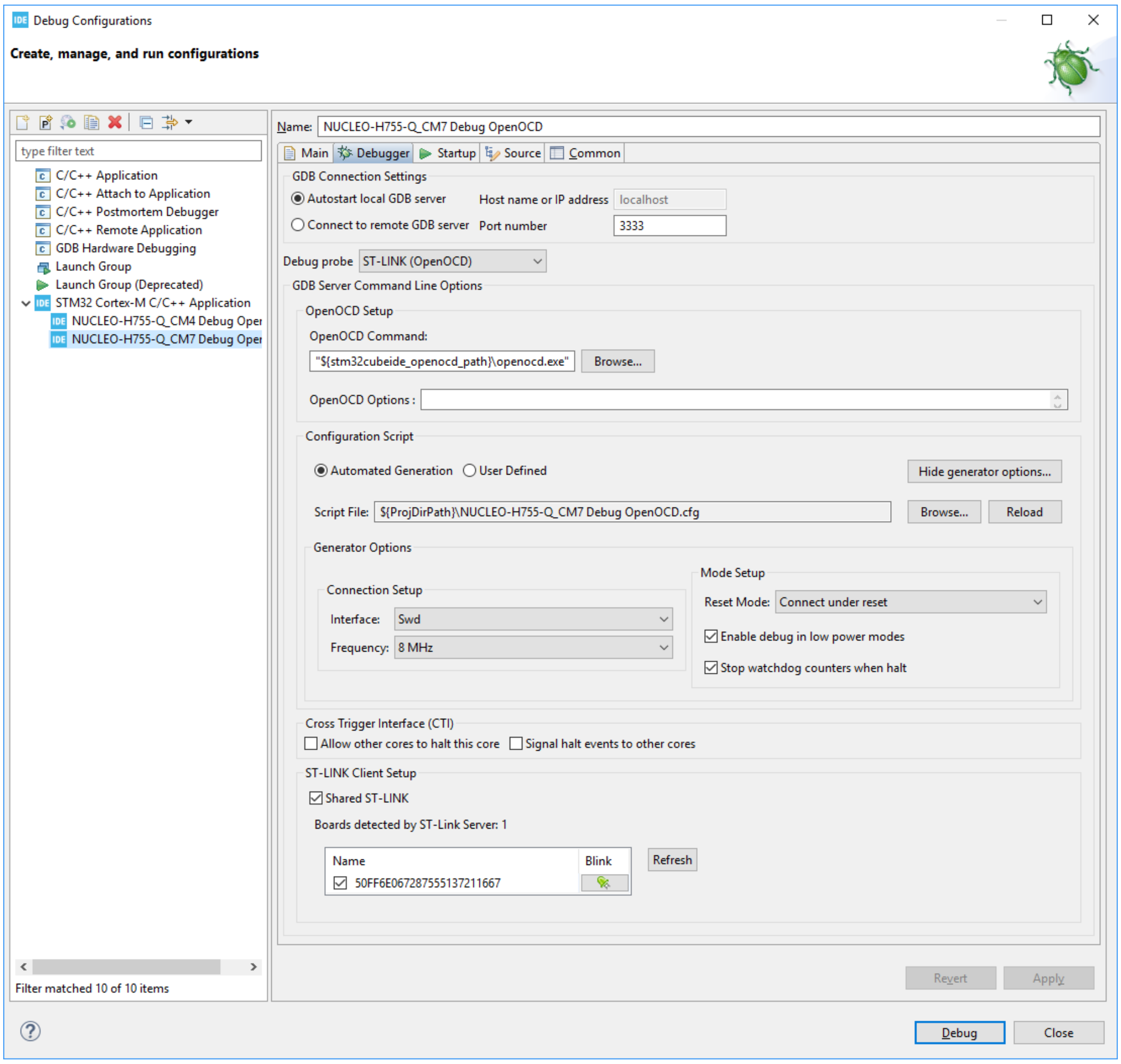
Note: Checking both checkboxes in both debug configurations enables both cores to halt each other.

3.2 Setting up with OpenOCD

Select *ST-LINK (OpenOCD)* as the [**Debug probe**]. Select *Autostart local GDB server* for the configuration that launches first, which is the Cortex®-M7 in the example in Figure 20. Set all the default options and verify that:

- *Connect under reset* is selected as [**Reset Mode**].
- [**Shared ST-LINK**] is selected; this option is mandatory to run the multicore target.

Figure 20. **OpenOCD debug configuration (2 of 3)**



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- Launch Group (Deprecated)
- STM32 Cortex-M C/C++ Application
  - NUCLEO-H755-Q_CM4 Debug Oper
  - NUCLEO-H755-Q_CM7 Debug Oper

Filter matched 10 of 10 items

Name: NUCLEO-H755-Q_CM7 Debug OpenOCD

Main | Debugger | Startup | Source | Common

GDB Connection Settings
- (●) Autostart local GDB server — Host name or IP address: localhost
- ( ) Connect to remote GDB server — Port number: 3333

Debug probe: ST-LINK (OpenOCD)

GDB Server Command Line Options

OpenOCD Setup
OpenOCD Command: "${stm32cubeide_openocd_path}\openocd.exe" [Browse...]
OpenOCD Options :

Configuration Script
(●) Automated Generation ( ) User Defined [Hide generator options...]
Script File: ${ProjDirPath}\NUCLEO-H755-Q_CM7 Debug OpenOCD.cfg [Browse...] [Reload]

Generator Options

Connection Setup
- Interface: Swd
- Frequency: 8 MHz

Mode Setup
- Reset Mode: Connect under reset
- [✓] Enable debug in low power modes
- [✓] Stop watchdog counters when halt

Cross Trigger Interface (CTI)
- [ ] Allow other cores to halt this core
- [ ] Signal halt events to other cores

ST-LINK Client Setup
- [✓] Shared ST-LINK

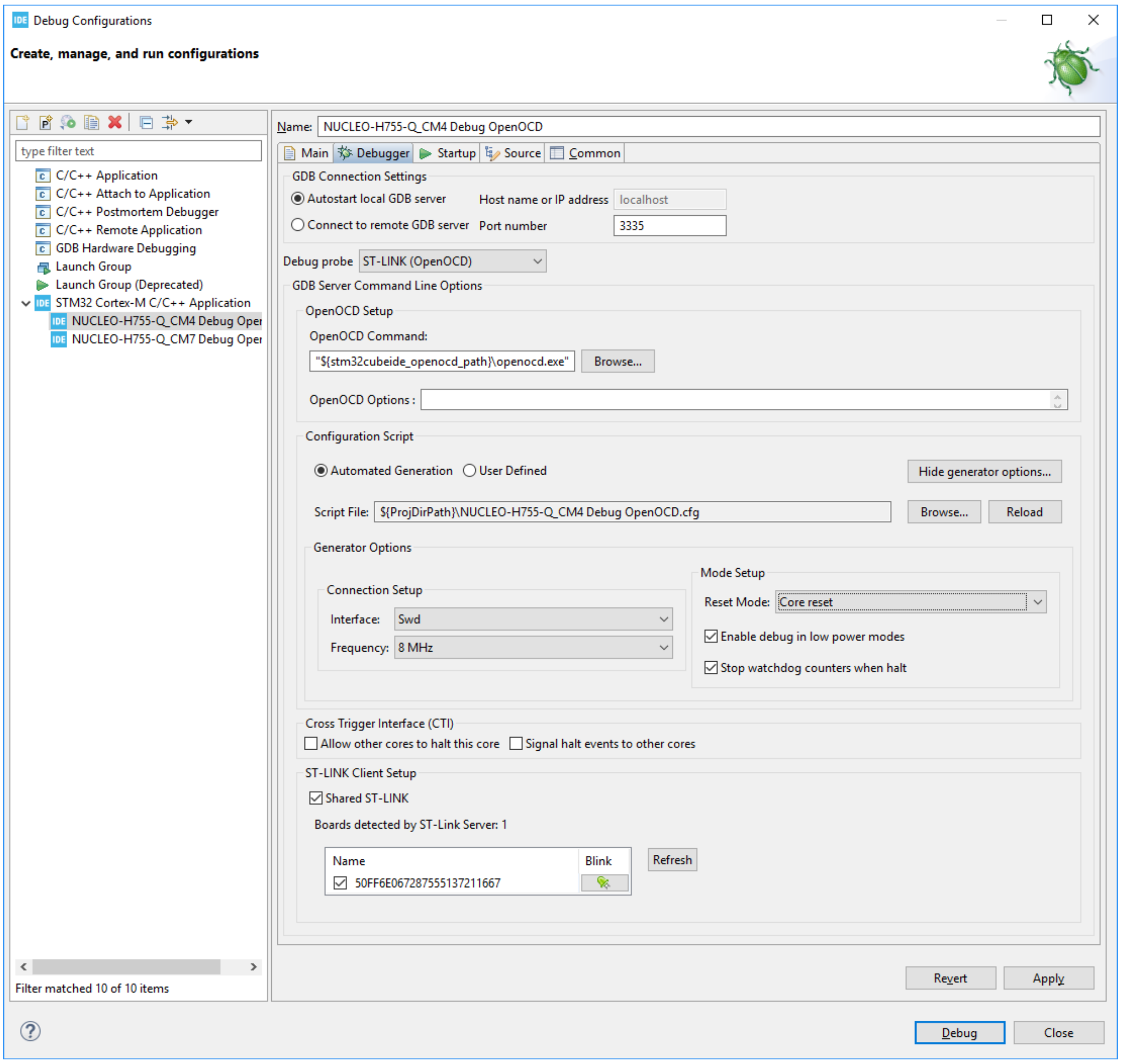
Boards detected by ST-Link Server: 1

Name	Blink
[✓] 50FF6E067287555137211667	

Create the debug configuration for the other core, which is the Cortex$^{®}$-M4 in the example in Figure 21:

- Select *ST-LINK (OpenOCD)* as the [**Debug probe**]
- Select *Autostart local GDB server* as default
- Make sure that the *Port number* exceeds the value of the previous debug configuration by at least 2 (3335 in this example)
- Open [**Generator Options**] and select *Core reset* as [**Reset Mode**]

Figure 21. **OpenOCD debug configuration (3 of 3)**


Debug Configurations
Create, manage, and run configurations

type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
Launch Group
Launch Group (Deprecated)
STM32 Cortex-M C/C++ Application
NUCLEO-H755-Q_CM4 Debug Oper
NUCLEO-H755-Q_CM7 Debug Oper
Filter matched 10 of 10 items

Name: NUCLEO-H755-Q_CM4 Debug OpenOCD
Main | Debugger | Startup | Source | Common

GDB Connection Settings
(●) Autostart local GDB server — Host name or IP address: localhost
( ) Connect to remote GDB server — Port number: 3335

Debug probe: ST-LINK (OpenOCD)

GDB Server Command Line Options
OpenOCD Setup
OpenOCD Command: "${stm32cubeide_openocd_path}\openocd.exe" Browse...
OpenOCD Options:

Configuration Script
(●) Automated Generation ( ) User Defined — Hide generator options...
Script File: ${ProjDirPath}\NUCLEO-H755-Q_CM4 Debug OpenOCD.cfg — Browse... Reload

Generator Options
Connection Setup
Interface: Swd
Frequency: 8 MHz
Mode Setup
Reset Mode: Core reset
☑ Enable debug in low power modes
☑ Stop watchdog counters when halt

Cross Trigger Interface (CTI)
☐ Allow other cores to halt this core ☐ Signal halt events to other cores

ST-LINK Client Setup
☑ Shared ST-LINK
Boards detected by ST-Link Server: 1

Name	Blink
☑ 50FF6E067287555137211667	

Refresh

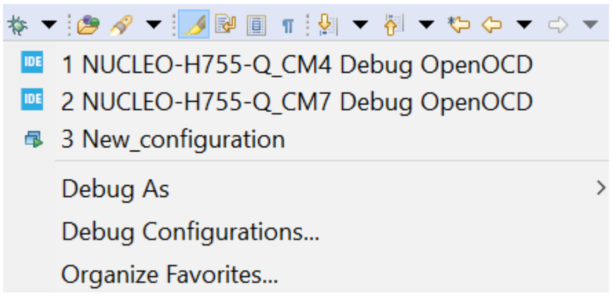
Revert Apply
Debug Close

The configuration of the *Startup* tab is the same as with the ST-LINK GDB server probe for both debug configurations (refer to Startup tab - Cortex-M7 and Startup tab - Cortex-M4 in Section 3.1 Setting up with ST-LINK GDB server).

3.2.1 Launching the configurations

After both debug configurations are set up, save them and click on the arrow next to the debug icon to make sure launching the autostarting local GDB server first (the Cortex®-M7 in this example). After this first launch, click on the same arrow again and launch the other configuration.

Figure 22. **OpenOCD debug configuration launch**

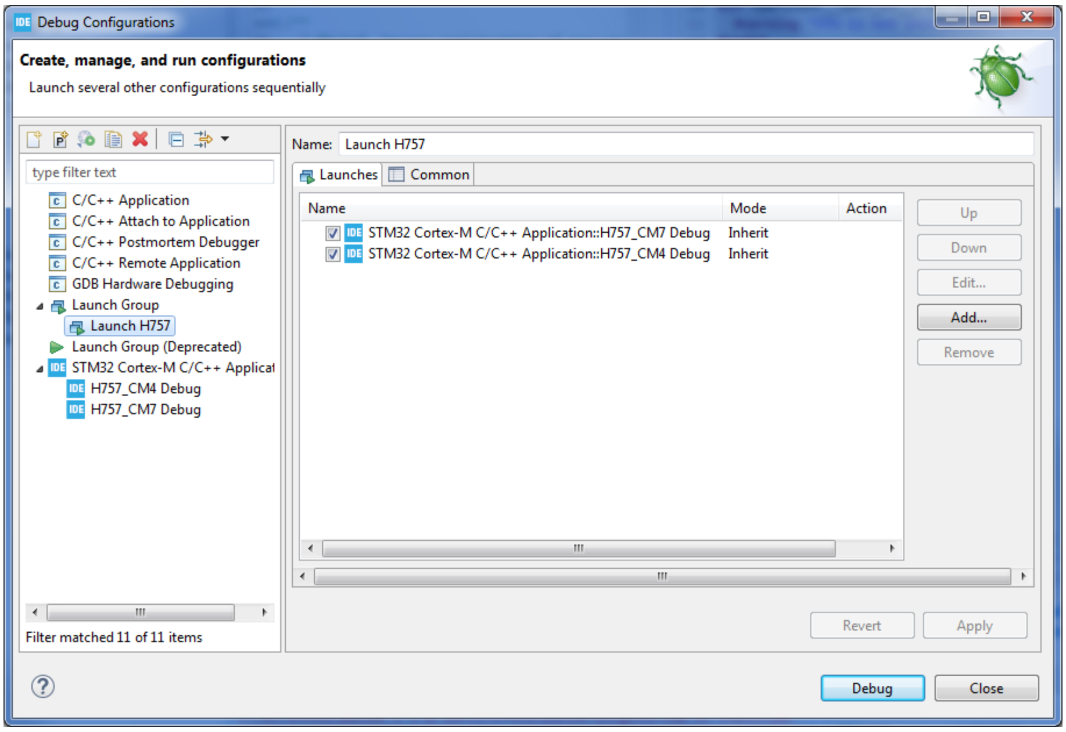


Note: *After creating the debug configurations for both cores, they are not shown in the scroll-down menu if they have never been launched before. This is because the arrow provides access to the history of latest launches, with a grayed "no history" message if there are none. First-time degug launch must be done through the debug configuration wizard.*

When debugging an STM32H7 device with OpenOCD, it ispossible to create a launch group, which offers the following advantages:

- The user can launch the debug session on both cores with only one launch configuration.

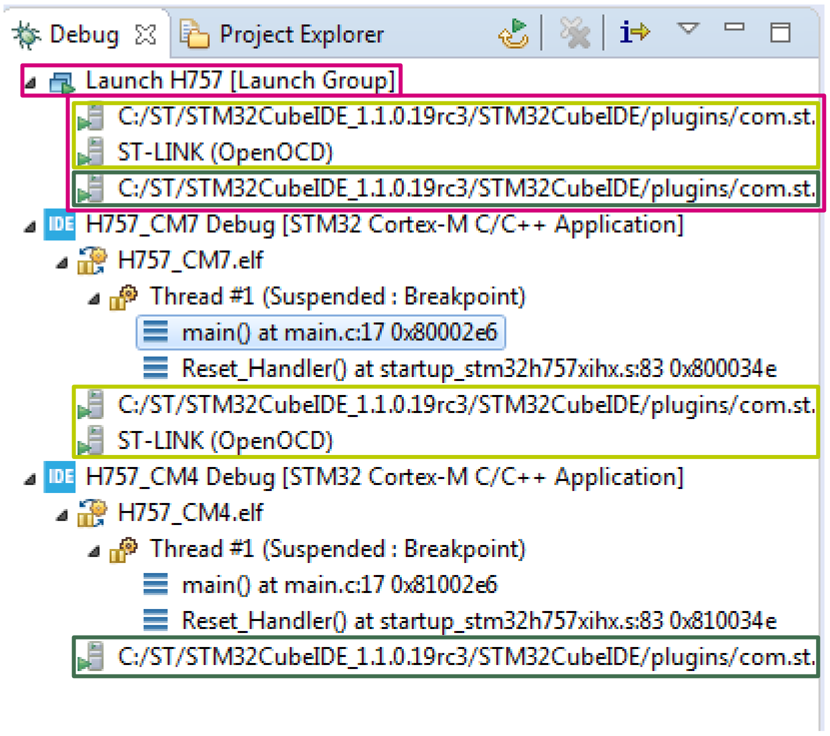
Figure 23. **Launch OpenOCD debug on both cores**



Name	Mode	Action
☑ STM32 Cortex-M C/C++ Application::H757_CM7 Debug	Inherit	
☑ STM32 Cortex-M C/C++ Application::H757_CM4 Debug	Inherit	

- The user can stop both cores using the launch group. This avoids having to manually terminate both debug sessions individually. In Figure 24, the launch group in pink handles each of the cores in green.

Figure 24. Stop OpenOCD debug on both cores



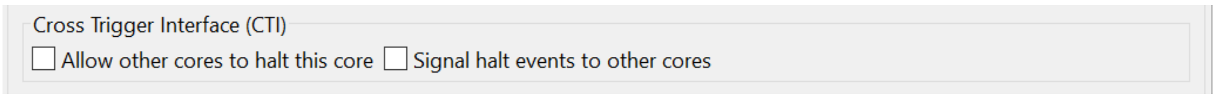
```
Debug  Project Explorer
Launch H757 [Launch Group]
    C:/ST/STM32CubeIDE_1.1.0.19rc3/STM32CubeIDE/plugins/com.st.
    ST-LINK (OpenOCD)
    C:/ST/STM32CubeIDE_1.1.0.19rc3/STM32CubeIDE/plugins/com.st.
H757_CM7 Debug [STM32 Cortex-M C/C++ Application]
  H757_CM7.elf
    Thread #1 (Suspended : Breakpoint)
      main() at main.c:17 0x80002e6
      Reset_Handler() at startup_stm32h757xihx.s:83 0x800034e
    C:/ST/STM32CubeIDE_1.1.0.19rc3/STM32CubeIDE/plugins/com.st.
    ST-LINK (OpenOCD)
H757_CM4 Debug [STM32 Cortex-M C/C++ Application]
  H757_CM4.elf
    Thread #1 (Suspended : Breakpoint)
      main() at main.c:17 0x81002e6
      Reset_Handler() at startup_stm32h757xihx.s:83 0x810034e
    C:/ST/STM32CubeIDE_1.1.0.19rc3/STM32CubeIDE/plugins/com.st.
```

3.2.2 Cross-trigger interface

The cross-trigger interface is used to send halt signals from one core to the other. To enable the Cortex®-M4 to halt the Cortex®-M7, apply the following configuration:

- In the Cortex®-M4 debug configuration: select *Signal halt events to other cores*
- In the Cortex®-M7 debug configuration: select *Allow other cores to halt this core*

Figure 25. OpenOCD debug cross-trigger interface



Cross Trigger Interface (CTI)
☐ Allow other cores to halt this core ☐ Signal halt events to other cores

Note: Checking both checkboxes in both debug configurations enables both cores to halt each other.

Revision history

Table 1. **Document revision history**

Date	Revision	Changes
21-Nov-2019	1	Initial release.
23-Jul-2020	2	Reorganized *Section 3 Debugging*: • Updated *Section 3.1 Setting up with ST-LINK GDB server*, *Section 3.1.1 Launching the configurations*, and added *Figure 18. STLINK GDB server launch group* • Updated *Section 3.2 Setting up with OpenOCD*, *Figure 20. OpenOCD debug configuration (2 of 3)*, *Figure 21. OpenOCD debug configuration (3 of 3)* and *Section 3.2.2 Cross-trigger interface*
10-Jun-2021	3	Updated *Figure 11* in *Section 3 Debugging*.
9-Feb-2022	4	Updated the Figure 12 and its introduction in Section 3.1 Setting up with ST-LINK GDB server.

Contents

List of figures