AN5952

Application note

How to use CMake in STM32CubeIDE

Introduction

This application note describes how to control the software compilation process using the CMake utilities for C/C++ projects in the STMicroelectronics STM32CubeIDE integrated development environment.

1 General information

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

arm

Note: Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

1.1 Purpose

STM32CubeIDE provides support for CMake, a build configuration tool requested by users. This feature helps developers create and manage projects for STM32 microcontroller (MCU) and microprocessor (MPU) platforms.

1.2 The use cases in this document

In the STM32CubeIDE context, a user can compile C/C++ projects using either the makefile or the CMake solutions. This document details the use of CMake for two use cases:

- The user wants to work with an existing CMake project structure
- The user wants to start a CMake development from scratch

1.3 Compatible toolchain

The STM32CubeIDE CMake support described in this application note is compatible with the following minimum toolchain version:

- STMicroelectronics STM32CubeIDE v2.0.0

1.4 Prerequisites

The compatibility with STM32CubeIDE is from CMake v3.30.8 onwards.

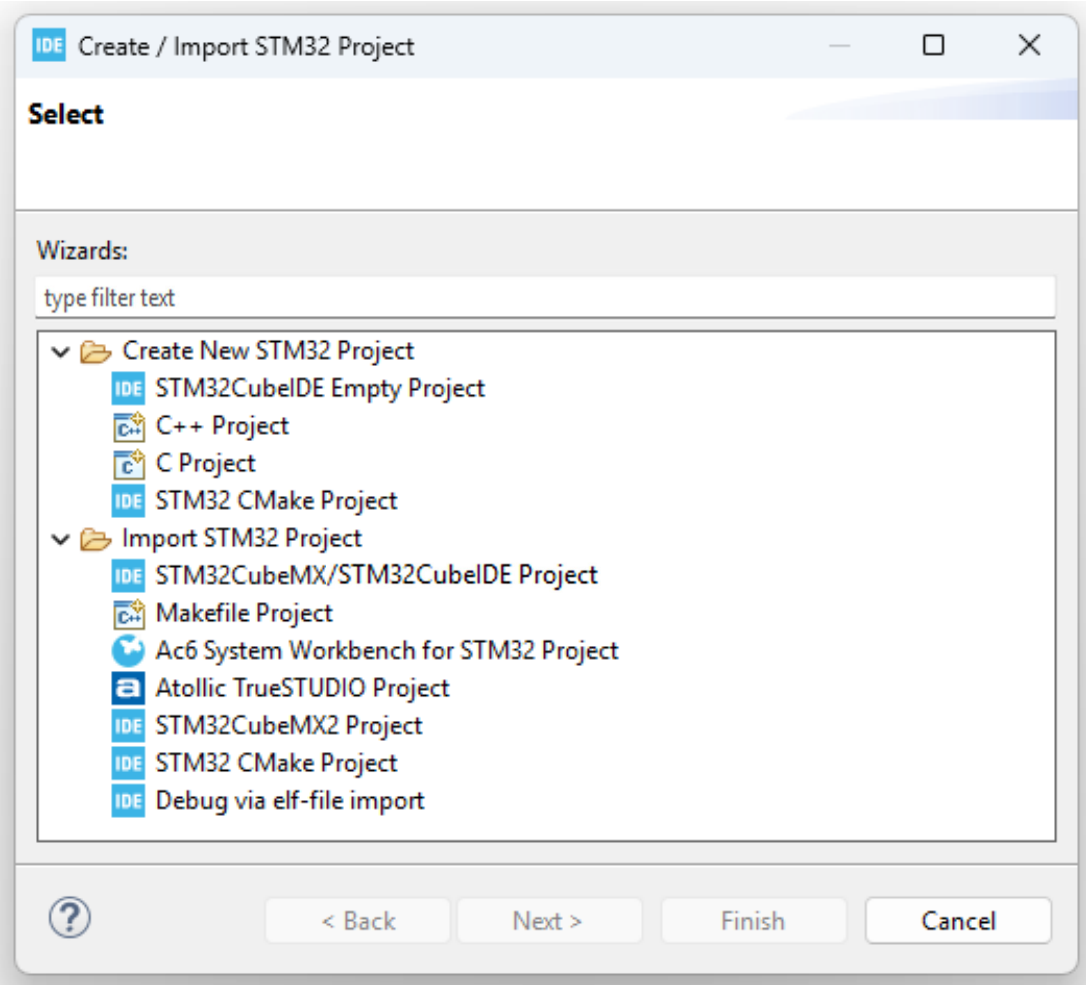
2 Create projects

2.1 Create a new project from an existing CMake project structure

Creating a new project with an existing CMake structure offers an easy way to use an already developed or a downloaded project or library for use in STM32CubeIDE. To do so, start by following the steps below:

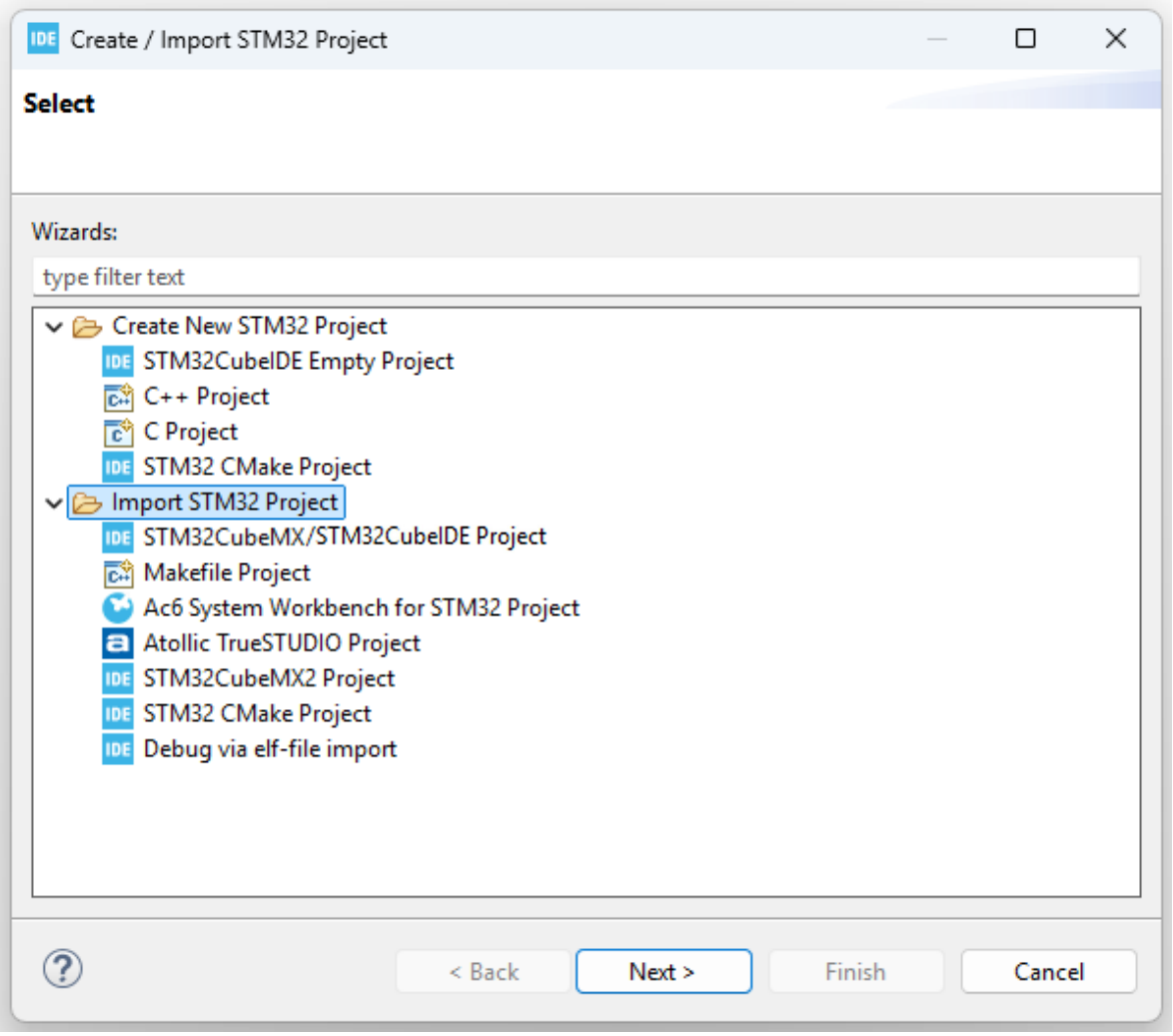
- Select [**File**]>[**STM32 Project Create/Import**] as illustrated in the figure below.

Figure 1. CMake project creation



- Select [**Import STM32 Project**] as illustrated in the figure below.

Figure 2. STM32 CMake project import



- Select [**Next >**]

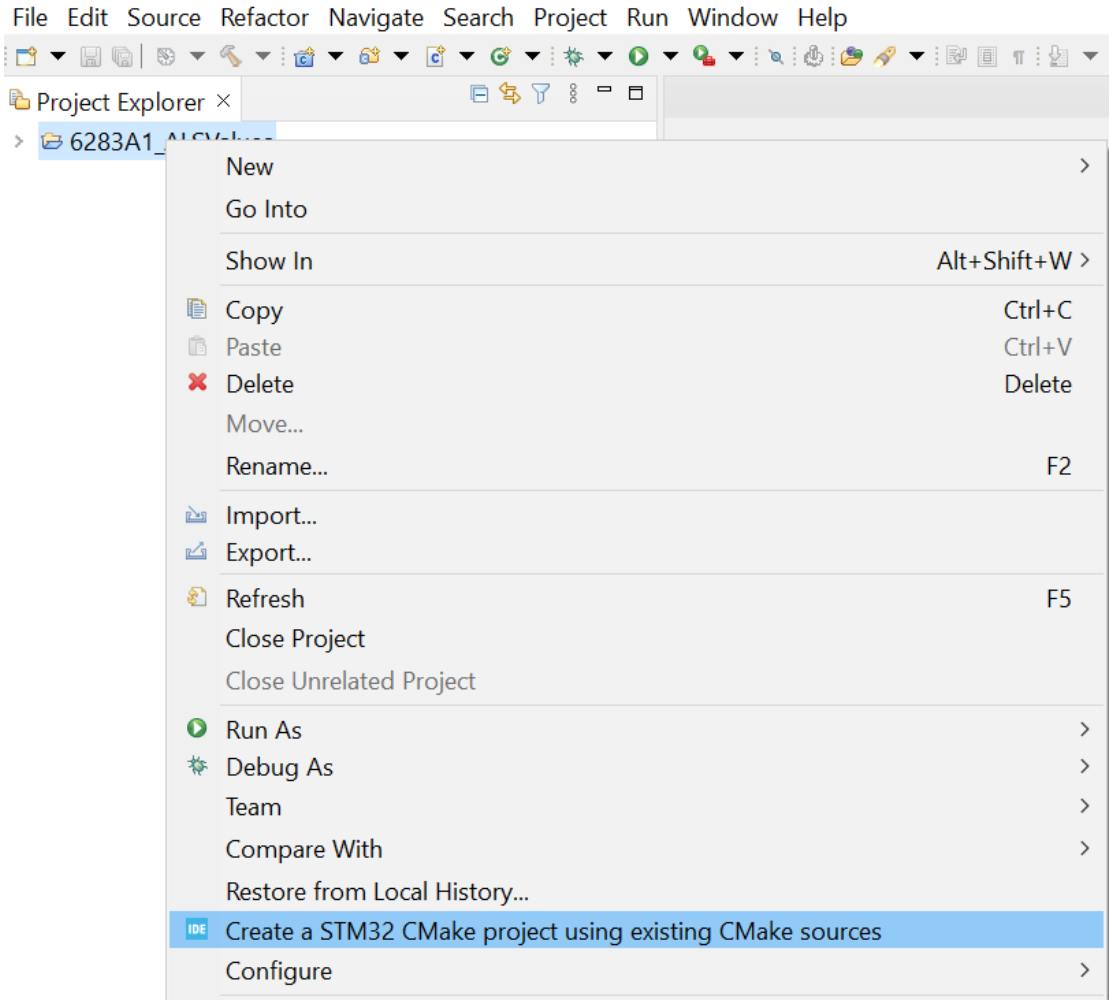
The next wizard page allows the setup of a CMake project in two different ways:

- Project creation inside an existing CMake project structure (refer to Section 2.1.2)
- Project creation external to an existing CMake project structure (refer to Section 2.1.3)

2.1.1 Create a CMake project as a subproject

Alternatively, to create a CMake project as a subproject in a more complex project structure, use the following context menu option as illustrated in the figure below.

Figure 3. CMake project as a subproject



Then, for a project creation inside or external to an existing CMake project structure, refer to Section 2.1.2 or Section 2.1.3.

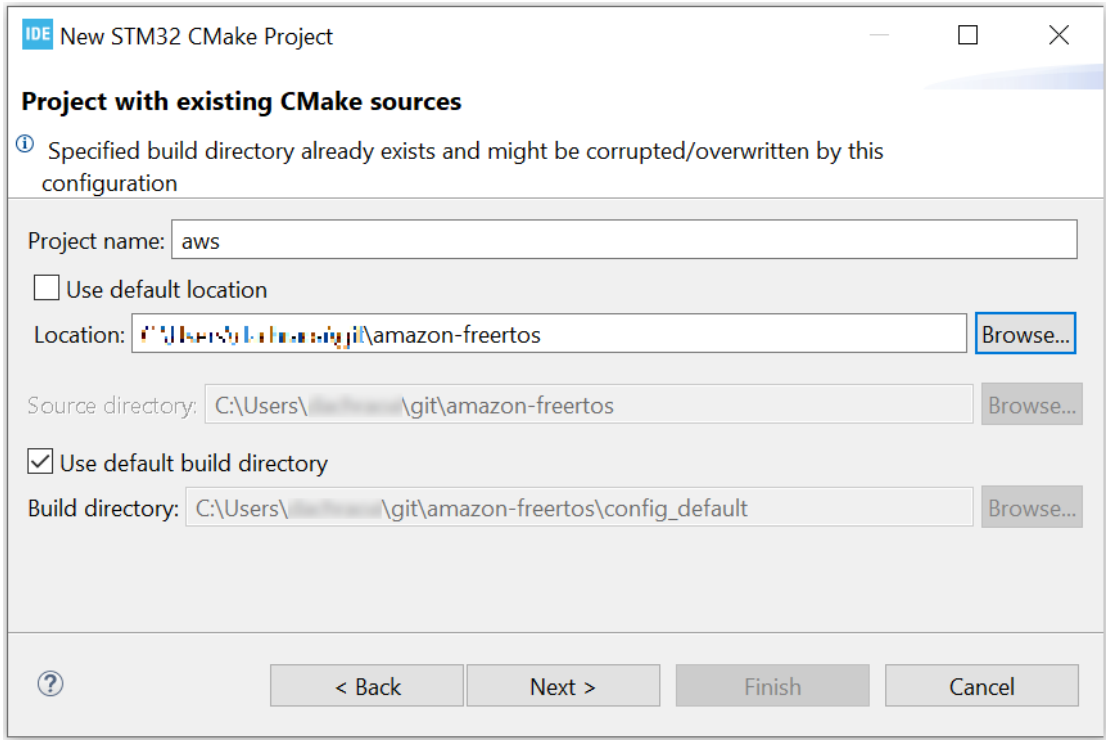
2.1.2 Create a new project in an existing CMake project structure

To create a new project inside an existing package that was already downloaded or created, do the following:

- Specify a name for the project.
- Uncheck [**Use default location**].
- Use the [**Browse…**] button and select the root directory of a CMake-based project. This disables the [**Source directory**] field since the project location and the CMake source directory are the same.
- Click on [**Next >**].

The selection window is illustrated in the figure below

Figure 4. Project creation inside an existing CMake project structure



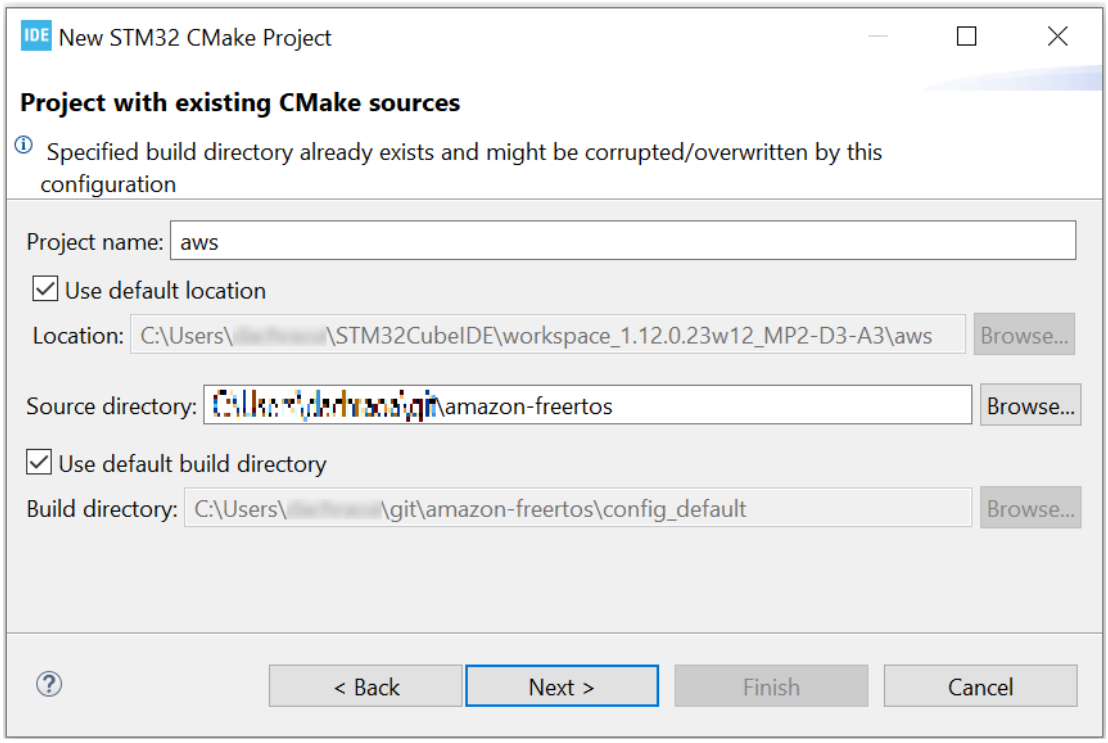
2.1.3 Create a new project outside an existing CMake project structure

In this other way to set up the project, it is created in the user's workspace. This is useful in the case of a complex project structure to link the existing package from the source directory to the project.

- Specify a name for the project.
- Specify an empty directory for the project. Keep [**Use default location**] checked so that the project is created inside the current workspace directory.
- Use the [**Browse…**] button and select the root directory of a CMake-based project. The selected directory is linked into the project.
- Click on [**Next >**].

The selection window is illustrated in the figure below

Figure 5. Project creation external to an existing CMake project structure

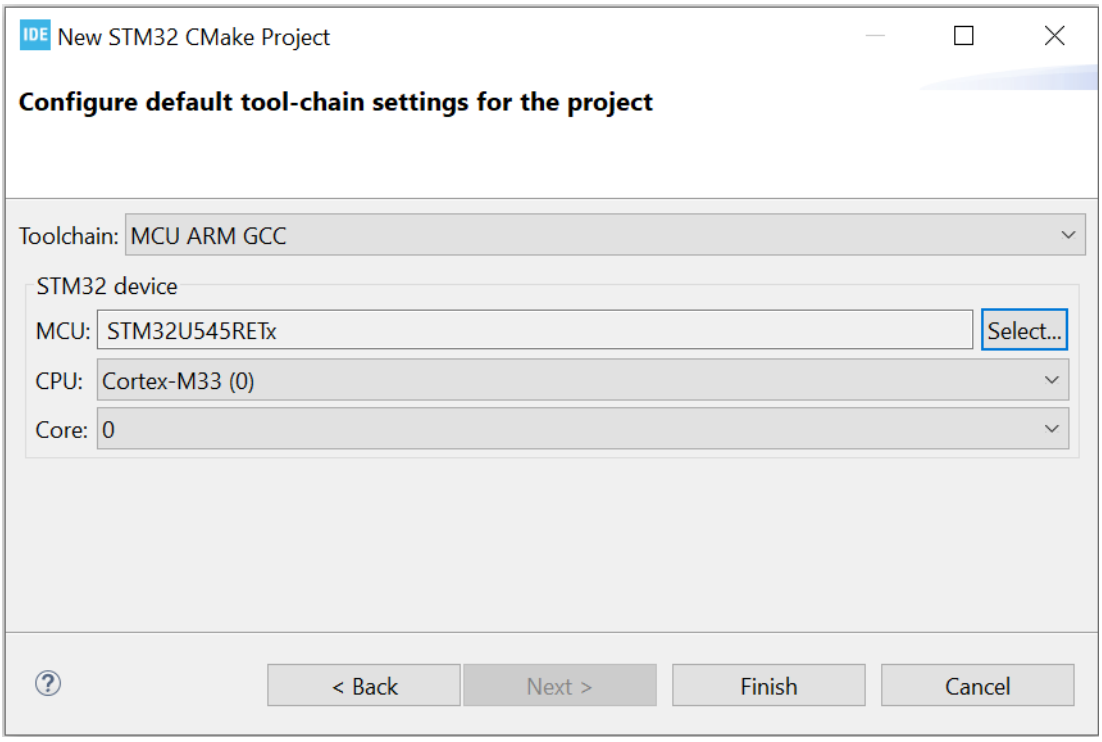


The next wizard page allows the configuration of a default toolchain and its relevant options for the created project. This is not directly relevant for building the CMake project structure but the STM32CubeIDE might require this information for certain of its features to function properly.

- Fill in:
  - The information about the MCU and core for the debugger
  - The proper indexing, for instance for code completion
  - The selected toolchain, which is added to the PATH variable when building the project
- Select [**Finish**]

The selection window is illustrated in the figure below

Figure 6. **CMake default toolchain**



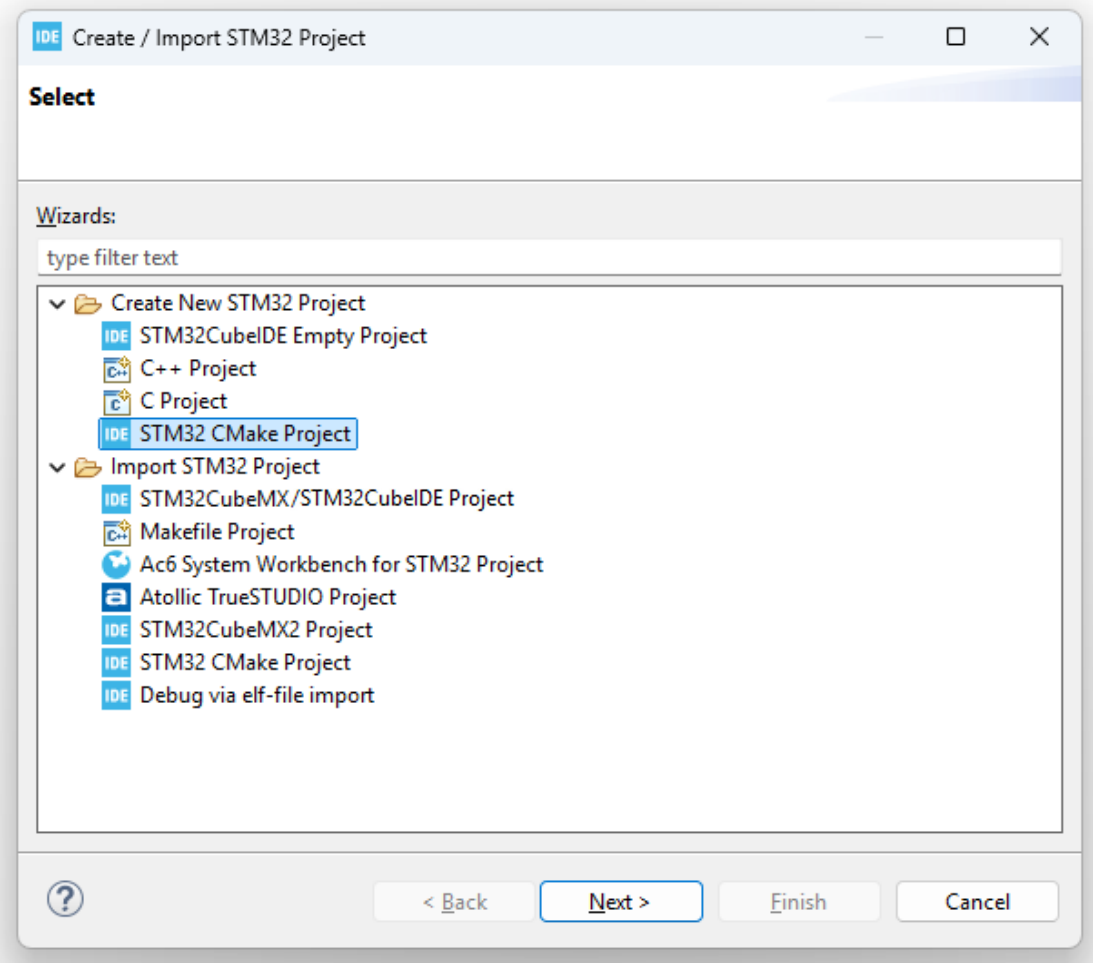
At this stage, the project is created and visible in the *Project Explorer* view.

2.2 Create a new CMake project from scratch

STM32CubeIDE allows the creation of a user specific project using CMake. To do so, proceed as per the steps below:

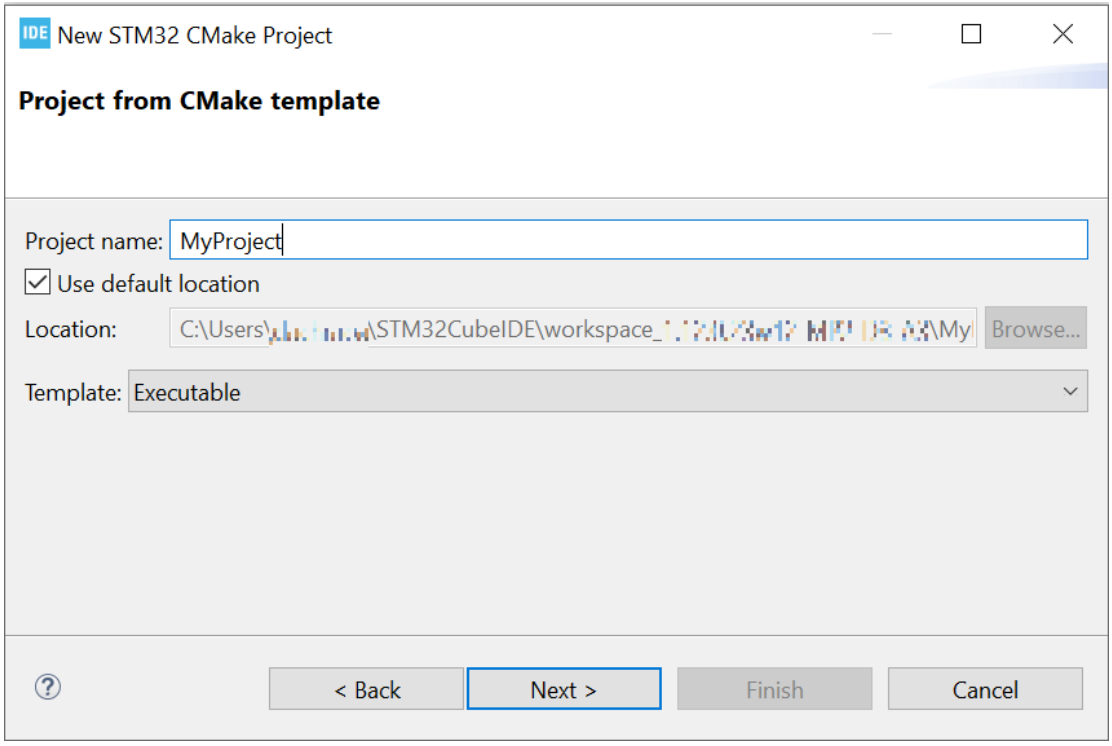
- Select [**[File]>[STM32 Project Create/Import]>[Create New STM32 Project] >[STM32 CMake Project]**] as shown in Figure 7.

Figure 7. New STM32 CMake Project



- Select [**Next >**]
- Provide the information requested for the project creation as illustrated in Figure 8:
  - Specify a name for the project.
  - Specify an empty directory for the project. Keep [**Use default location**] checked so that the project is created inside the current workspace directory.
  - Select the template to use for the project. Currently, the templates for *"Executable"* and *"Static Library"* targeting the *"MCU ARM GCC toolchain"* are available.
- Click on [**Next >**]

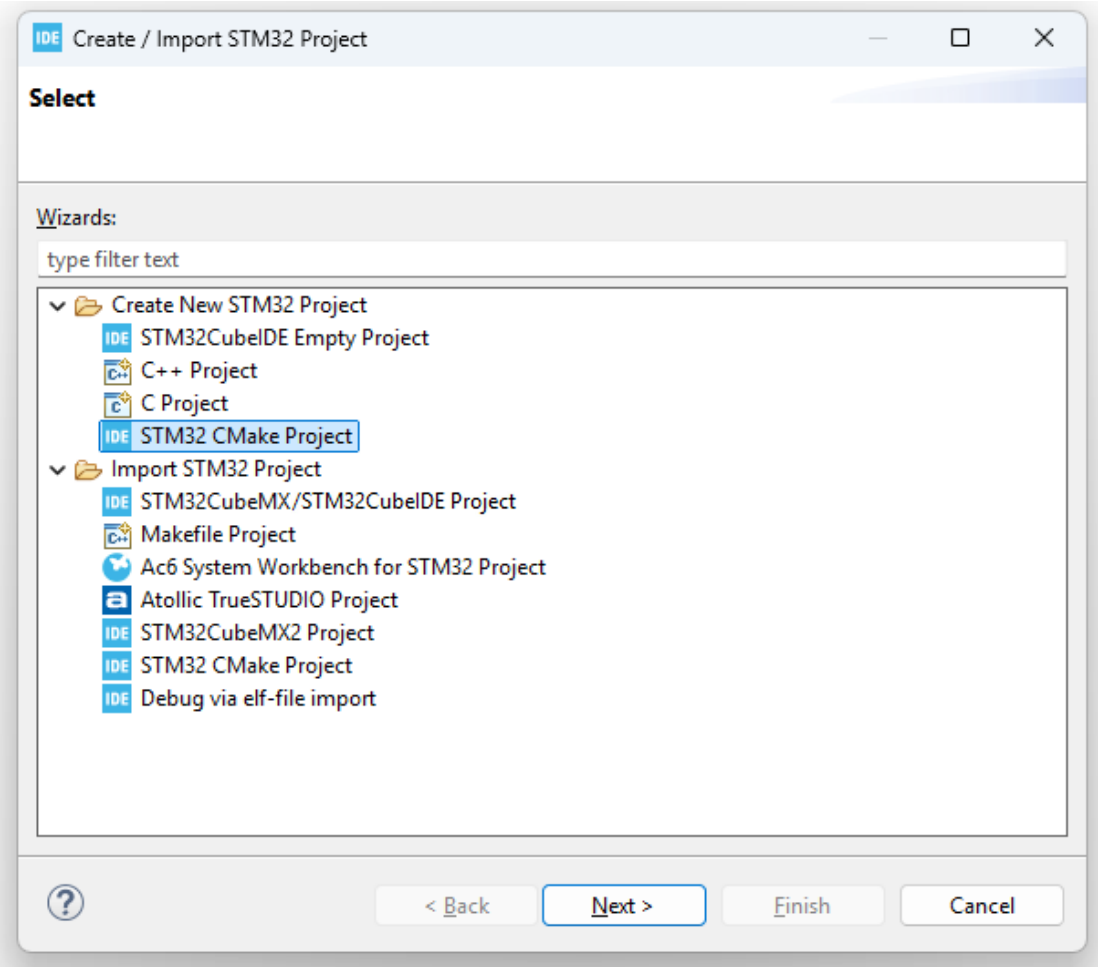
Figure 8. **CMake Project from template**



The next wizard page allows the configuration of a default toolchain and its relevant options for the created project.

- Select the options for the STM32 device to be used for the new project as illustrated in Figure 9

Figure 9. CMake default toolchain (alt.)



- Click on [**Finish**]

3 Configure and build

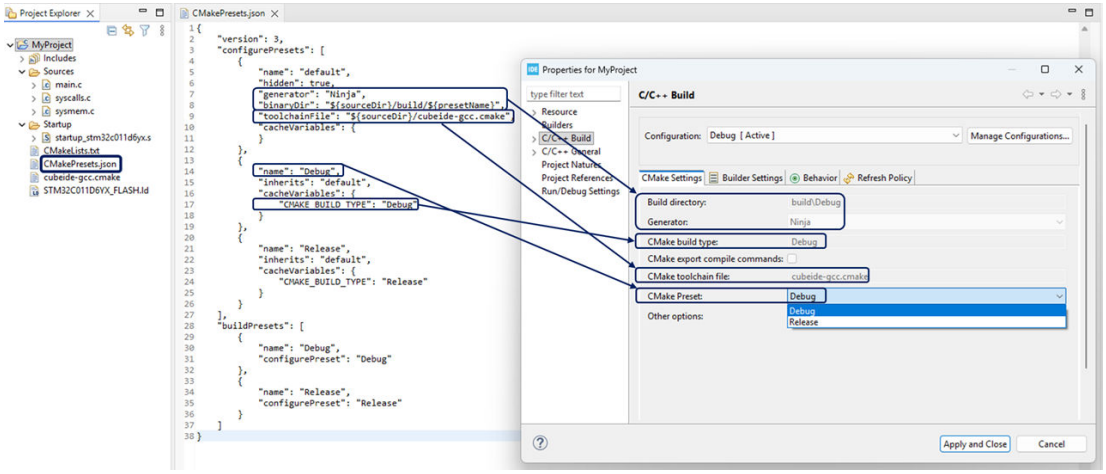
3.1 CMake build settings

The CMake-related build settings can be specified in the project properties illustrated in Figure 10:

- Select *C/C++ Build* available in the project properties.
- Select the *CMake Settings* tab.
- For CMake project containing a CMakePresets.json file, select the CMake Preset from the list of preset names available in the CMake Preset combo list, as example Debug or Release. Once Preset selected, the following parameters are automatically updated from CMakePresets.json file:
  - BuildDirectory
  - Generator
  - buildType
  - Toolchain file.

  The values of the parameters above can be changed by editing the CMakePresets.json file. If a CMake project does not include a CMakePresets.json file, the user can enter the CMake settings manually.
- Specify other CMake build properties in Other Options field when needed

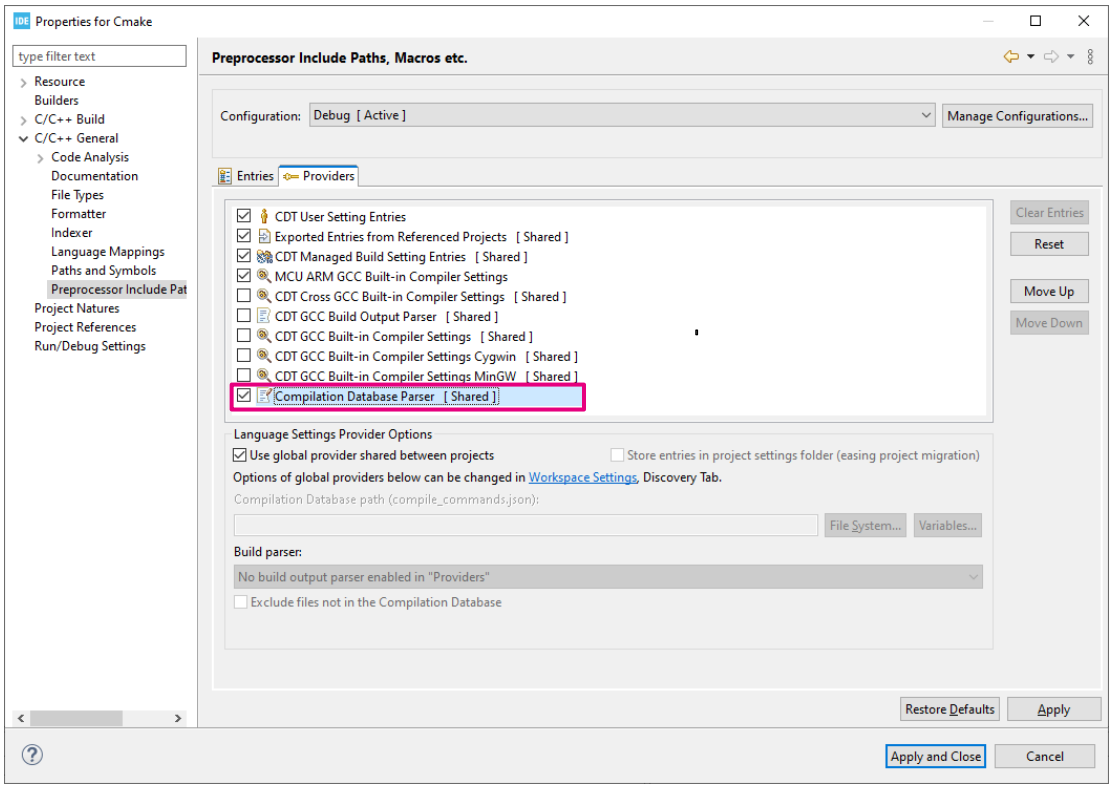
Figure 10. CMake build options (CMake Preset)



DT72064V2

- Configure the parameters as shown in Figure 11, when working with CMake projects in STM32CubeIDE, to mitigate indexer issues resulting from CDT™ limitations.

Figure 11. Compilation database parser



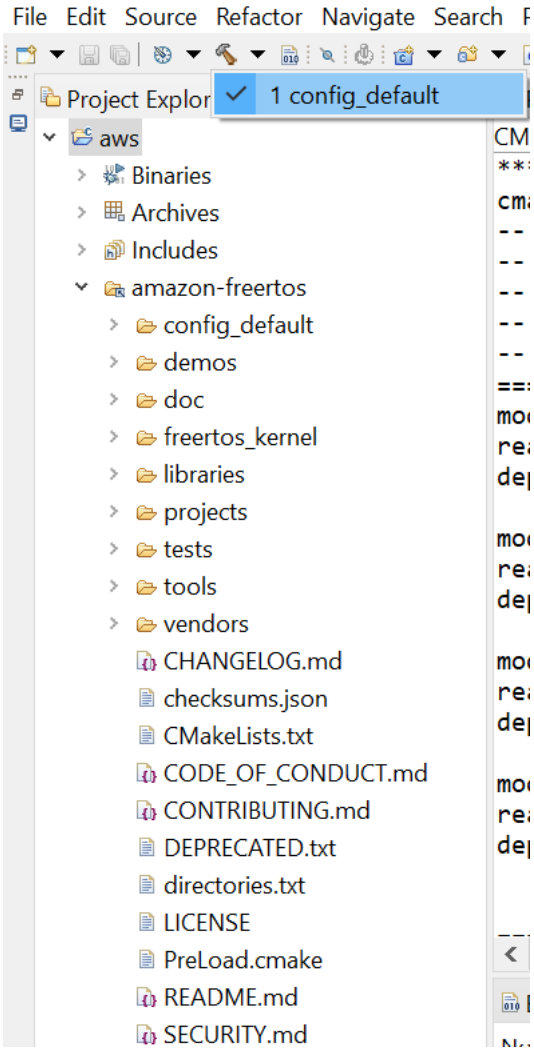
3.2 Build a CMake-based project

For CMake-based projects, the STM32CubeIDE build configuration management and user interfaces are similar to those for non-CMake-based projects, including:

- Menus
- Toolbars

Multiple project build configurations can be created and associated with different sets of CMake settings, such as *“Debug”* and *“Release”*.

Figure 12. CMake project manual build



DT72070V1

During the first project build, the STM32CubeIDE automatically performs:

- The CMake configure step
- The setup of the CMake cache
- The build files generation into the specified build directory

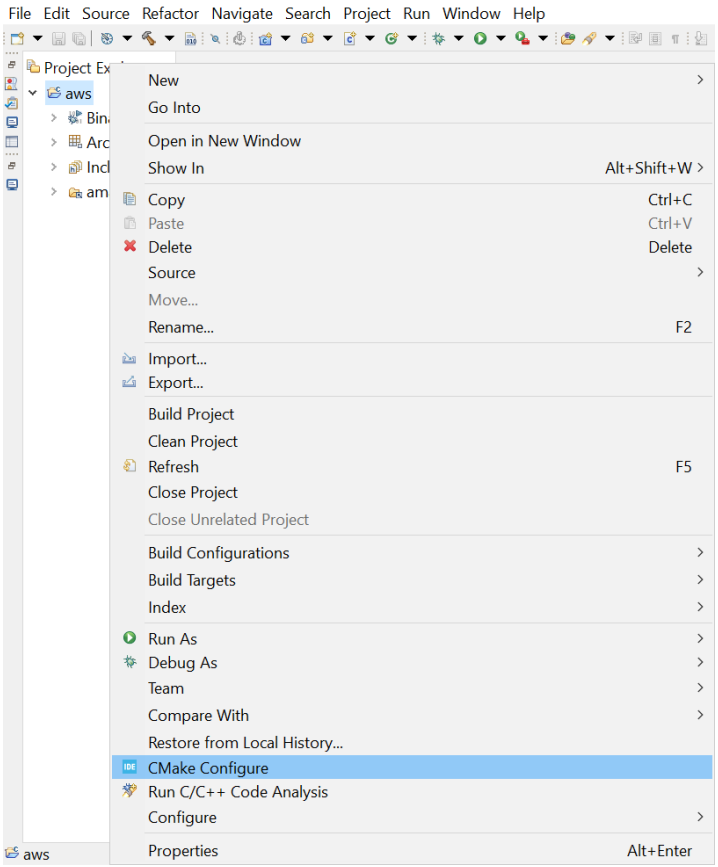
Any subsequent builds are performed in the build directory using the existing CMake cache.

3.3 Manual CMake configure step

The CMake configuration, build, and file generation can also be performed manually using one of the following methods:

- Either the project context menu as shown in Figure 13.

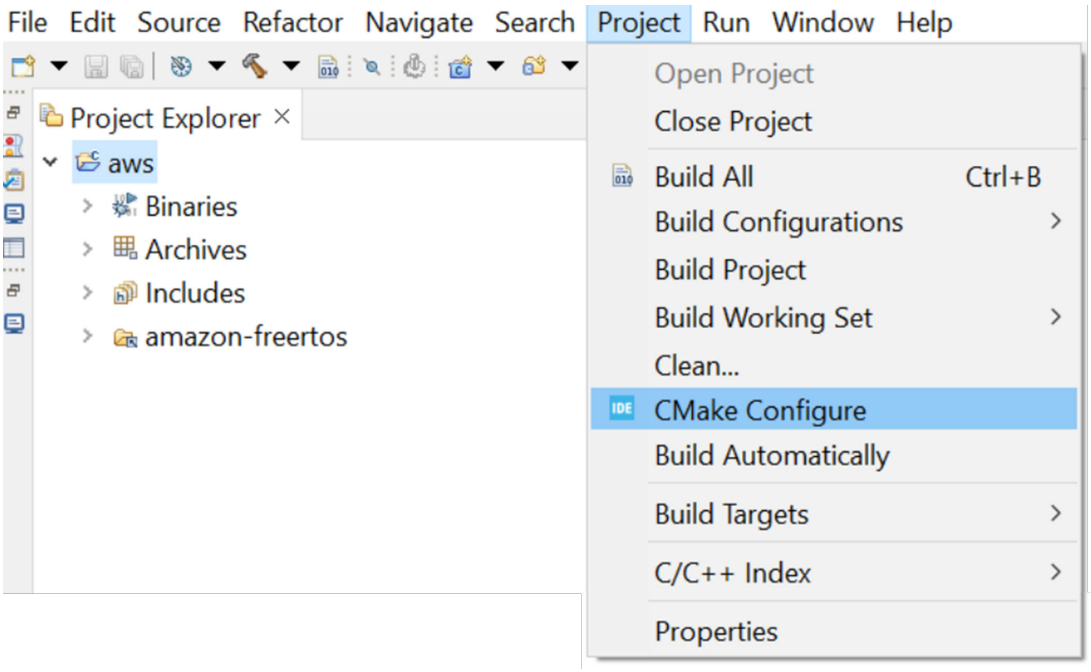
Figure 13. CMake Project context menu



DT72068V1

- Or the *Project* menu as shown in Figure 14.

Figure 14. CMake project menu

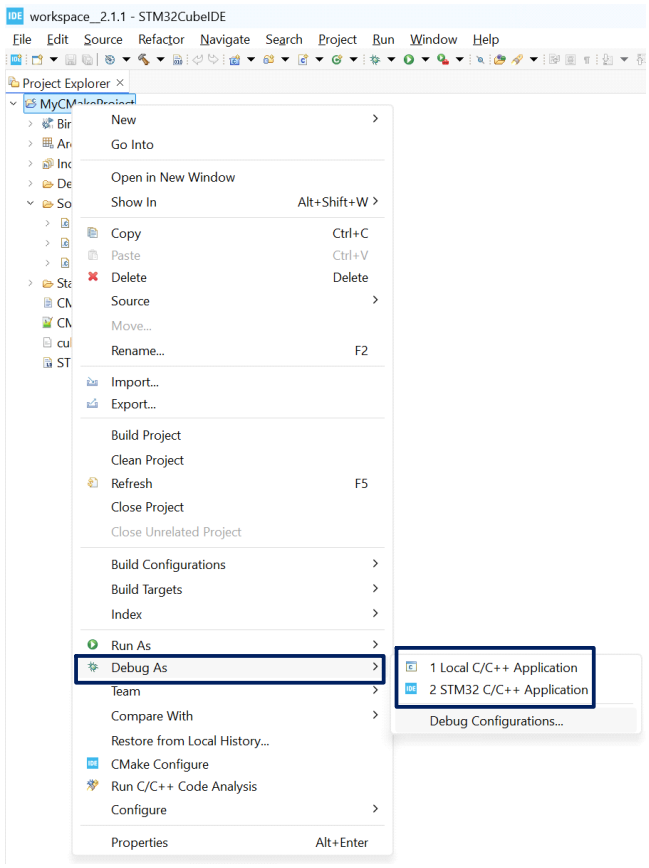


3.4 Debug a CMake-based project

STM32CubeIDE allows the debugging of a CMake-based projects through the same entries point as non-CMake-based projects:

- Menus: Right click on: CMake project > Debug As > Debug Configuration ... > STM32 C/C++ Application > New configuration as displayed in Figure 15

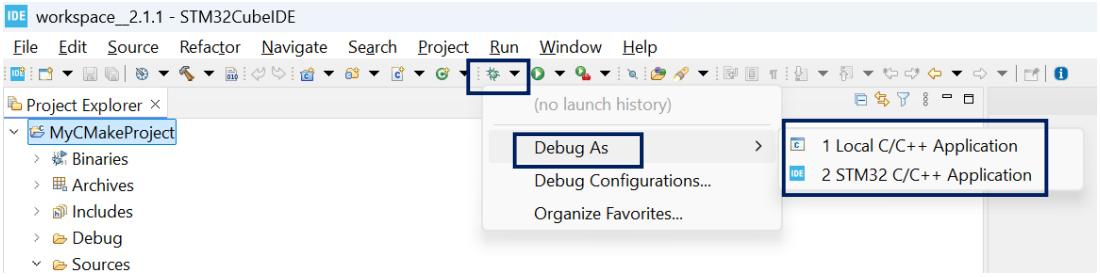
Figure 15. CMake based projects



DT78303V1

- Toolbars illustrated in Figure 16

Figure 16. Toolbar selection



DT78304V1

Revision history

Table 1. **Document revision history**

Date	Revision	Changes
07-Jul-2023	1	Initial release.
19-Feb-2025	2	Updated Section 3.1: CMake build settings: • Updated Figure 10. CMake build options (CMake Preset) • Added Figure 12. CMake build options (CMake build type) and Figure 11. Compilation database parser
17-Sep-2026	3	Added: • Section 3.4: Debug a CMake-based project Updated: • Section 1.1: Purpose • Section 1.3: Compatible toolchain • Section 1.4: Prerequisites • Section 2.1: Create a new project from an existing CMake project structure – Figure 1. CMake project creation – Figure 2. STM32 CMake project import • Section 2.1.1: Create a CMake project as a subproject • Section 2.1.2: Create a new project in an existing CMake project structure • Section 2.1.3: Create a new project outside an existing CMake project structure • Section 2.2: Create a new CMake project from scratch – Figure 7. New STM32 CMake Project • Section 3.1: CMake build settings – Figure 10. CMake build options (CMake Preset) • Section 3.2: Build a CMake-based project • Figure 14. CMake project menu Removed CMake build options (CMake build type) figure

Contents

1 **General information** 2
1.1 Purpose 2
1.2 The use cases in this document 2
1.3 Compatible toolchain 2
1.4 Prerequisites 2
2 **Create projects** 3
2.1 Create a new project from an existing CMake project structure 3
2.1.1 Create a CMake project as a subproject 5
2.1.2 Create a new project in an existing CMake project structure 6
2.1.3 Create a new project outside an existing CMake project structure 7
2.2 Create a new CMake project from scratch 9
3 **Configure and build** 12
3.1 CMake build settings 12
3.2 Build a CMake-based project 14
3.3 Manual CMake configure step 15
3.4 Debug a CMake-based project 17
Revision history 18
List of figures 20

List of figures

Figure 1. CMake project creation . . . 3
Figure 2. STM32 CMake project import . . . 4
Figure 3. CMake project as a subproject . . . 5
Figure 4. Project creation inside an existing CMake project structure . . . 6
Figure 5. Project creation external to an existing CMake project structure . . . 7
Figure 6. CMake default toolchain . . . 8
Figure 7. New STM32 CMake Project . . . 9
Figure 8. CMake Project from template . . . 10
Figure 9. CMake default toolchain (alt.) . . . 11
Figure 10. CMake build options (CMake Preset) . . . 12
Figure 11. Compilation database parser . . . 13
Figure 12. CMake project manual build . . . 14
Figure 13. CMake Project context menu . . . 15
Figure 14. CMake project menu . . . 16
Figure 15. CMake based projects . . . 17
Figure 16. Toolbar selection . . . 17

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved