AN6265

Application note

Getting started with STM32N6 MCUs in STM32CubeIDE

Introduction

This application note describes how to get started with projects based on STM32N6 microcontrollers in the STMicroelectronics STM32CubeIDE integrated development environment.

Table 1. Applicable products

Type	Products
Microcontrollers	STM32N6 series
Software development tool	STM32CubeIDE

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 STM32N6 microcontrollers:

- STM32CubeIDE 1.17.0 or newer
- STM32Cube_FW_N6_V1.0.0 or newer
- STM32CubeMX 6.13.0 or newer

Users are advised to keep updated with the documentation evolution of the STM32N6 microcontrollers at www.st.com/en/microcontrollers-microprocessors/stm32n6-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 STM32N6 microcontrollers. 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 want to learn with and explore example projects:
  Refer to Section 2.3: Import a project from the STM32CubeN6 MCU Package
- I want to start a first STM32N6 project:
  - Empty project – No STM32CubeMX support for maximum flexibility.
    Refer to Section 2.2: Create an empty project based on the template in the STM32CubeN6 MCU Package
  - STM32CubeN6 project – STM32CubeMX-managed project.
    Refer to Section 2.1: Create a new STM32 project

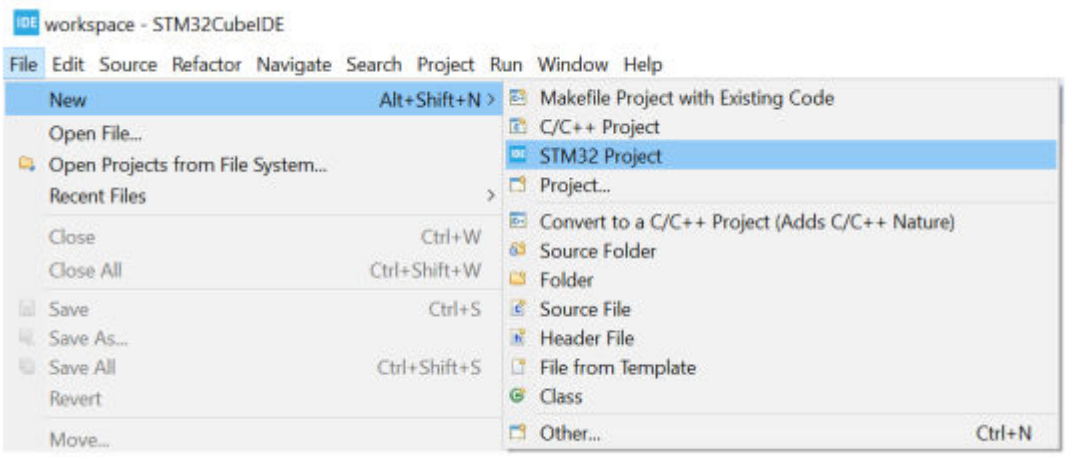
2 Create and import projects

This chapter describes how to create or import projects for STM32N6 microcontrollers.

2.1 Create a new STM32 project

To start a new project, go to [**File**]>[**New**]>[**STM32 Project**] as shown in Figure 1.

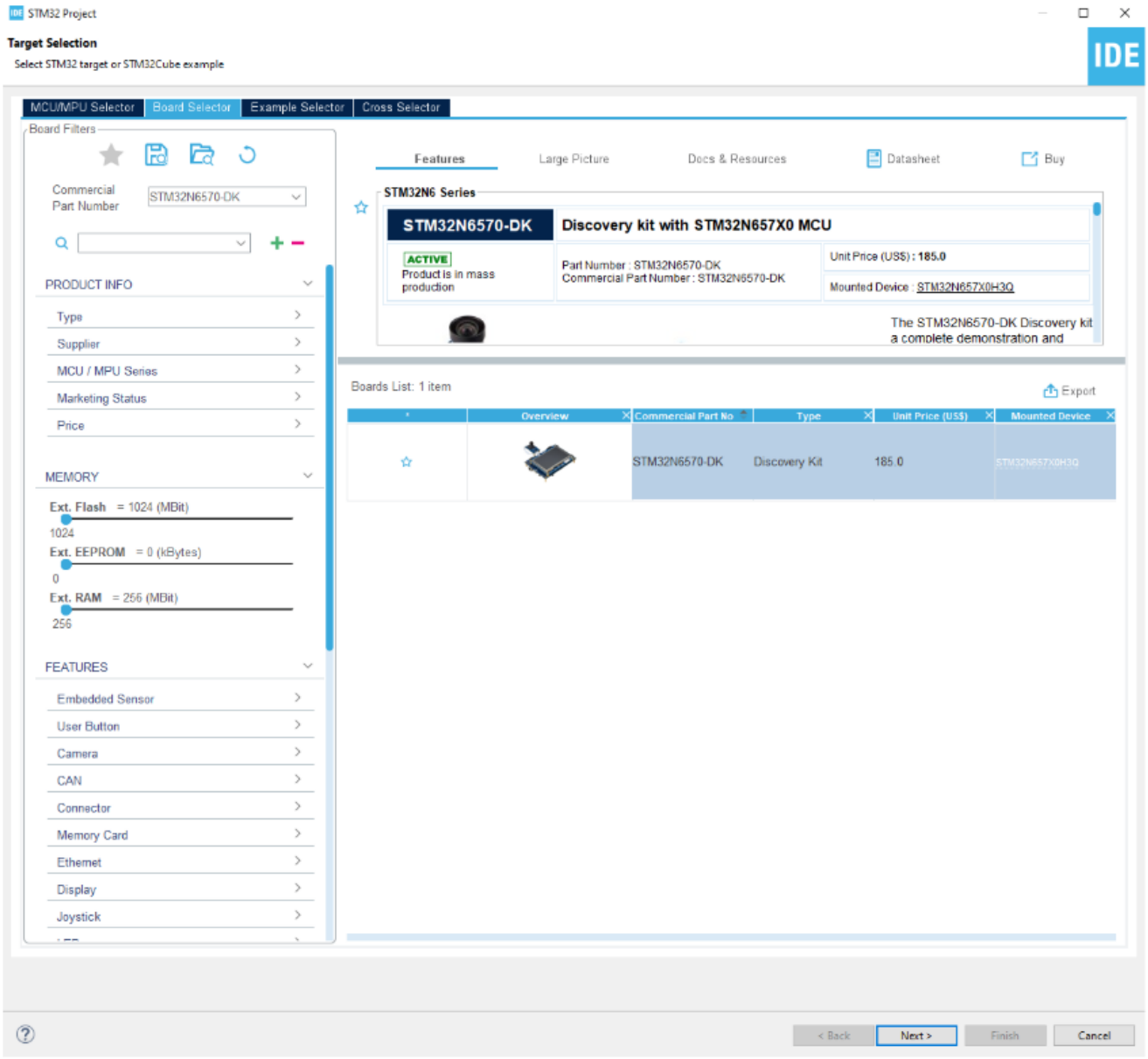
Figure 1. New STM32 project



DT76526V1

Select the desired MCU or board. In the example illustrated in Figure 2, the selected board is the STM32N6570-DK. Click on [**Next >**].

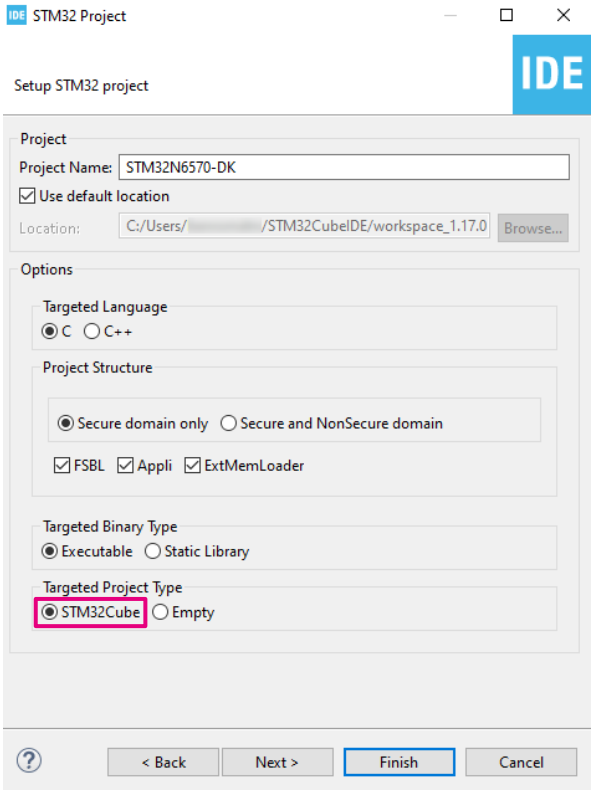
Figure 2. Target selection



DT76527V1

After the target selection comes the project setup step shown in Figure 3. 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 3. Project setup (STM32CubeMX-managed project creation)



DT76528V1

The created project contains up to three subprojects:

- STM32N6570-DK_FSBL: dedicated to the bootloader development.
- STM32N6570-DK_Appli: dedicated to the main application development.
- STM32N6570-DK_ExtMemLoader (optional and not generated by default): to be used when the user needs an own external flash loader.

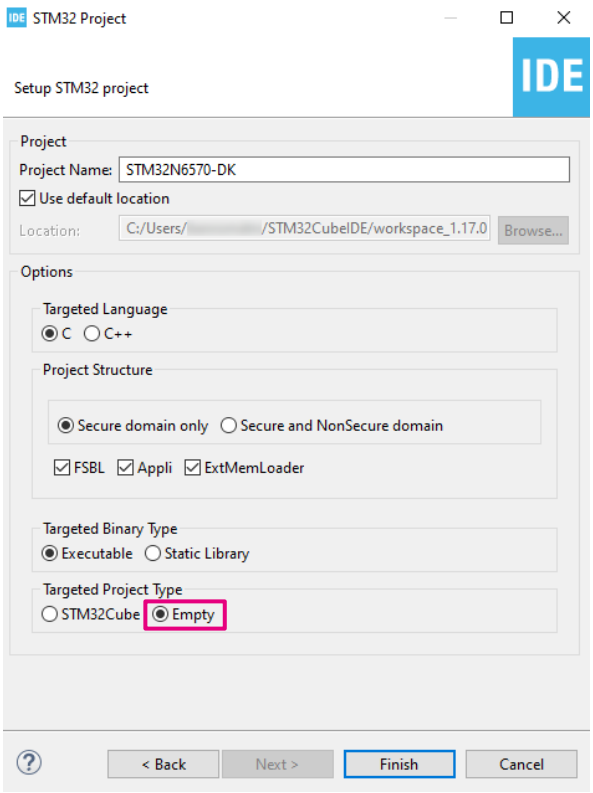
2.2 Create an empty project based on the template in the STM32CubeN6 MCU Package

To start a new project, go to [**File**]>[**New**]>[**STM32 Project**] as shown in Figure 1. New STM32 project.

Select the desired MCU or board. In the example illustrated in Figure 2. Target selection, the selected board is the STM32N6570-DK. Click on [**Next >**].

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. Project setup (empty project creation)



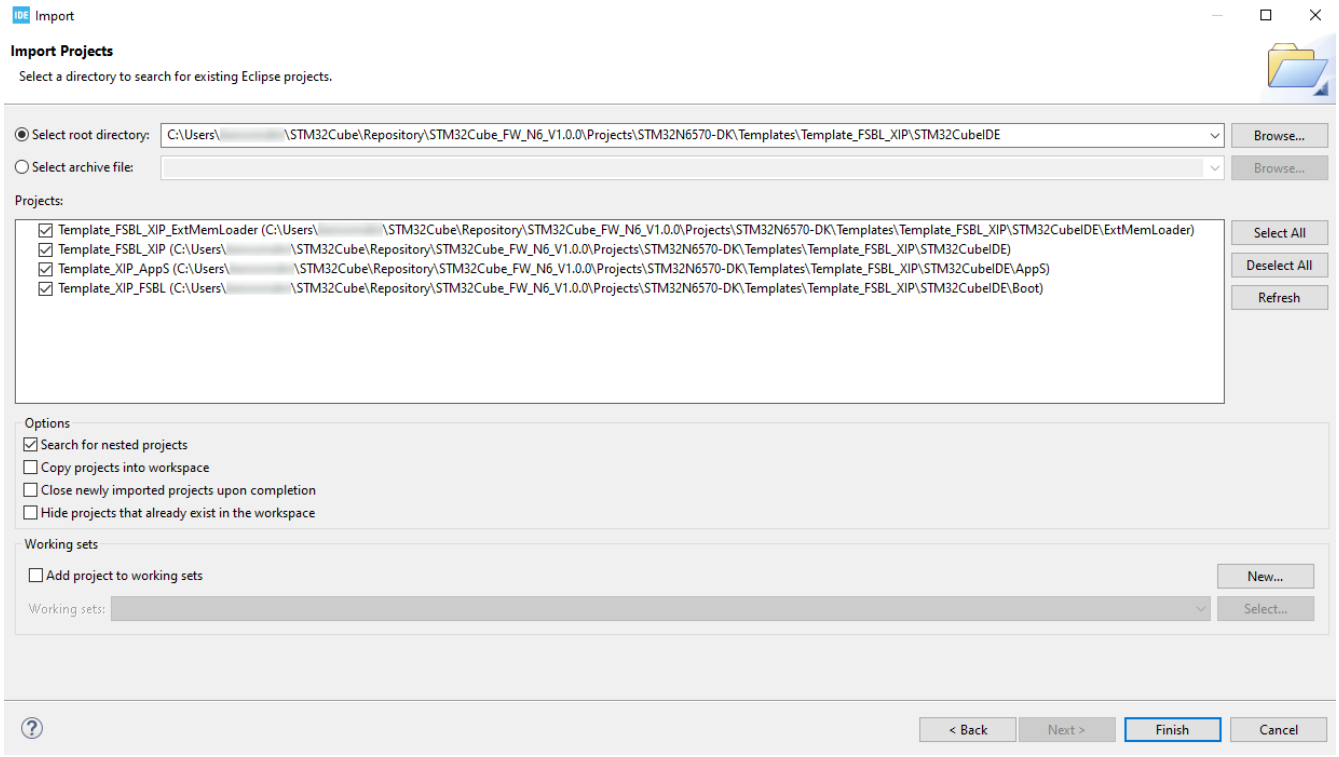
DT76529V1

2.3 Import a project from the STM32CubeN6 MCU Package

To import the STM32Cube firmware project into STM32CubeIDE, go to [**File**]>[**Open Projects from File System…**].

Then, select the corresponding project. A project example is by default located at `$Home\STM32Cube\Repository\STM32Cube_FW_N6_V1.0.0\Projects\STM32N6570-DK\Templates\Template_FSBL_XIP\STM32CubeIDE`.

Figure 5. Firmware project selection



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

3 Generate the trusted library

The user must load a signed binary file of the application into the memory. The signing operation is performed using the `STM32_SigningTool_CLI.exe`, which is included in the STM32CubeProgrammer (STM32CubeProg) package.

The user can access this signing tool in one of two ways:

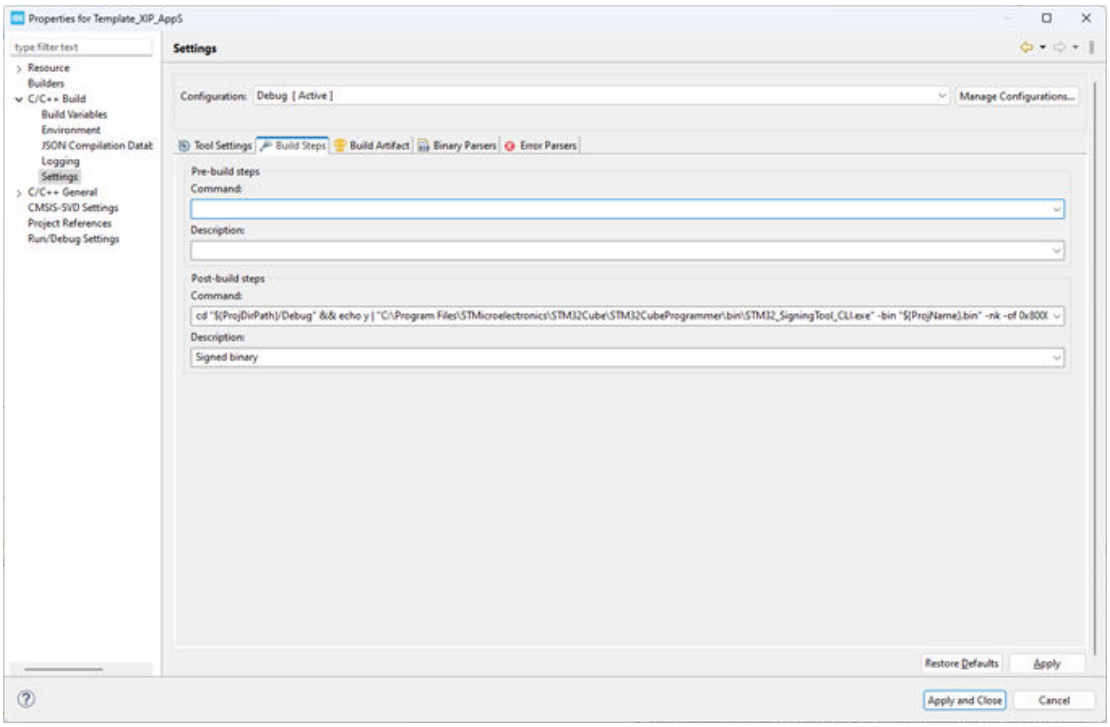
- Under the STM32CubeIDE installation path (Windows® example): `C:\ST\STM32CubeIDE_1.17.0\STM32CubeIDE\plugins\com.st.stm32cube.ide.mcu.externaltools.cubeprogrammer.win32_2.2.0.20240917 0845\tools\bin\`
- Under the STM32CubeProgrammer path (Windows® example): `C:\Program Files\STMicroelectronics \STM32Cube\STM32CubeProgrammer\bin`

To sign the binary file, the user must launch the following command: `STM32_SigningTool_CLI.exe -bin Template_XIP_AppS.bin -nk -of 0x80000000 -t fsbl -o Project-trusted.bin -hv 2.3 -dump Project-trusted.bin`

It is also possible to perform the signing step from within the STM32CubeIDE tool using the postbuild command option as described below:

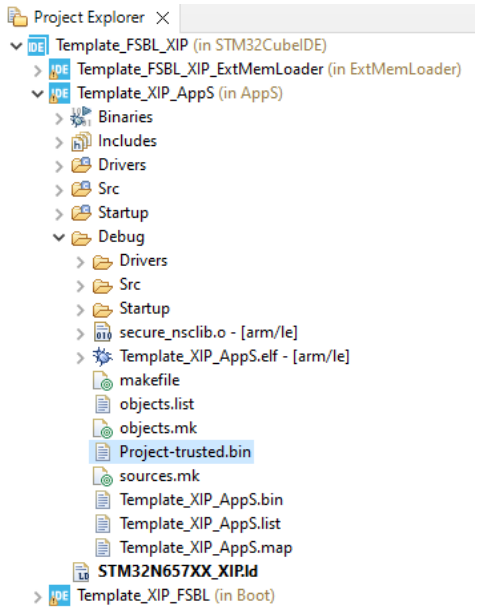
- Right-click on `Template_XIP_AppS(in Apps)`, then go to [**Properties**]>[**C/C++ Build**]>[**Settings**]>[**Build Steps**]
- Type the following in the [**Command**] field of the *Post-build steps* section: `cd "${ProjDirPath}/Debug" && echo y | "C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer\bin\STM32_SigningTool_CLI.exe" -bin "${ProjName}.bin" -nk -of 0x80000000 -t fsbl -o "${ProjName}-Trusted.bin" -hv 2.3 -dump "${ProjName}-Trusted.bin"`

Figure 6. Postbuild signing command



- Click on [**Apply and Close**], then build the Template_XIP_AppS to generate the `Project-trusted.bin` file as shown in Figure 7

Figure 7. Signed binary creation



DT76532V1

4 Debugging

This chapter highlights some of the points to bear in mind while debugging an STM32N6 microcontroller. In the next two sections, this application note covers the configurations needed to start debug sessions with ST-LINK GDB server and OpenOCD.

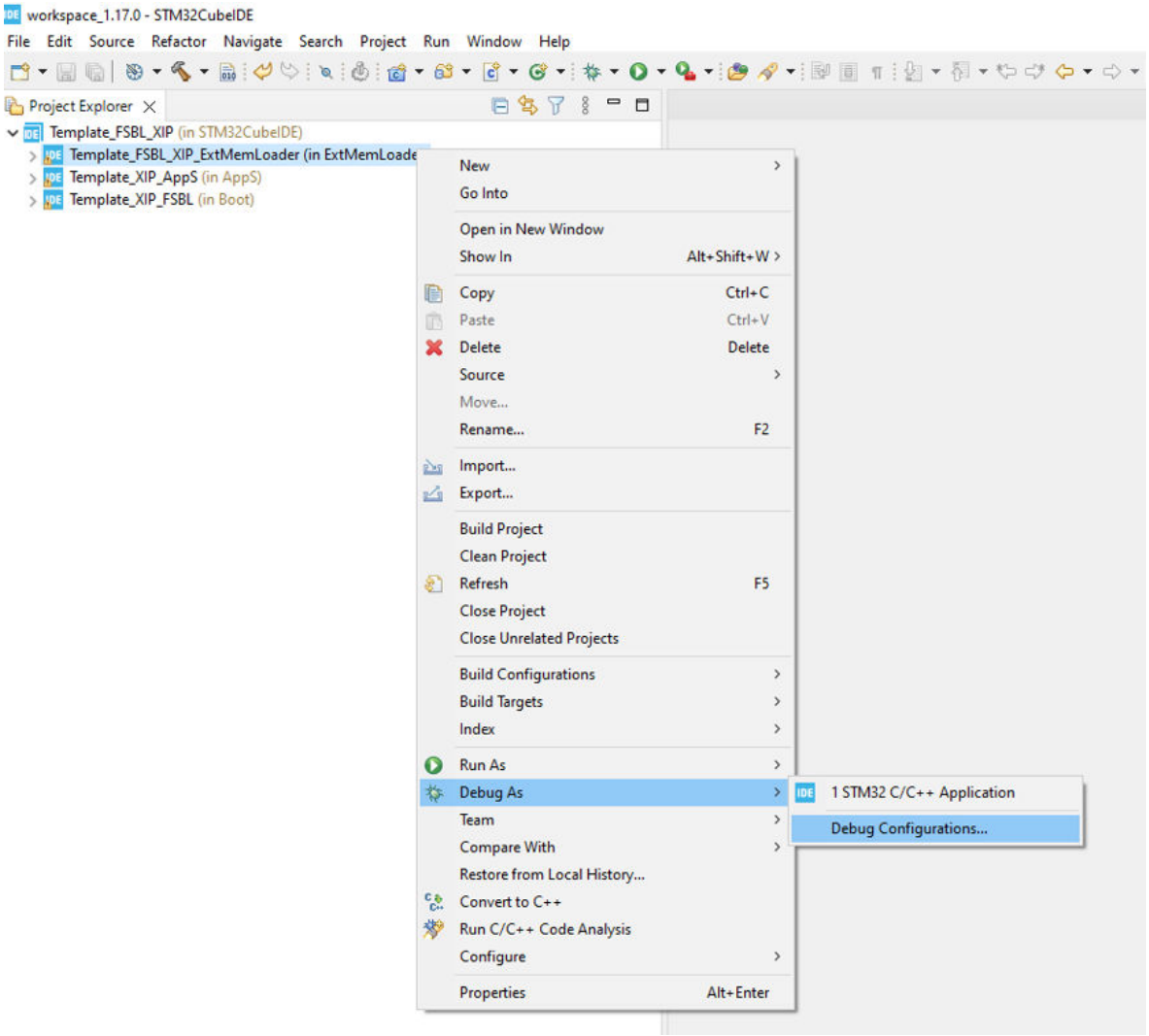
Note: For the rest of this chapter, examples are based on the STM32N6570-DK board.

4.1 Setting up with ST-LINK GDB server

4.1.1 Debugging the ExtMemLoader

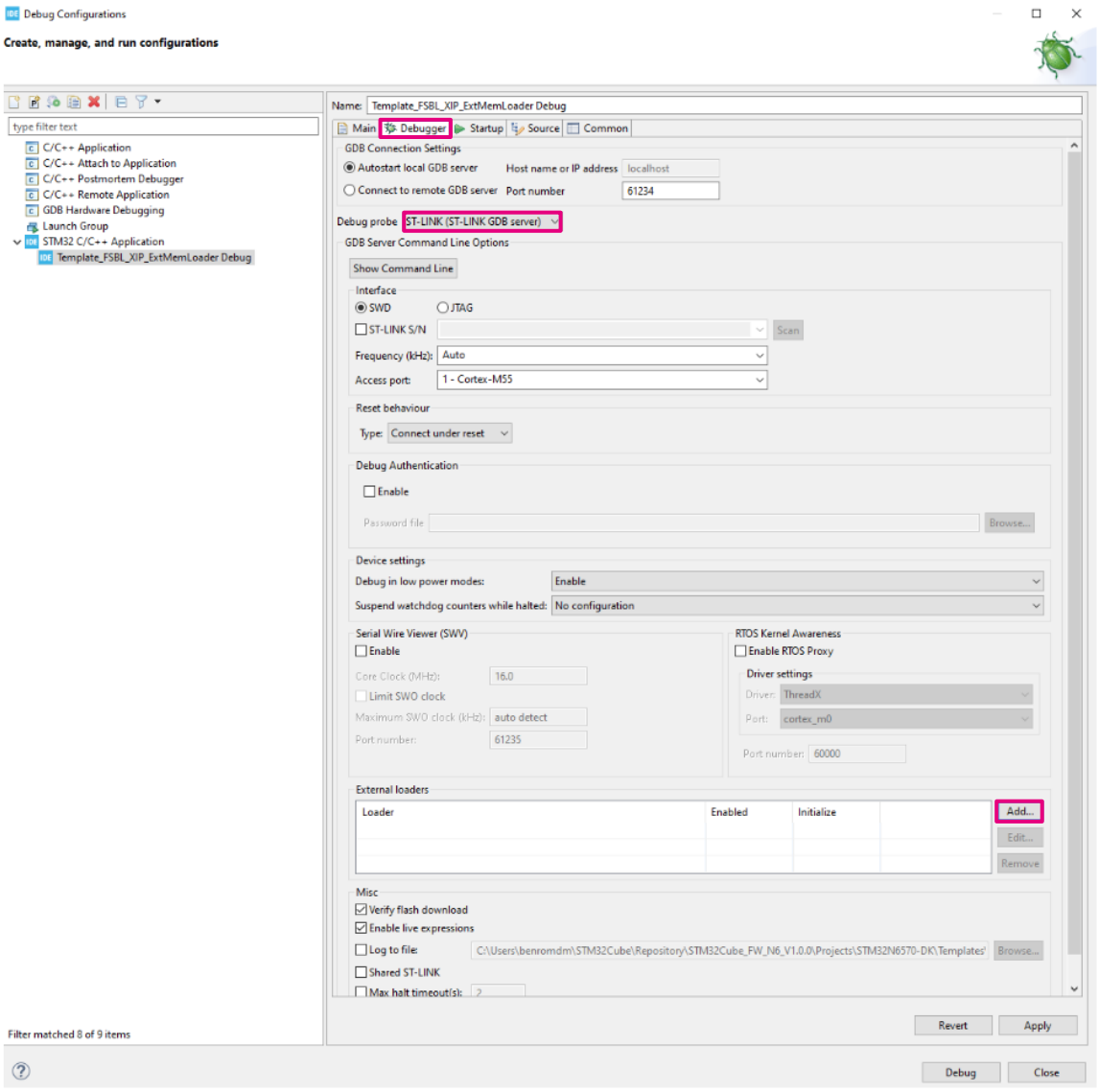
To start configuring the launch of the debug session, right-click the subproject (in this example, the Template_FSBL_XIP_ExtMemLoader) and select [**Debug As**]>[**Debug Configurations...**] as shown in Figure 8.

Figure 8. ST-LINK GDB server debug configuration - ExtMemLoader (1 of 3)



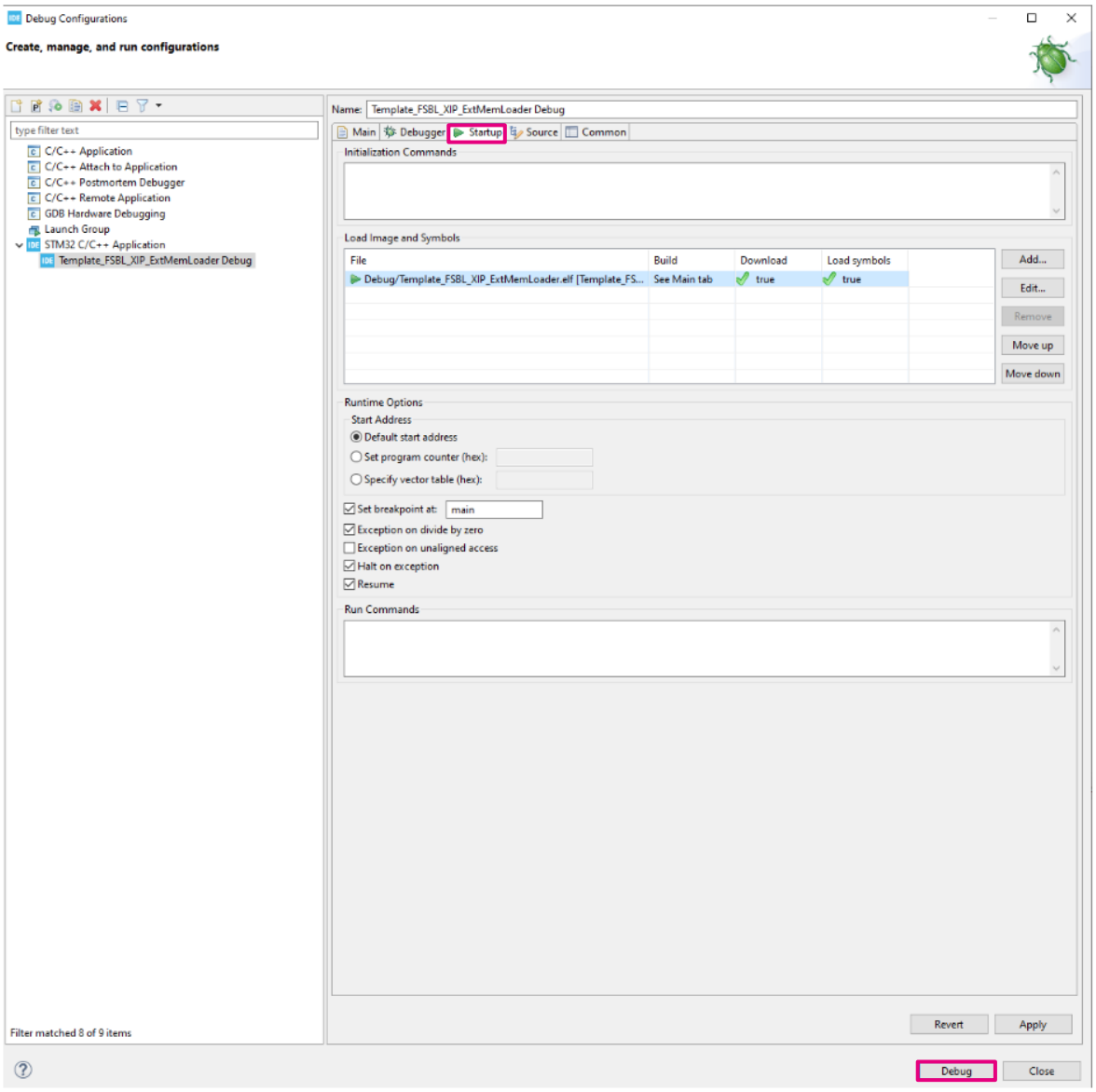
Then, in the *Debugger* tab, click [**Add…**] to select Template_FSBL_XIP_ExtMemLoader, and switch to the *Startup* tab to finish the configuration as shown in Figure 9 and Figure 10.

Figure 9. ST-LINK GDB server debug configuration - ExtMemLoader (2 of 3)



DT76544V1

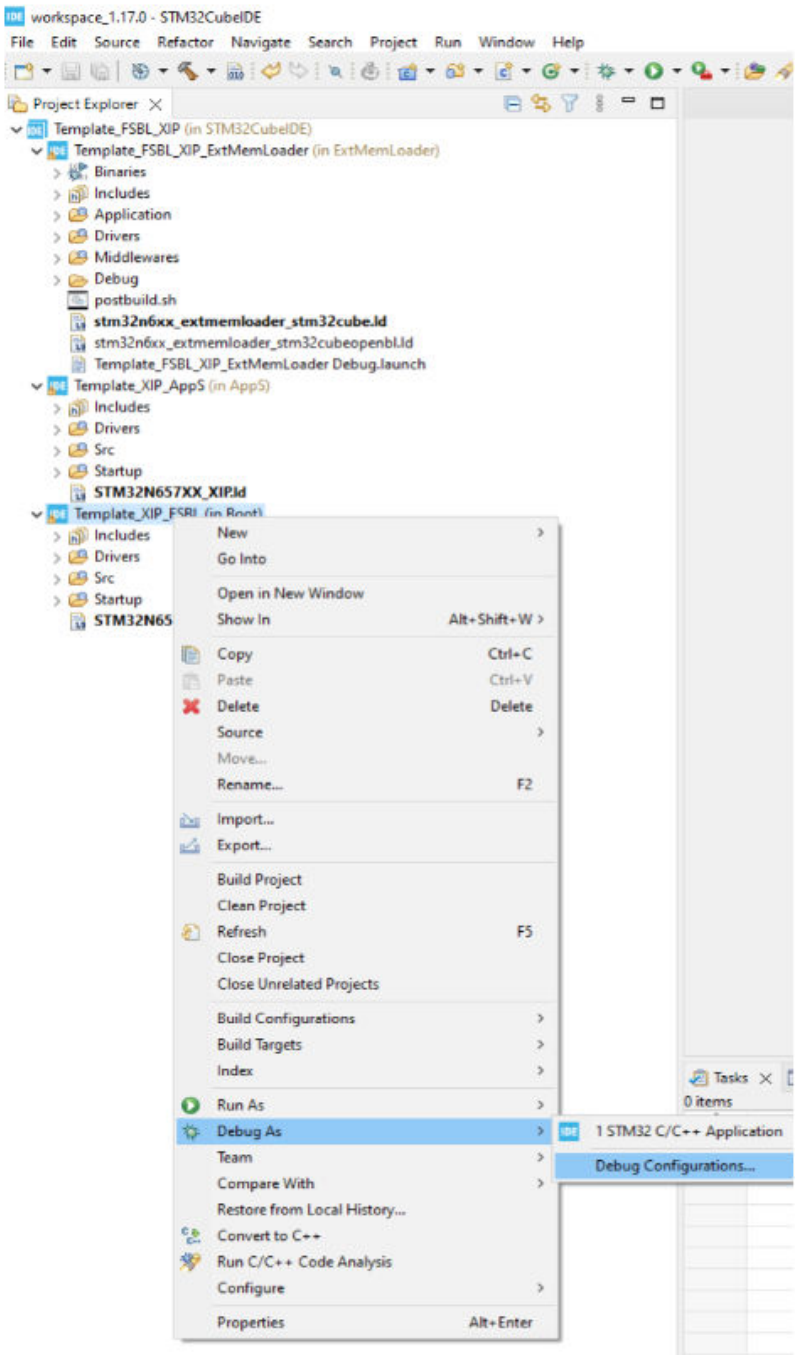
Figure 10. **ST-LINK GDB server debug configuration - ExtMemLoader (3 of 3)**



4.1.2 Debugging the FSBL

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

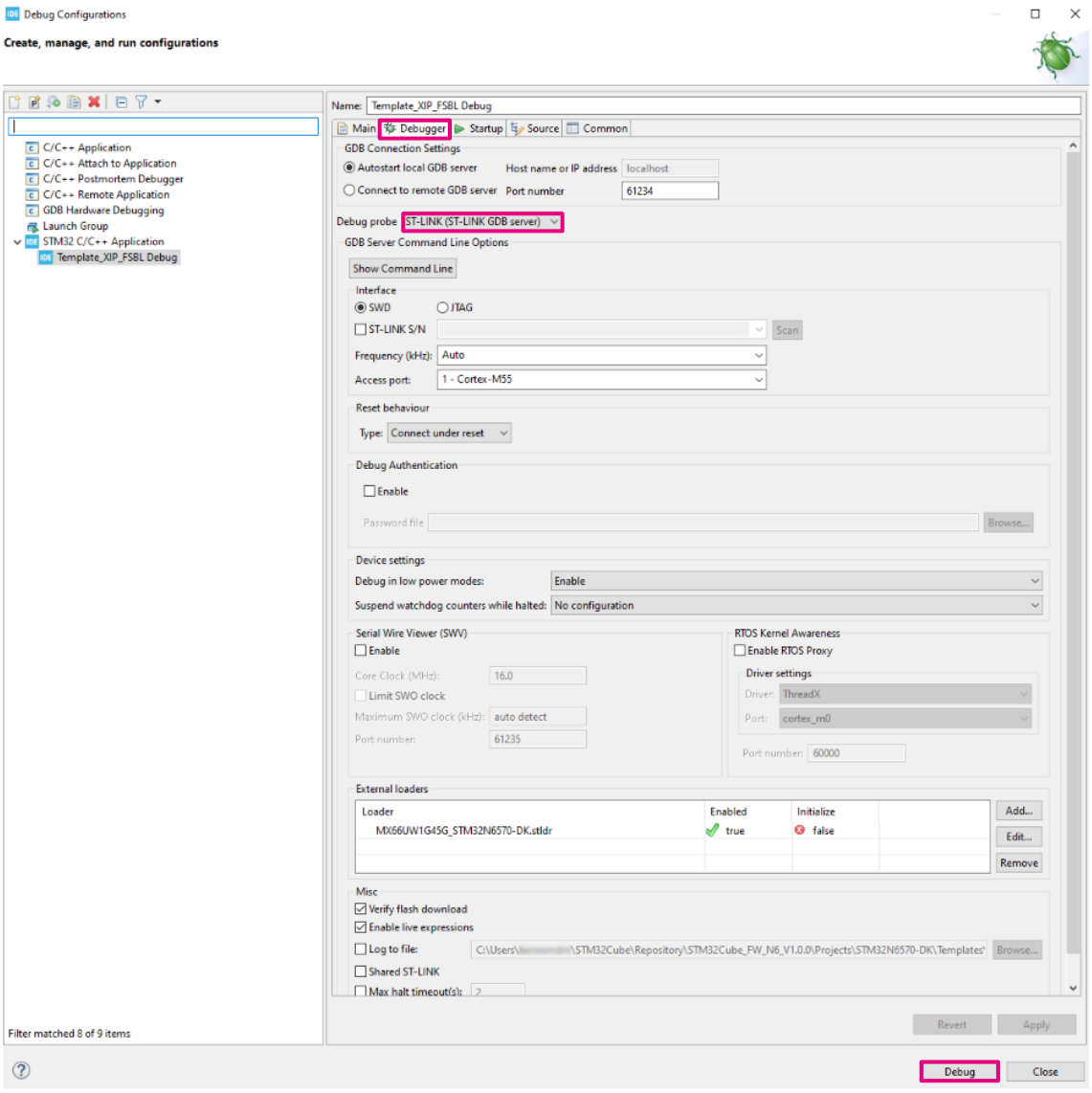
Figure 11. ST-LINK GDB server debug configuration - FSBL (1 of 3)



DT76546V1

By default, the internal RAM is used to load the FSBL. However, because the STM32N6 microcontroller is a flashless device, the external memory must be used to load the program binary. Multiple linker files are generated for use according to the memory type desired. If the external memory is used, an external flash loader must be set in the debug configuration as shown in Figure 12.

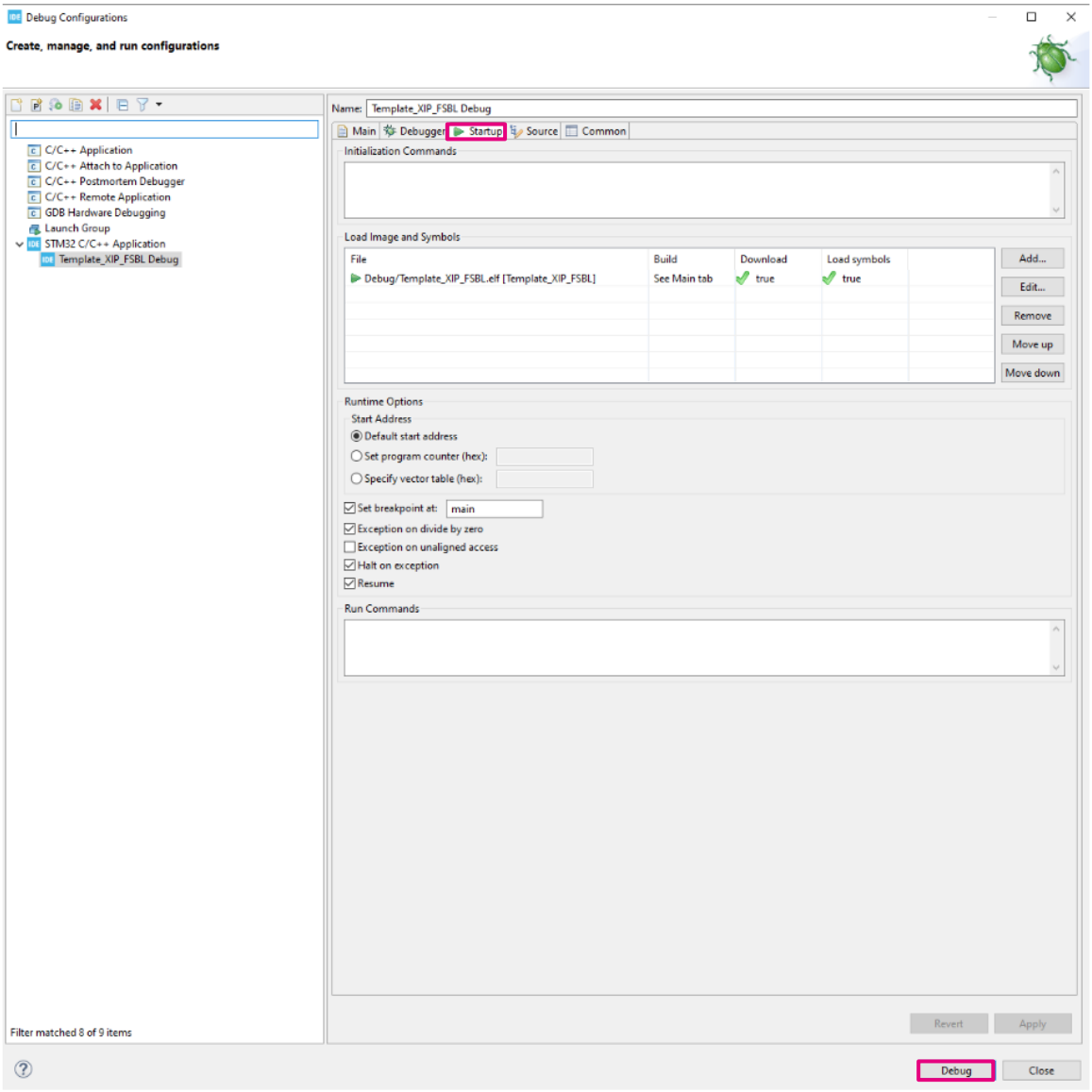
Figure 12. ST-LINK GDB server debug configuration - FSBL (2 of 3)



DT76547V1

Then, switch to the *Startup* tab to finish the configuration as shown in Figure 13.

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

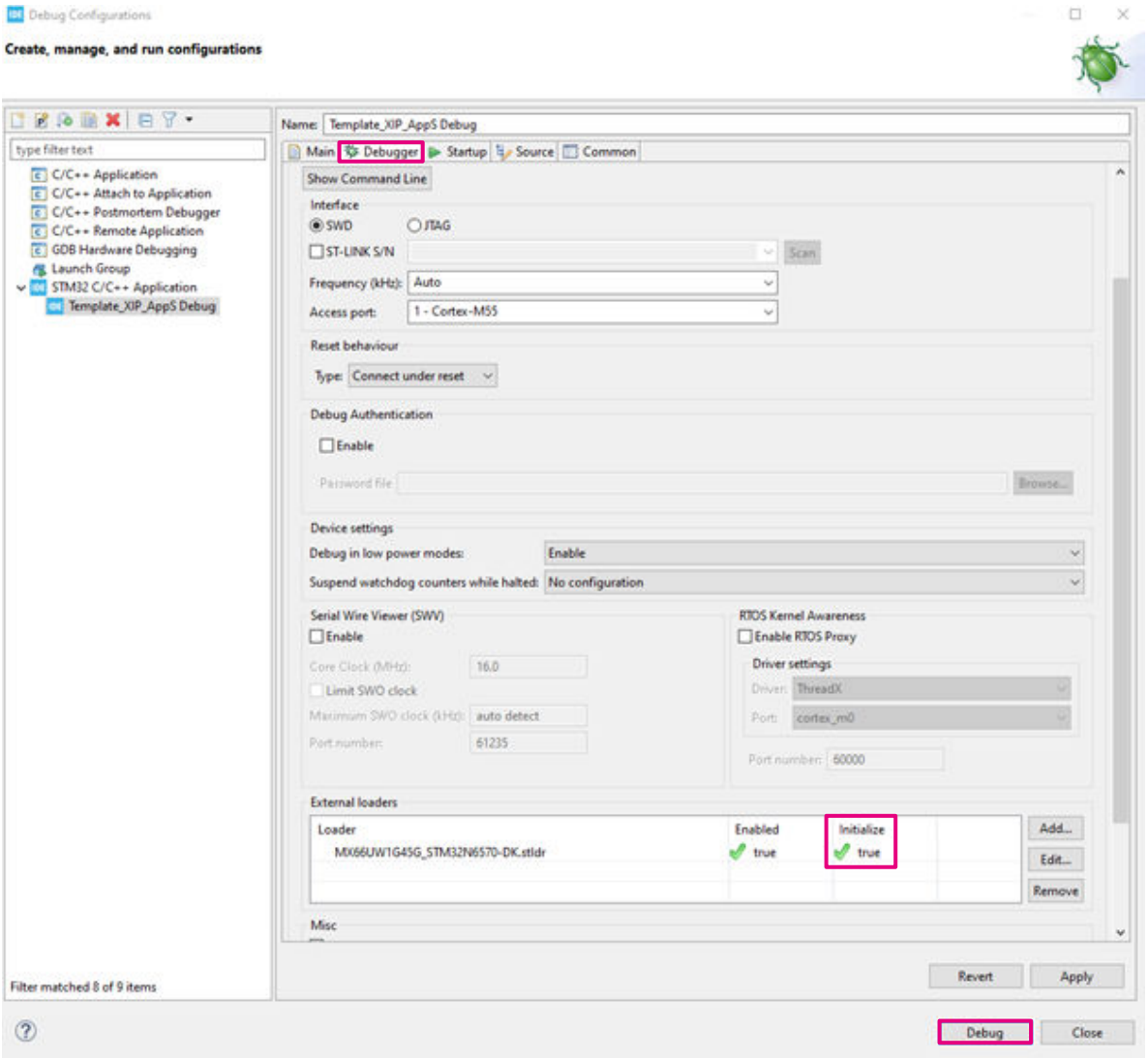


4.1.3 Debugging the application

The Template_XIP_Apps configuration is responsible for loading both the Template_XIP_Apps and Template_XIP_FSBL images.

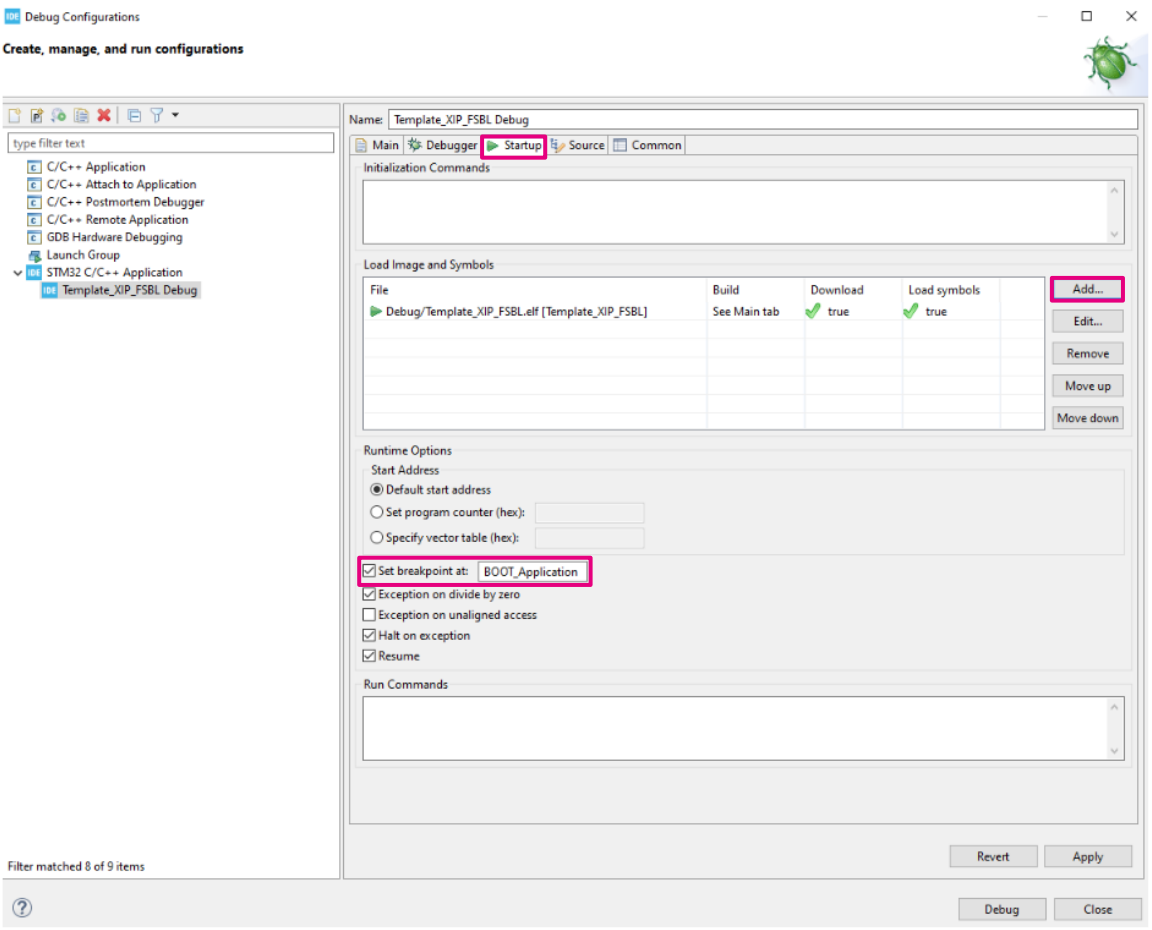
The external flash loader must be set in the debug configuration as shown in Figure 14.

Figure 14. ST-LINK GDB server debug configuration - Application (1 of 5)



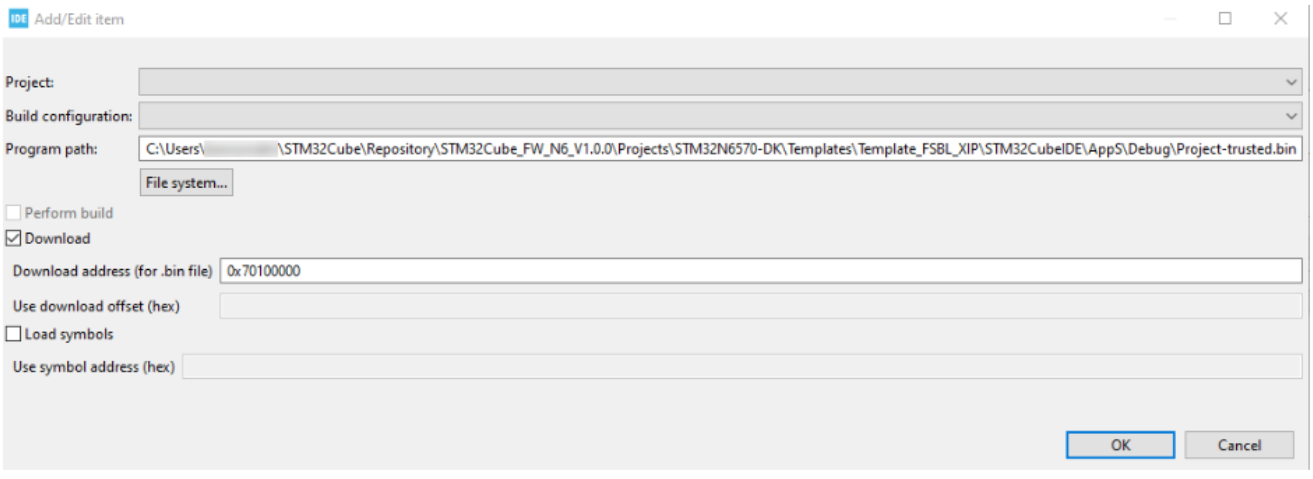
Set a breakpoint at the BOOT_Application function as shown in Figure 15.

Figure 15. ST-LINK GDB server debug configuration - Application (2 of 5)



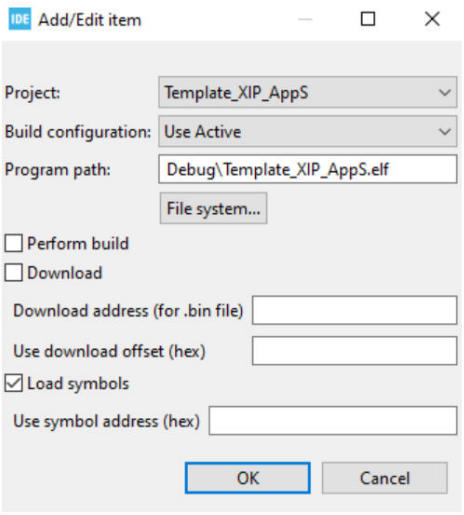
Then, add the `Project_trusted.bin` as shown in Figure 16. The download address for the Template_FSBL_XIP is 0x70100000.

Figure 16. ST-LINK GDB server debug configuration - Application (3 of 5)



To download also the Template_XIP_Apps, click on [**Add...**], browse to the correct project. The result is shown in Figure 17.

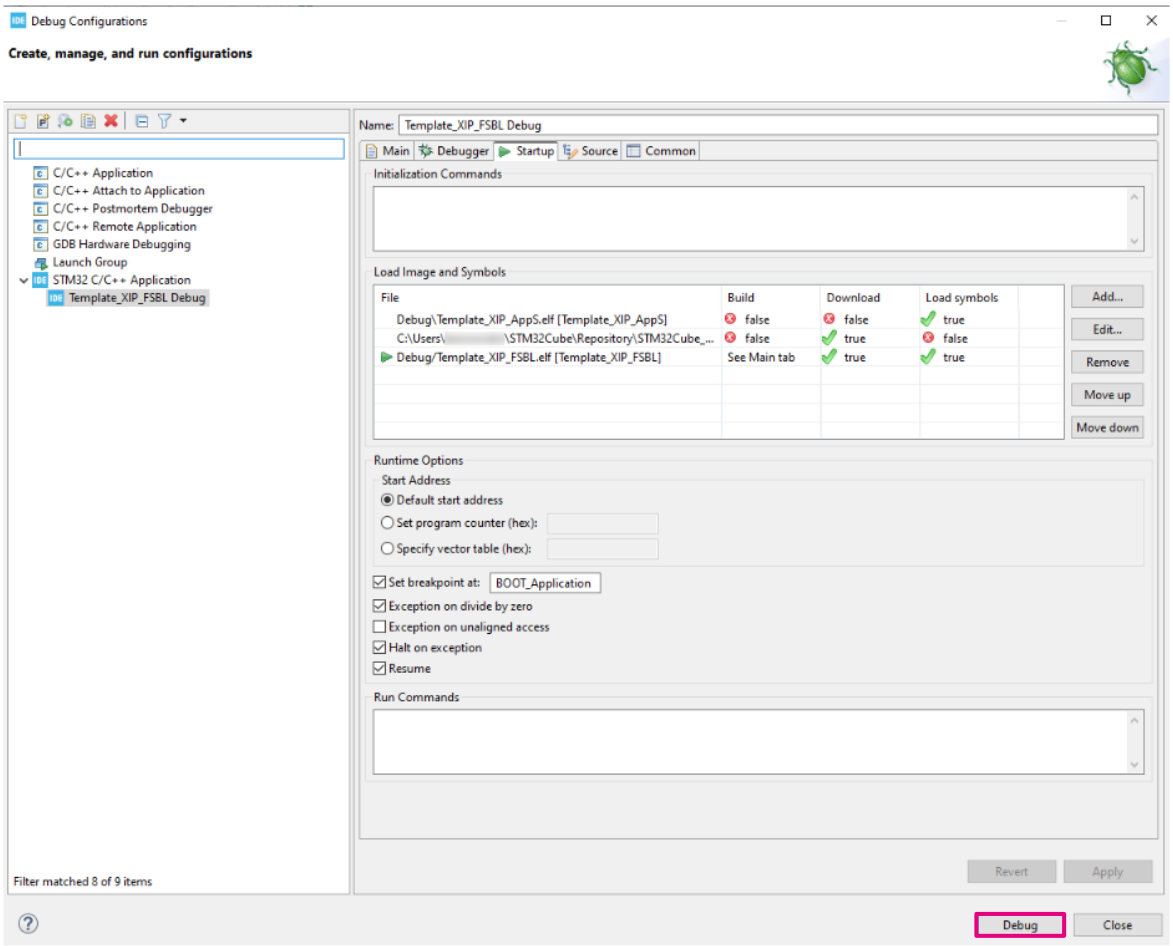
Figure 17. ST-LINK GDB server debug configuration - Application (4 of 5)



The order in the load list is very important. The last entry in the list, marked by a green arrow (refer to Figure 18), is the image debugged with this debug configuration. Consequently, the debugger fetches the program counter value (PC) from this image.

The configuration is complete.

Figure 18. **ST-LINK GDB server debug configuration - Application (5 of 5)**

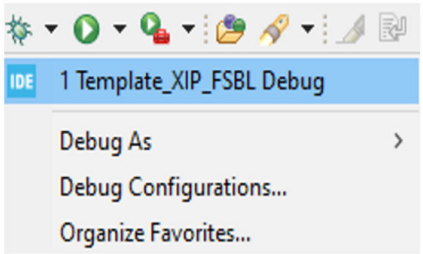


DT76552V1

4.1.4 Launching the configurations (ST-LINK GDB server)

To debug the project, launch the Template_XIP_FSBL debug session. The execution starts at the beginning of the main function from the FSBL project. Then, it is possible to jump to the application.

Figure 19. **ST-LINK GDB server debug configuration launch**

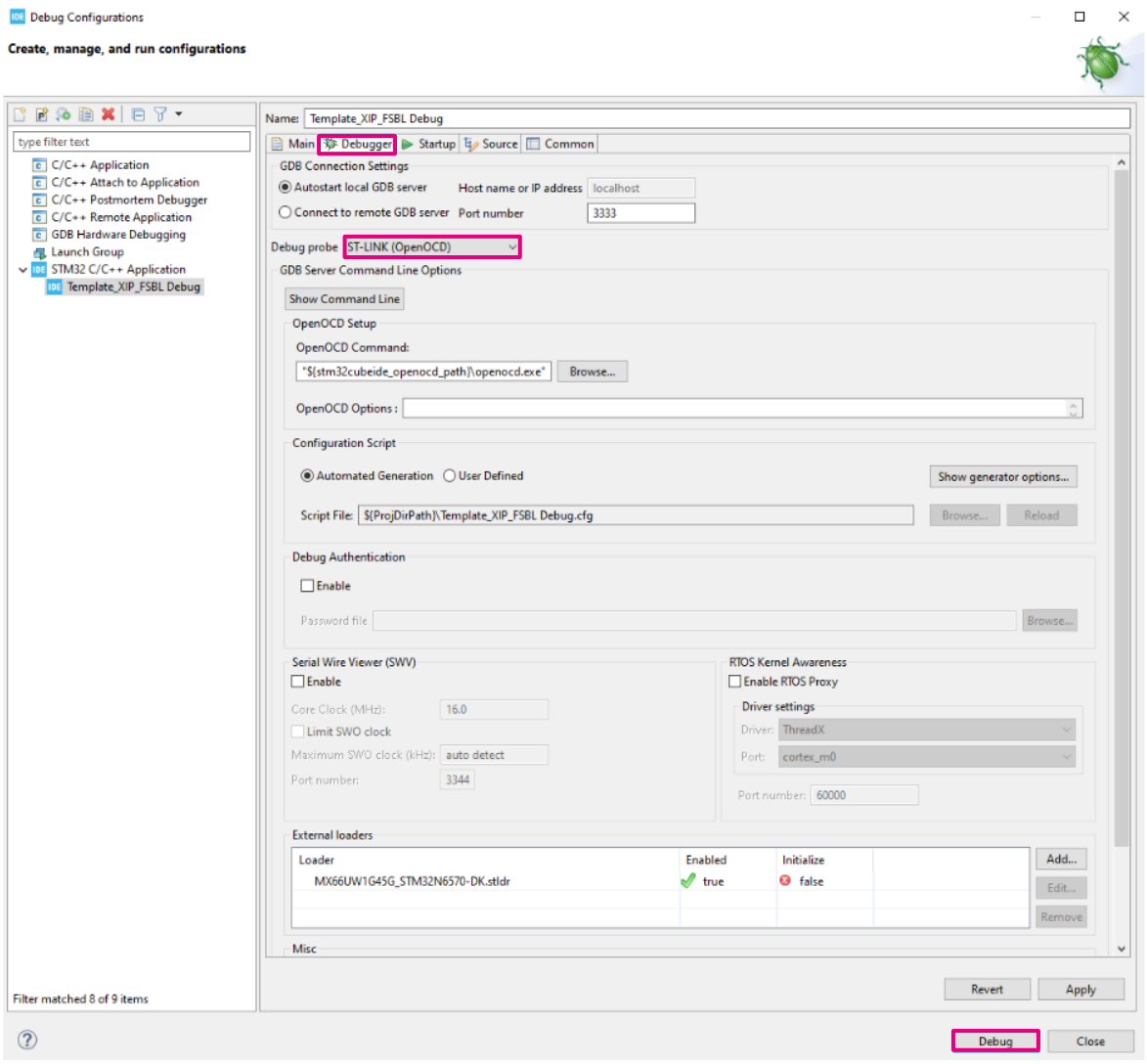


DT76553V1

4.2 Setting up with OpenOCD

Select *ST-LINK (OpenOCD)* as the [**Debug probe**] in the debug configuration of the Template_XIP_FSBL.

Figure 20. OpenOCD debug configuration



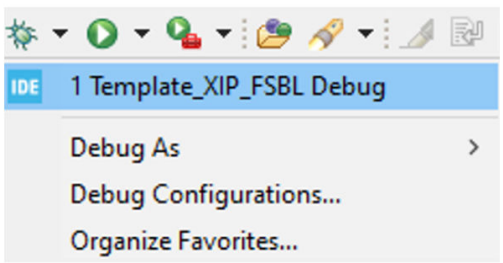
DT76554V1

The rest of the configuration is the same as for debugging with the ST-LINK GDB server (refer to Figure 18).

4.2.1 Launching the configurations (OpenOCD)

The same approach is used as with the ST-LINK GDB server (refer to Section 4.1.4).

Figure 21. OpenOCD debug configuration launch



DT76555V1

Revision history

Table 2. **Document revision history**

Date	Revision	Changes
17-Jun-2025	1	Initial release.

Contents

List of tables

List of figures