STM32CubeIDE Documentation

Release 2.3.0

Nov 13, 2026

Table Of Contents

1 STM32CubeIDE documentation 4
1.1 Introduction to STM32CubeIDE . . . 4
1.1.1 What is STM32CubeIDE? . . . 4
1.1.2 Getting started with STM32 and STM32CubeIDE . . . 4
1.1.3 Videos related to STM32CubeIDE . . . 5
1.1.4 Other resources related to STM32CubeIDE . . . 5
1.1.5 Toolchain and Build tools manuals (GNU, Binary utilities, Newlib, Make, GDB...) . . . 6
1.1.6 Documentation . . . 6

2 Release Notes 7
2.1 Release details . . . 7
2.1.1 Introduction . . . 7
2.1.2 What is new in STM32CubeIDE 2.3.0? . . . 7
2.1.3 Release information . . . 8
2.1.4 Upgrades . . . 8
2.1.5 Host PC system requirements . . . 8
2.1.6 Customer support . . . 9
2.1.7 Software updates . . . 9
2.1.8 General information . . . 9
2.2 Release fixes . . . 10
2.3 Known limitations . . . 10
2.3.1 Limitations . . . 11
2.3.2 General issues . . . 11
2.3.3 STM32CubeCLT issues . . . 13
2.3.4 Target-specific issues . . . 13

3 Quick Start Guide 16
3.1 Introduction . . . 16
3.2 Information center . . . 16
3.3 Workspace and projects . . . 17
3.3.1 Project types in STM32CubeIDE . . . 18
3.3.2 Start from scratch with STM32CubeIDE . . . 18
3.3.3 Start from existing project with STM32CubeIDE . . . 23
3.4 Project information . . . 27
3.5 Debugging . . . 29

4 Installation Guide 31
4.1 Introduction . . . 31
4.2 System requirements . . . 31
4.2.1 Microsoft® Windows® . . . 31
4.2.2 Linux® . . . 31
4.2.3 MacOS® . . . 32
4.2.4 Hardware requirements . . . 32
4.3 Important information . . . 32
4.3.1 Product installer . . . 32
4.3.2 Installing from USB memory . . . 32
4.3.3 Product upgrades . . . 32
4.4 STM32CubeIDE installation (Windows®) . . . 32
4.5 STM32CubeIDE installation (Linux®) . . . 35
4.6 STM32CubeIDE installation (macOS®) . . . 36
4.6.1 Rosetta® installation on M1-based computers . . . 39
4.7 Update an STM32CubeIDE installation . . . 40
4.8 Uninstall STM32CubeIDE (Windows®) . . . 42

4.9 Uninstall STM32CubeIDE (Linux®) . . . 42
4.10 Uninstall STM32CubeIDE (macOS®) . . . 42

5 Software License Agreement (SLA0048) 43
5.1 **Software Package License Agreement** . . . 43
5.2 DISCLAIMER . . . 44

6 FAQ 45
6.1 Authentication issue . . . 45
6.2 How to refresh the code after an update within STM32CubeMX standalone tool . . . 45
6.2.1 Reset perspectives . . . 45
6.2.2 Customize perspectives . . . 45
6.2.3 Create an own perspective . . . 46
6.3 Quick access to views, menus, preferences . . . 46
6.4 Key bindings . . . 47
6.5 Shortcuts . . . 48
6.5.1 Shortcuts to debug . . . 48
6.5.2 Shortcuts to code . . . 48
6.6 Dark theme . . . 48
6.7 Workspace path . . . 49
6.8 Link with editor . . . 49
6.9 Task tags . . . 50
6.10 Code templates . . . 50
6.11 Include browser . . . 51
6.12 Eclipse Marketplace . . . 51
6.13 Cyclic redundancy check - An error detection method . . . 52
6.14 Project migration from IAR Embedded Workbench® for Arm or Keil® MDK-ARM to STM32CubeIDE 52
6.15 The link time optimization (LTO) . . . 55
6.16 Memory management . . . 56
6.17 Debug an already-running target . . . 57

7 Community 58
7.1 Support link . . . 58
7.2 Community link . . . 58
7.3 Knowledge articles link . . . 58

8 Previous documents 59

1 STM32CubeIDE documentation

Welcome to the documentation portal for STM32CubeIDE.

This document covers the STM32CubeIDE tool only.

For information about STM32CubeIDE for Visual Studio Code , see the STM32CubeIDE for Visual Studio Code documentation

For users who want a brief introduction, watch our introductory YouTube video by clicking the image below and do not miss the video tutorials:



1.1 Introduction to STM32CubeIDE

1.1.1 What is STM32CubeIDE?

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem. For a complete description of STM32CubeIDE, as well as download links, refer to the STM32CubeIDE product page.

1.1.2 Getting started with STM32 and STM32CubeIDE

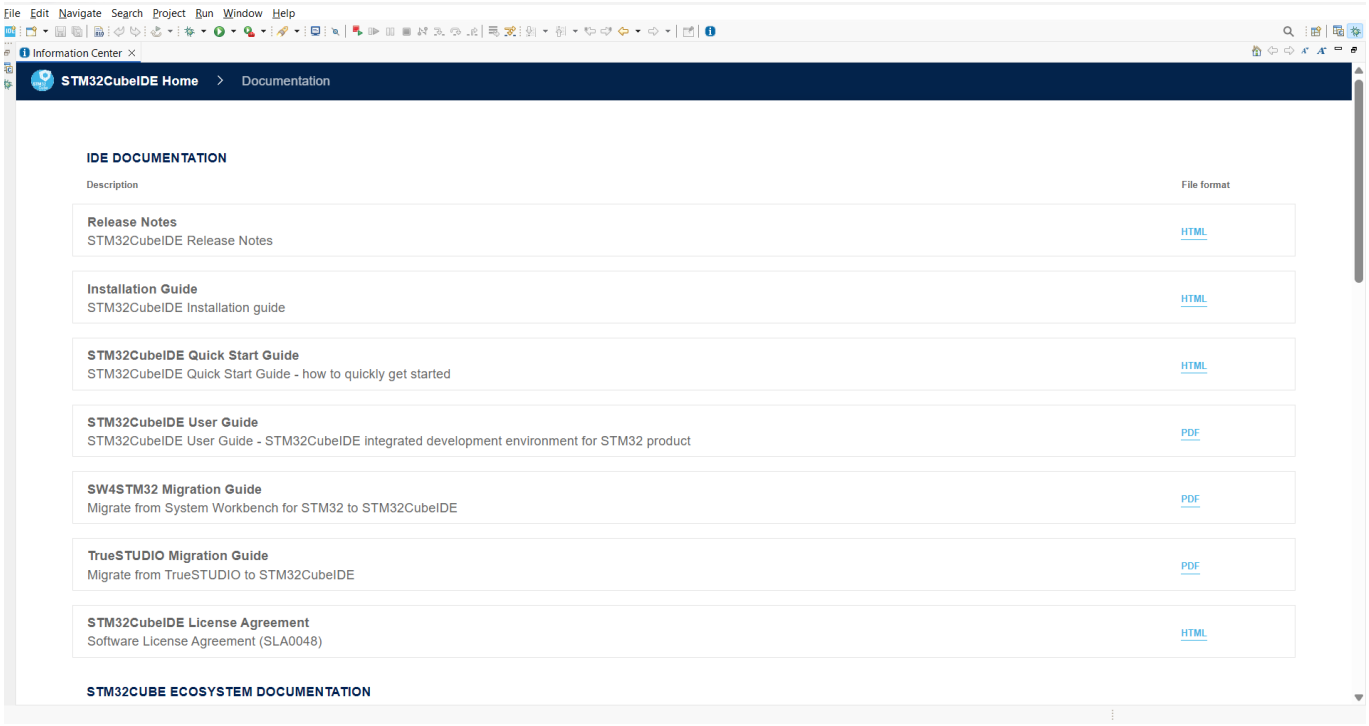
Getting started in STM32CubeIDE. Maybe you are a completely new user to the STM32 products and ecosystem, or maybe you are looking to migrate a legacy STM32 project into STM32CubeIDE.

Regardless of your background and your ambitions with STM32CubeIDE, we recommend that you start by creating a simple Empty project to try out the build and debug related features on a minimal project. Consult the STM32CubeIDE Quick start guide for basic information and the STM32CubeIDE User manual for detailed information. Or why not watch a quick video tutorial on how to create an STM32Cube project?

If your ambition is to migrate a project from the Ac6 System Workbench for STM32 or Atollic TrueSTUDIO®, then the documentation on how to import and convert your project is available in the “Manuals and release notes” section below.

There are also specific application notes on how to get started with application development for more complex STM32 products such as STM32H7 dual-core and STM32L5 with TrustZone. These application notes are available in the Application Notes section below.

Documentation is also available as part of STM32CubeIDE installation. To access documentation, go to Help > Information Center, then click on STM32CubeIDE manuals.



The STM32 device-specific application notes are deliberately not part of installation since they may be updated more frequently.

1.1.3 Videos related to STM32CubeIDE

- STM32CubeIDE 2.0.0: Stand-alone tools and updated project workflows
- How to use STM32CubeIDE

Some video tutorials have been created to explain STM32CubeIDE principles around project creation, build and launch configurations, and debug:

- STM32CubeIDE Tutorials

A specific video has been created to show users how to quickly get started with STM32L5 using various STM32Cube tools including STM32CubeIDE:

- How to enable TrustZone® and start a project with STM32L5

The following YouTube playlists contain several videos where you can learn about STM32CubeIDE basics and advanced debug features:

- MOOC - STM32CubeIDE Basics (12 videos)
- STM32CubeIDE Advanced Debug Features (5 videos)

1.1.4 Other resources related to STM32CubeIDE

The documentation resources related to STM32CubeIDE are continuously growing and hosted in different formats. The subchapters below list various documents, with links for easy access.

Application Notes

- AN6265 Getting started with STM32N6 MCUs in STM32CubeIDE

- AN6127 Getting started with STM32H7Rx/7Sx MCUs in STM32CubeIDE
- AN5952 How to use CMake in STM32CubeIDE
- AN5361 Getting started with projects based on dual-core STM32H7 microcontrollers in STM32CubeIDE
- AN5731 STM32CubeMX and STM32CubeIDE thread-safe solution
- AN5564 Getting started with projects based on dual-core STM32WL microcontrollers in STM32CubeIDE
- AN5394 Getting started with projects based on the STM32L5 Series in STM32CubeIDE
- AN5360 Getting started with projects based on the STM32MP1 Series in STM32CubeIDE
- AN4296 Use STM32F3/STM32G4 CCM SRAM with IAR Embedded Workbench®, Keil® MDK-ARM, STMicroelectronics STM32CubeIDE, and other GNU-based toolchains

1.1.5 Toolchain and Build tools manuals (GNU, Binary utilities, Newlib, Make, GDB...)

The following manuals are available in STM32CubeIDE and can be opened from STM32CubeIDE Information Center. To access documentation, go to Help > Information Center, then click on STM32CubeIDE manuals.

- GNU Assembler
- GNU Compiler Collection
- GNU C Library
- GNU C Preprocessor
- GNU Linker
- GNU Binary Utilities
- Red Hat Newlib C Library
- Red Hat Newlib Math Library
- Newlib nano readme
- Make
- Debugging with GDB
- GDB Quick reference card
- GNU Tools for STM32 patch list

1.1.6 Documentation

2 Release Notes

2.1 Release details

2.1.1 Introduction

This release note is updated periodically to reflect the latest changes, issues, and limitations of **STM32CubeIDE 2.3.0**.

For the most recent release summary, see the table below.

STM32CubeIDE 2.3.0 release summary	
Type	Summary
Major release	• **Microcontroller, microprocessor, and board lists:** o Added support for the STM32G5 series. o Added support for the STM32H7P microcontrollers in the STM32H7 series. o Added support for the STM32C5+ microcontrollers in the STM32C5 series. • Added support for new boards: NUCLEO-G564RE, B-G564E-MC1S, B-G564E-DPOW1, STM32H7P5I-DK, and NUCLEO-H7P5II, NUCLEO-C5Q1ZG, NUCLEO-C5K1RC. • Online documentation is now available for STM32CubeIDE. • **Added support for:** o OpenSTLinux 7.0.0 through the update site mechanism o macOS® 27. o GCC toolchain through the update site mechanism. • **Main updated binaries:** o CMake to version 4.4.3. o ST Arm Clang to version 23.1.0. o Segger Jlink to version 9.78

2.1.2 What is new in STM32CubeIDE 2.3.0?

- Added support for new microcontrollers/ series :
  - o STM32G5 series with 256 Kbytes and 512 Kbytes of flash memory.
  - o STM32H7P microcontrollers with 2 Mbytes of flash memory in the STM32H7 series.
  - o STM32C5+ microcontrollers with 256 Kbytes and with 1 Mbytes of flash memory in the STM32C5 series.
- Added support for new boards:
  - o NUCLEO-G564RE, B-G564E-MC1S, B-G564E-DPOW1, STM32H7P5I-DK, NUCLEO-H7P5II, NUCLEO-C5Q1ZG, NUCLEO-C5K1RC.
- Added online documentation for STM32CubeIDE, including the Release Note, Installation Guide, Quick Start Guide, and Software Package License Agreement. Links to the FAQ, ST Community, and previous documents are also available.
- OpenSTLinux 7.0.0 is now available through the STM32CubeIDE update site. Upgrades to the Linux kernel and Yocto Project are also available.
- macOS ® 27 is fully supported in STM32CubeIDE.
- GCC 15 support: GCC 15 is now available through the STM32CubeIDE update site. Once installed, it can be selected for STM32CubeIDE projects to compile and link applications, configure compiler and linker options, and use the language support, diagnostics, and optimizations provided by GCC 15.

- CMake version in STM32CubeIDE has been updated to 4.4.3.
- Updated the ST ARM Clang toolchain to version 23.1.0. The update is available through the STM32CubeIDE update site.
- Segger Jlink version in STM32CubeIDE has been updated to 9.78.

2.1.3 Release information

2.1.4 Upgrades

Dependencies Upgrades ID	Summary
BusyBox	1.31.0.st_20240131-1810
Eclipse and CDT	Eclipse 2025-12 and CDT 12.3.0
GCC binaries	14.3.rel1.20251027-0700
JRE	Adoptium® Temurin™ 21.0.12.1 and JavaFX 21.0.12
Make	4.4.1_st_20260330-0700
CMake	4.4.3
Ninja	1.13.2
OpenOCD	0.12.0+dev-00669-g404a3c4b6
RTOS Proxy	0.18.0
SEGGER J-Link	9.78
ST Arm Clang	23.1.0+st.1
ST-LINKGDBServer	7.14.0
STLink Server	2.1.2-2 (for macOS installations) 2.1.1-1 (for other OS)
STM32CubeProgramme	2.24.0

2.1.5 Host PC system requirements

Supported operating systems and architectures

Supported operating systems and architectures

- Windows ® 10 and 11, 64 bits (x64)
- Linux ®: Ubuntu ® LTS 24.04, LTS 26.04, and Fedora ® 44
- macOS ® 26 (Tahoe), macOS ® 27 (Golden Gate).

Note:

- *Windows is a trademark of the Microsoft group of companies.*
- *Linux ® is a registered trademark of Linus Torvalds.*
- *Ubuntu ® is a registered trademark of Canonical Ltd.*
- *Fedora ® is a trademark of Red Hat, Inc.*
- *macOS ® is a trademark of Apple Inc., registered in the U.S. and other countries and regions.*

Memory and storage

- RAM: 4 Gbytes recommended
- Hard-disk space: 6 Gbytes of free space for non-STM32 MPU OpenSTLinux developers, 15 Gbytes for STM32 MPU OpenSTLinux usage

2.1.6 Customer support

For more information or help concerning STM32CubeIDE, contact the nearest STMicroelectronics sales office or use the ST community at community.st.com. For a complete list of STMicroelectronics offices and distributors, refer to the www.st.com webpage.

2.1.7 Software updates

Software updates and all the latest documentation can be downloaded from the STMicroelectronics support webpage at www.st.com/stm32cubeide.

2.1.8 General information

Overview

STM32CubeIDE is an integrated development environment (IDE) based on the Eclipse :sup:‘®‘framework. It is aimed at users developing embedded software in C/C++ for STMicroelectronics STM32 products. It uses an enhanced GNU toolchain for STM32, based on GNU Arm Embedded. STM32CubeIDE integrates the command-line version of STM32CubeProgrammer for flash memory operation and uses the ST-LINK GDB server for debugging. This enables programming and debugging of STM32 devices through standard debug interfaces such as JTAG and SWD.

Licensing

STM32CubeIDE is delivered under the SOFTWARE LICENSE AGREEMENT (“AGREEMENT”) (SLA0048).

The open-source and third-party software components used in the development of STM32CubeIDE and their licenses are listed in additional license terms for STM32CubeIDE file available from the product page on the STMicroelectronics www.st.com website.

The below Table provides the description of the licenses of additional components in STM32CubeIDE.

Complementary component licenses Name	Version	Copyright	License
STLink-USB-Driver		STMicroelectronics	
STLink-USB-Driver-lib		STMicroelectronics	
ST-LINK server (for macOS installations)	2.1.2-2	STMicroelectronics	Proprietary
ST-LINK server (for other OS)	2.1.1-1	STMicroelectronics	Proprietary

2.2 Release fixes

ID	Summary
TIDEC1-6000	Fixed an issue where adding a user-created file named main.c could cause a project build failure.
TIDEC1-6132	Updated STM32WB example projects with the required CubeIDE project nature, fixing peripheral HAL driver generation in the Example Selector.
TIDEC1-6165	Fixed an issue where LPTIM1 behaved incorrectly in Stop mode while debugging STM32U031 projects.
TIDEC1-6344	Updated STM32WB firmware to fix GCC 14 build errors in affected applications and examples.
TIDEC1-6347	Fixed an STM32Cube_FW_WB0 v1.5.0 issue causing some WB0 examples and applications to fail with GCC 14 due to an unsupported relocation.
TIDEC1-6907	Fixed a macOS issue that prompted users to open ST-LINK Server at debugger launch, even when already installed.
TIDEC1-6979	Updated GNU Tools for STM32 to fix incorrect repeated call stack frames when debugging FreeRTOS on STM32H5.
TIDEC1-7301	Fixed macOS failures in bundled JRE commands java –version and STLinkUpgrade.sh in STM32CubeCLT.
TIDEC1-7435	Updated the -mtune=cortex-a35 label to mark it incompatible with Cortex-M targets.
TIDEC1-7464	Updated STM32C5 SVD data to show all peripheral registers in the debugger.

2.3 Known limitations

2.3.1 Limitations

ID	Summary
TIDEC1-3618	To start debugging in the NS zone when product status is TZ-Closed for STM32H5 using ST-Link GDBServer, add this in **Run Commands**: `set breakpoint always-inserted on tbreak Reset_Handler monitor reset set breakpoint always-inserted off si` For **Reset & Restart**: add the same script and select `None` as the type.
TIDEC1-3756	Post-build commands can be restored by either applying code generation or changing the default behavior of **Restore Defaults**.
TIDEC1-3805	On STM32H745 Cortex-M4 dual-core projects, the predefined breakpoint is kept but not caught on run.
TIDEC1-3929	From GCC11, build errors may occur for: `_close`, `fstat`, `_getpid`, `_isatty`, `_kill`, `_lseek`, `_open`, `_read`, `_write`, `_gettimeofday`. In `main.c`, add weak stubs for missing functions (for example `_close`, `_lseek`, `_read`, `_write`).
TIDEC1-4297	For STM32CubeMP13 bare-metal + Azure ® RTOS ThreadX, add a `.stack` section in the linker script to avoid compilation issues.
TIDEC1-4465	Enum storage size is minimized by the toolchain to the declared value range.
TIDEC1-4585	STM32CubeIDE cannot auto-build after IP configuration changes. Flash size must be increased manually when needed.
TIDEC1-5161 / TIDEC1-5973	**Connect under reset** is not supported on STM32WB05/WB09/WL30/WL31/WL33. A hardware reset pulse is used instead.
TIDEC1-5220	For Cortex-M55 (for example STM32N6), GCC12 and earlier may warn with `-Ofast/-O3`. Workarounds: use GCC13, or compile with `-mcpu=cortex-m55+nomve`.
TIDEC1-5757	Importing STM32CubeMX projects into STM32CubeIDE on macOS/Ubuntu may show a pop-up or fail.
TIDEC1-5975	On STM32WBA65I-DK, incremental flashing is not supported for external flash; legacy mode is used.
TIDEC1-6327	With OpenOCD on STM32H7S78-DK and NUCLEO-H7S3L8, memory read failures may occur, but debugging remains possible.
TIDEC1-6338 / 6339 / 6341 / 6344 / 6347	With GCC14, some examples may show errors/warnings for STM32G0/H5/H7/H7RS/N6/WB/WB0/WBA/WL/WL3. Workaround: set GCC13.3.rel1 as default in **Window > Preferences > STM32Cube > Toolchain manager**.
TIDEC1-6311	For CMake projects, changing generator/build system may require **Clean Project** before rebuild.
TIDEC1-6318	CMake projects are not supported for MPx products.
TIDEC1-6658	Prebuild/postbuild scripts may use BusyBox tools (for example `wget`) with limitations; use controlled/safe URLs and files.
TIDEC1-6733	For Cortex-M0/M0+/M33/M55, core reset mode is not supported in STM32CubeIDE debug configuration.

2.3.2 General issues

ID	Summary
TIDEC1-353	Having a space or a non-ASCII character in the project/workspace/installation path is not fully supported.
TIDEC1-846	Hierarchical projects cannot be imported with **Copy into workspace**.
TIDEC1-866	Hierarchical projects cannot be renamed.
TIDEC1-1081	To debug an STM32CubeIDE Makefile project: right-click project -> Properties -> C/C++ Build -> MCU settings -> select requested MCU -> Apply.

continues on next page

Table 1 – continued from previous page

ID	Summary
TIDEC1-2004	Restart configuration works only for flash projects, not RAM projects. Manual workaround: in Restart configuration, set program counter to ResetHandler in RAM (example: `set $pc = 0x20000xxx`).
TIDEC1-3120	Editing live expressions while target is running does not work with SEGGER J-Link.
TIDEC1-3120 / TIDEC1-3929	GCC-10 is stricter regarding global variable declarations in headers. Read GCC-10 porting guide; warnings/errors may occur.
TIDEC1-3483	STM32CubeIDE cannot debug with FreeRTOS-aware debugging using J-LINK. Initialization commands workaround: `info threads`, `info threads`, `thread 2` (and `stepi` for attach mode).
TIDEC1-4051	Missing string table in Build Analyzer when displaying byte size of `.rodata`.
TIDEC1-4094	Issues detected when updating from 1.9.0 to 1.13.1. Recommended: reinstall 1.13.1 from scratch and reuse old workspace.
TIDEC1-4175	After programming in Run mode, STM32CubeIDE opens/focuses startup file tab every time.
TIDEC1-4420	embOS compile failure due to missing stack variable. Add linker section defining `_stack_start__` and `__stack_end__` when required by ThreadX.
TIDEC1-4883	Restore-default build configuration removes build commands.
TIDEC1-5170	Exported ZIP project can fail when imported in another workspace. Workaround: Import -> Projects from Folder or Archive.
TIDEC1-5194	Reset chip/restart debug session does not work as expected on Nucleo-N657X0-Q. Workaround: add `load` in Restart configuration additional commands.
TIDEC1-5405	In 1.17.0, HEX files are not generated automatically. Add post-build command: `arm-none-eabi-objcopy -O ihex "project_name.elf" "project_name.hex"`.
TIDEC1-5441	GCC 12.3 compilation error in 1.14.1. Workaround: download/load GCC12 locally (name `GCC12`, prefix `arm-none-eabi-`, location set to GCC12 plugin `tools\\bin` path).
TIDEC1-5853	GCC fallthrough attributes can be flagged as errors.
TIDEC1-6256	For CMake projects, add `-fstack-usage` for Static Stack Analyzer; add `-fcyclomatic-complexity` (GCC only) for Cyclomatic view in toolchain `.cmake` files. Cyclomatic is not supported with LLVM.
TIDEC1-6267	Debug may fail with `CMAKE_BUILD_TYPE=Release` because release maps to `-Oz` in `starm-clang.cmake`. Update release flag in toolchain file.
TIDEC1-6293	Multi-context CMake import outputs multiple ELF files. User must select the intended ELF at debug launch.
TIDEC1-6295	With GCC14, startup assembly may be malformed. Add: `.type LoopCopyDataInit, %function` and `.type LoopFillZerobss, %function`.
TIDEC1-6311	After importing some STM32CubeMX projects, changing CMake generator then reconfigure can cause unexpected build completion/failure.
TIDEC1-6335	On Ubuntu 22.04/macOS 15, opening `.ioc` may fail. Set STM32CubeMX as default editor for `*.ioc` in File Associations or Open With -> External Programs.
TIDEC1-6351	With LLVM, Static Stack Analyzer may be empty unless `-fstack-usage` is added. Cyclomatic view requires `-fcyclomatic-complexity` (GCC only), not supported with LLVM.
TIDEC1-6882	Some regenerated STM32CubeWBA example files may show warning icons. Remove warning-marked extra files before build.
TIDEC1-6909	On first launch after fresh install on macOS 15+, allow local network access popup for TCP/debug operations.
TIDEC1-6960	SWV output may not appear for some CMake projects. Manually set core clock in debug configuration to initial STM32CubeMX value.
TIDEC1-7059	Updating from older versions may fail when installed under system paths (for example `/opt/st/`) without required admin/sudo permissions.

continues on next page

Table 1 – continued from previous page

ID	Summary
TIDEC1-7121	For STM32U3 (256 KB flash), debug via ST-Link GDB server is not yet available; use OpenOCD probe.
TIDEC1-7144	For STM32CubeWL3 v1.5.0 apps on STM32WL33CC, OpenOCD flashing/debugging may fail in v2.1.1+; ST-Link GDB server works.
TIDEC1-7260	On Windows with the Hungarian keyboard layout, pressing AltGr+B in the editor may trigger Disable All Breakpoints instead of inserting `{`. This is an Eclipse key-binding issue. Workaround: go to Window > Preferences > General > Keys, search for Ctrl+Alt+B, select the binding, choose **Unbind Command**, then click **Apply and Close**.
TIDEC1-7408	CubeIDE may fail to start when OpenText UFT One sets the `_JAVA_OPTIONS` and `JAVA_TOOL_OPTIONS` environment variables. Workaround: temporarily unset both before launch; restore them if needed by UFT One.
TIDEC1-7437	The -ftls-model option is not effective with GCC 14 and earlier and is only partially supported with GCC 15. Full TLS support is expected with GCC 16.
TIDEC1-7617	In STM32CubeIDE 1.19.0 and later, a subsequent NUCLEO-L432KC debug launch with the ST-LINK GDB server may fail when option bytes select the PH3/BOOT0 pin for BOOT0 (`nSWBOOT0` enabled). The first debug launch may succeed. This issue has not been observed with OpenOCD. Workaround: In STM32CubeProgrammer's **Option Bytes** panel, disable `nSWBOOT0` and configure `nBOOT0` to boot from Main Flash. Alternatively, use OpenOCD as the debug probe.

2.3.3 STM32CubeCLT issues

ID	Summary
TIDEC1-6104	For macOS install, no paths for `CLANG_GCC_CMSIS_COMPILER` and `GCC_TOOLCHAIN_ROOT`. Workaround: • Open terminal. • Source `/etc/profile`. • Re-run the `echo` command on these two variables in the same terminal session.

2.3.4 Target-specific issues

STM32H7

ID	Summary
TIDEC1-1160	Serial Wire Viewer configuration is not reset for STM32H7 devices on next launch if terminated with record active when using ST-LINK GDB server.
TIDEC1-1484	ST-LINK GDB server reports suspended at address `0x05F0001` when debugging STM32H7 dual-core devices and the core under debug goes to sleep.
TIDEC1-1596	OpenOCD does not support debugging STM32H7 dual-core devices when the core under debug goes to sleep.
TIDEC1-4615	For H7RS project on Ubuntu with ExtMemLoader context configured, post-build commands cannot be passed successfully when STM32CubeIDE is installed with sudo. Workarounds: • Install STM32CubeIDE on Ubuntu without sudo (add execute permission to installer, then install without sudo). • If installed with sudo, run postbuild commands manually from command line with sudo permission.
TIDEC1-5713	While using GCC13 in STM32CubeIDE 1.17.0 or 1.18.0, errors or warnings can occur when importing some projects. GCC12 is recommended in this scenario.
TIDEC1-6274	Debugging code running from external flash on STM32H7S78-DK may fail with software breakpoints, causing `too many breakpoints` errors. Workaround: set breakpoint type to hardware manually, or ensure target is initialized in memory-mapped mode before debugging.

STM32L0

ID	Summary
TIDEC1-5487	Errors appear when building imported project `ADC_DMA_Transfer` for NUCLEO-L010RB. Workaround: add `syscall.c` to the project.

STM32MP1

ID	Summary
TIDEC1-1187	Debugging in production mode requires a network connection. If STM32CubeIDE uses a network proxy, add the STM32MP1 board IP address to the **Proxy bypass** list.
TIDEC1-1188	Synchronization check between launch configuration IP address and target IP address does not abort launch when they mismatch.
TIDEC1-5941	For STM32MP135F-DK (Bare Metal), when enabling FreeRTOS CMSIS_V2, assign RAM alias to `DDR_BASE` to avoid build errors. Workaround: • Open linker file `STM32MP135FAFX_RAM.ld`. • Edit: `REGION_ALIAS("RAM", SYSRAM_BASE);` -> `REGION_ALIAS("RAM", DDR_BASE);`

STM32MP2

ID	Summary
TIDEC1-7655	Existing MP2x TF-M projects may fail to debug in Production Mode if their linker scripts contain outdated memory region values. Update the linker script to match the CubeMX-generated memory regions.

STM32N6

ID	Summary
TIDEC1-5220	GCC12 and earlier versions detect warnings with `-Ofast` and `-O3` optimization levels. Workaround: • Use the default compiler (GCC14), or install GCC13 via update site. • Compile without MVE extension by passing `-mcpu=cortex-m55+nomve` in **Settings > GCC compiler > Miscellaneous**.
TIDEC1-5776	When using `XSPI_PSRAM_MemoryMapped` example (or similar), data stored in external memory may not be visible in STM32CubeIDE Memory View.
TIDEC1-5778	To read data in external memory map mode for STM32N6, use Memory View with Hex rendering: open the three-dot menu, then **Table rendering preferences**, and switch from **Automatic** to **Manual**.
TIDEC1-5864	If the Application context contains any breakpoint (even disabled), debug session does not start. Workaround: • Add breakpoints only while debugging, after executing: `retr = MapMemory();` • Set startup breakpoint at `JumpToApplication` instead of `main`.

STM32WB0

ID	Summary
TIDEC1-4786	A false error can be detected in some BLE applications that include binary libraries. Typical messages: `missing .note.GNU-stack section implies executable stack` or `NOTE: This behaviour is deprecated and will be removed in a future version of the linker`. Workaround: use linker option `-z noexecstack`. More information: GNU info note
TIDEC1-4887	An error seems to appear once the ThreadX option is activated for WB05/WB07/WB09.
TIDEC1-5161	For STM32WB0x devices, **Connect Under Reset** is not supported.

3 Quick Start Guide

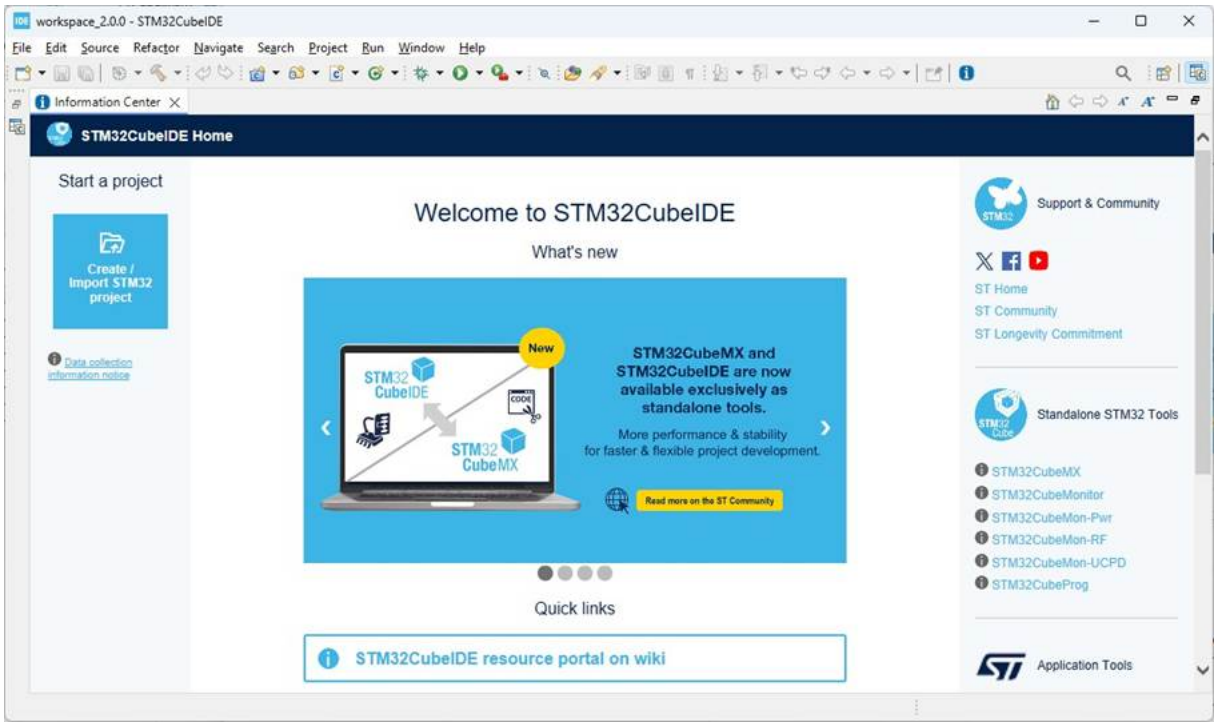
3.1 Introduction

This document is a brief guide for users to get started quickly with STMicroelectronics STM32CubeIDE. STM32CubeIDE is an integrated development environment (IDE) based on the Eclipse® framework. It is aimed at users developing embedded software in C/C++ for STMicroelectronics STM32 MCUs and MPUs. This manual provides basic information about:

- Information Center
- Workspace and projects
- Project information
- Debugging

3.2 Information center

When STM32CubeIDE is started the first time, the Information Center opens.



The Information Center enables the user to quickly reach information about the product and how to use it. By clicking on the corresponding hypertext links, manuals are opened and latest information is obtained from the STMicroelectronics website at www.st.com.. It is not required to read all material before using STM32CubeIDE for the first time. Rather, it is recommended to consider the Information Center as a collection of reference information to return to. The Information Center window can be reached at any time via the [Help]>[Information Center] command. When the Information Center tab is closed, the standard views in the C/C++ perspective are opened. STM32CubeIDE supports STM32 32-bit products based on the Arm® Cortex® processor.

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

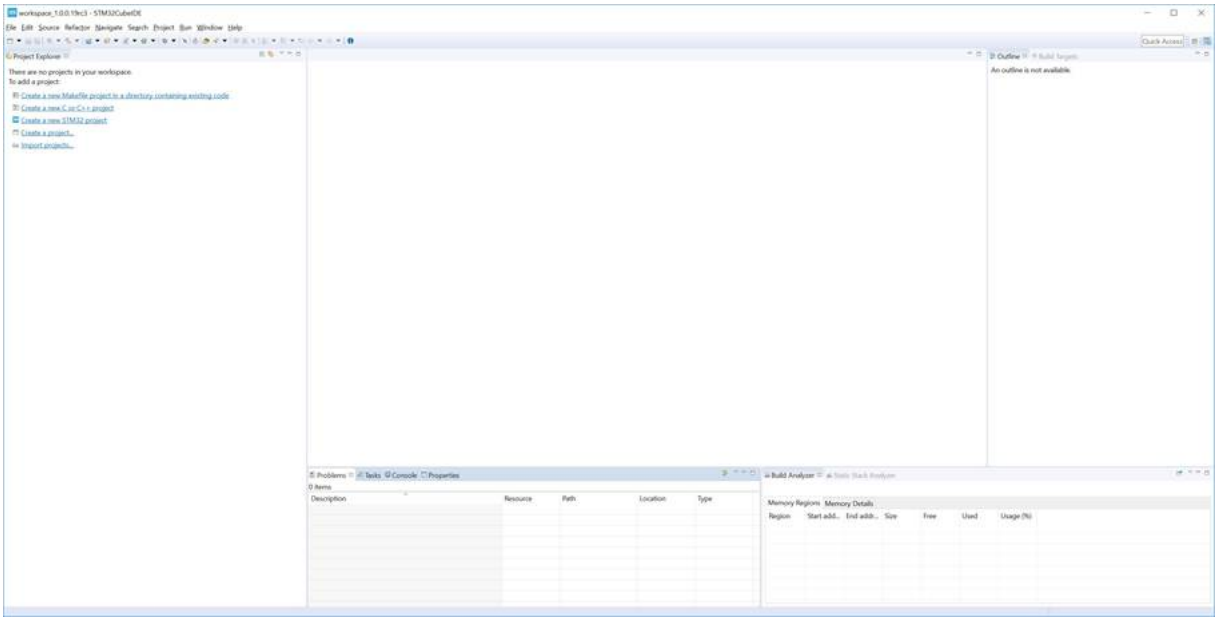
The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

3.3 Workspace and projects

STM32CubeIDE is built using the Eclipse ® framework; it inherits characteristics that might be unfamiliar to new users.

The Eclipse ® editing environment uses perspectives. A perspective is a set of windows dedicated to a purpose.

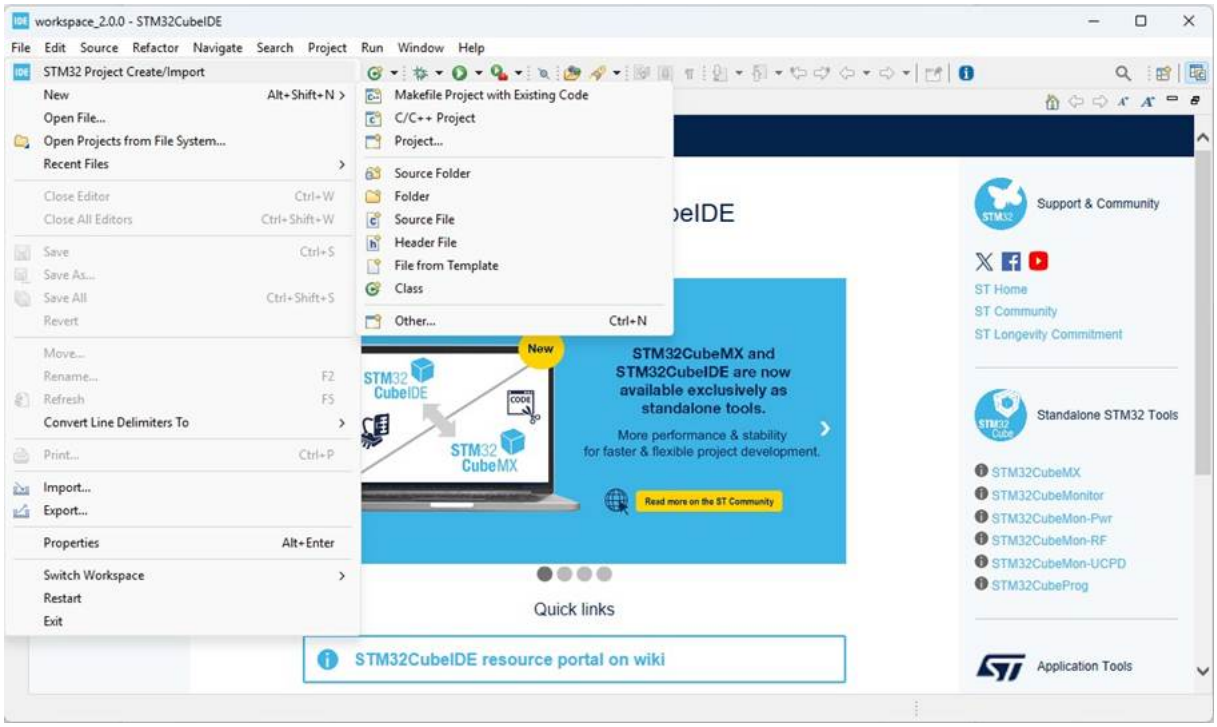
The C/C++ perspective is dedicated to writing and editing code, and navigating through projects. Users spend a lot of time in this perspective.



Another popular perspective is the Debug perspective, where users debug and test. Switching back and forth between perspectives is easy, and can even be automatic in some cases.

Workspace and projects

The next thing to do is create a project, which requires to introduce firstly the concept of workspace. A workspace is a container that includes project folders or information about project folders, and a .metadata folder that contains information about the projects. A workspace is simply a folder on a hard drive, which can be located anywhere in the storage media. When STM32CubeIDE starts up, it asks which workspace must be used. This may be changed at any time by selecting [File]>[Switch Workspace] and navigating to another folder.



Note: Workspace and project names must contain only ascii characters. This is also valid for the path to the workspace.

3.3.1 Project types in STM32CubeIDE

STM32CubeIDE enables users to:

- Create new projects from scratch for STM32 microcontrollers like empty projects or Empty CMake projects. This process allows full customization of the project configuration, including device selection, firmware package management, and peripheral initialization.
- Import existing projects, including those generated by STM32CubeMX or STM32CubeMX2, as well as projects using Makefile or CMake-based build systems.

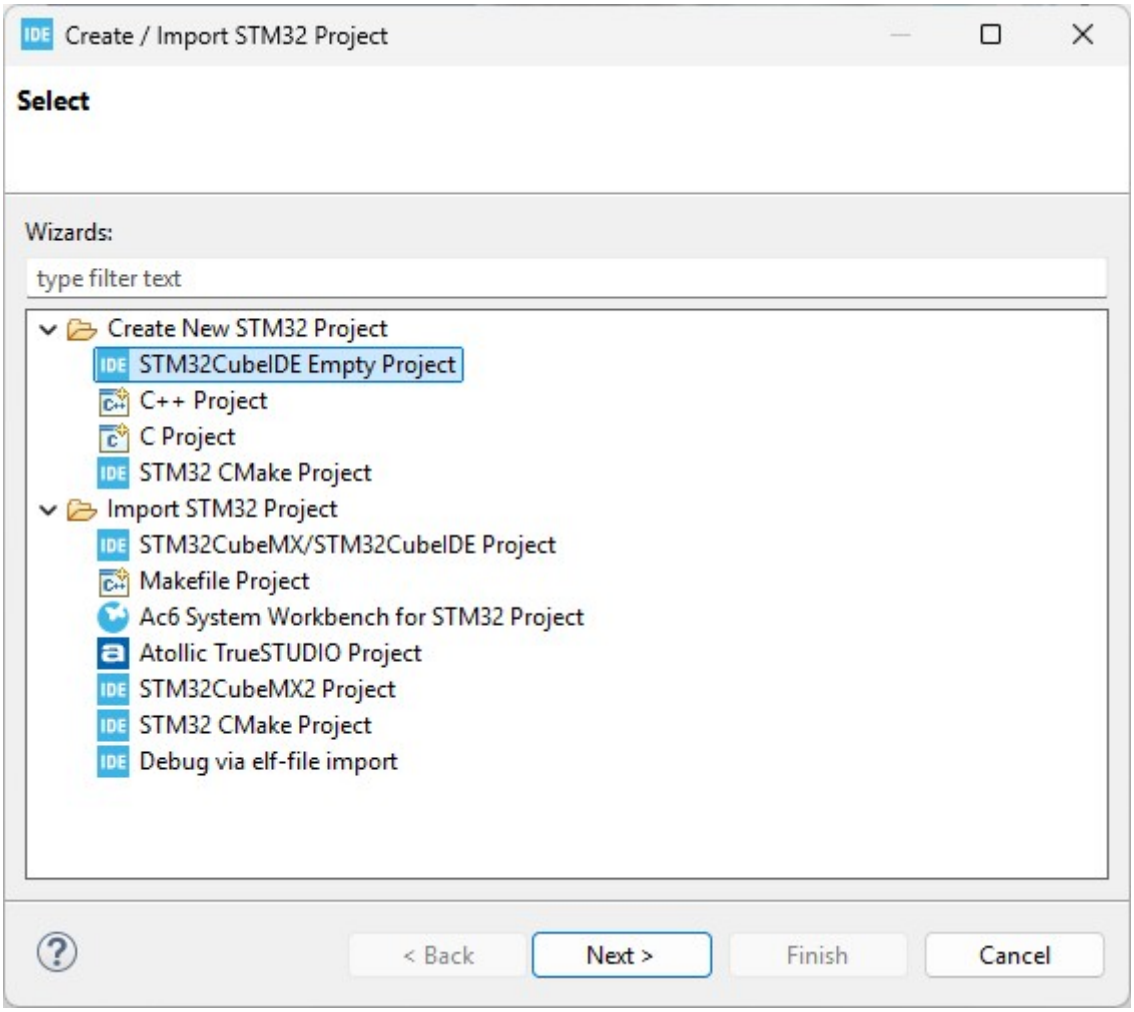
3.3.2 Start from scratch with STM32CubeIDE

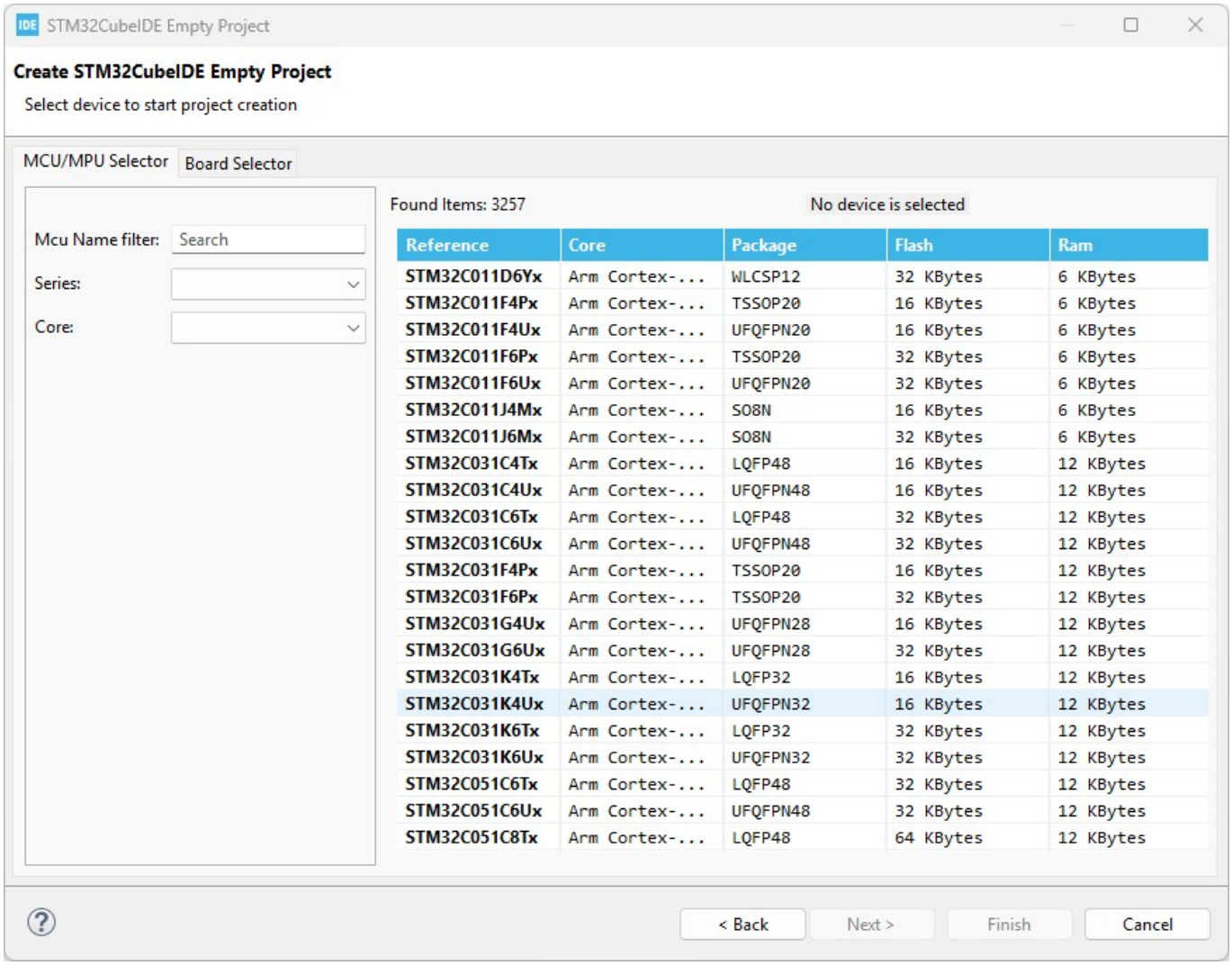
STM32CubeIDE allows users to create empty projects and empty CMake projects.

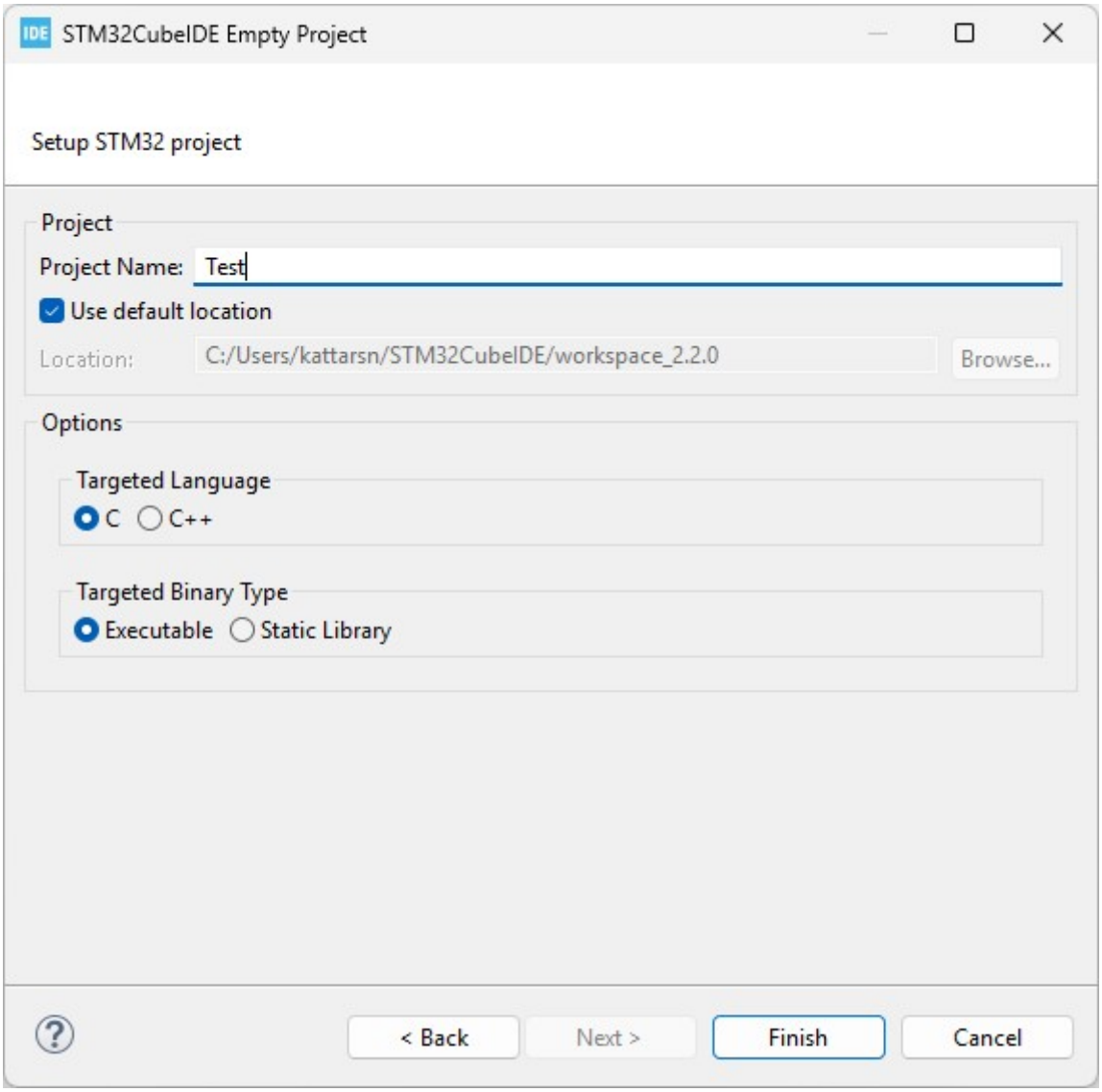
Start from scratch - STM32CubeIDE empty project

To create an empty project, proceed as follows:

1. Select the [[**File**]> [**STM32 Project Create/Import**]> [**Create New STM32 Project**]] command. The pop-up window is illustrated in Figure 4.
2. Select STM32CubeIDE project from the Create New STM32 Project menu.
3. Follow the wizard displayed to select MCU and enter the required data: project name, Targeted Language and Targeted Binary Type. This is illustrated in Figure 5.
4. Click on [Finish]. Illustrated in Figure 6.


Create / Import STM32 Project
Select
Wizards:
type filter text
Create New STM32 Project
STM32CubeIDE Empty Project
C++ Project
C Project
STM32 CMake Project
Import STM32 Project
STM32CubeMX/STM32CubeIDE Project
Makefile Project
Ac6 System Workbench for STM32 Project
Atollic TrueSTUDIO Project
STM32CubeMX2 Project
STM32 CMake Project
Debug via elf-file import
< Back
Next >
Finish
Cancel

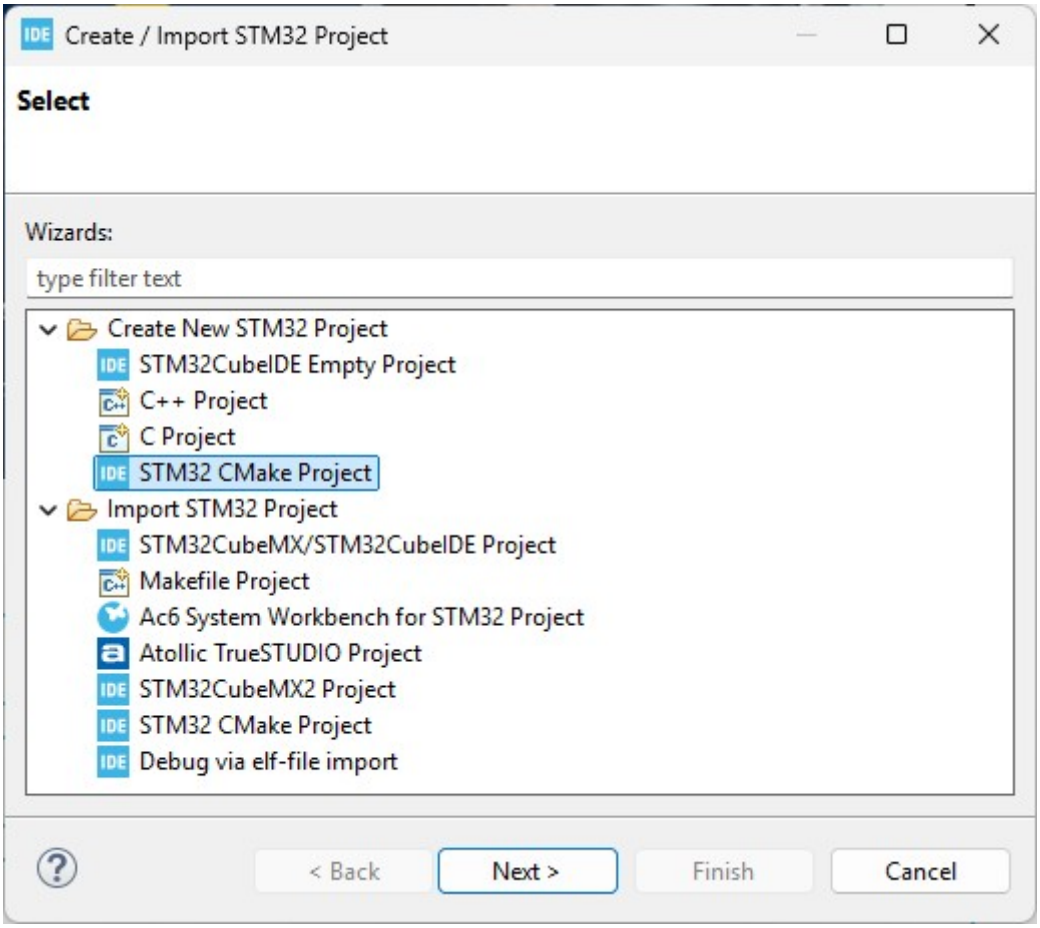

STM32CubeIDE Empty Project
Create STM32CubeIDE Empty Project
Select device to start project creation
MCU/MPU Selector
Board Selector
Mcu Name filter:
Search
Series:
Core:
Found Items: 3257
No device is selected
Reference
Core
Package
Flash
Ram
STM32C011D6Yx Arm Cortex-... WLCSP12 32 KBytes 6 KBytes
STM32C011F4Px Arm Cortex-... TSSOP20 16 KBytes 6 KBytes
STM32C011F4Ux Arm Cortex-... UFQFPN20 16 KBytes 6 KBytes
STM32C011F6Px Arm Cortex-... TSSOP20 32 KBytes 6 KBytes
STM32C011F6Ux Arm Cortex-... UFQFPN20 32 KBytes 6 KBytes
STM32C011J4Mx Arm Cortex-... SO8N 16 KBytes 6 KBytes
STM32C011J6Mx Arm Cortex-... SO8N 32 KBytes 6 KBytes
STM32C031C4Tx Arm Cortex-... LQFP48 16 KBytes 12 KBytes
STM32C031C4Ux Arm Cortex-... UFQFPN48 16 KBytes 12 KBytes
STM32C031C6Tx Arm Cortex-... LQFP48 32 KBytes 12 KBytes
STM32C031C6Ux Arm Cortex-... UFQFPN48 32 KBytes 12 KBytes
STM32C031F4Px Arm Cortex-... TSSOP20 16 KBytes 12 KBytes
STM32C031F6Px Arm Cortex-... TSSOP20 32 KBytes 12 KBytes
STM32C031G4Ux Arm Cortex-... UFQFPN28 16 KBytes 12 KBytes
STM32C031G6Ux Arm Cortex-... UFQFPN28 32 KBytes 12 KBytes
STM32C031K4Tx Arm Cortex-... LQFP32 16 KBytes 12 KBytes
STM32C031K4Ux Arm Cortex-... UFQFPN32 16 KBytes 12 KBytes
STM32C031K6Tx Arm Cortex-... LQFP32 32 KBytes 12 KBytes
STM32C031K6Ux Arm Cortex-... UFQFPN32 32 KBytes 12 KBytes
STM32C051C6Tx Arm Cortex-... LQFP48 32 KBytes 12 KBytes
STM32C051C6Ux Arm Cortex-... UFQFPN48 32 KBytes 12 KBytes
STM32C051C8Tx Arm Cortex-... LQFP48 64 KBytes 12 KBytes
< Back
Next >
Finish
Cancel

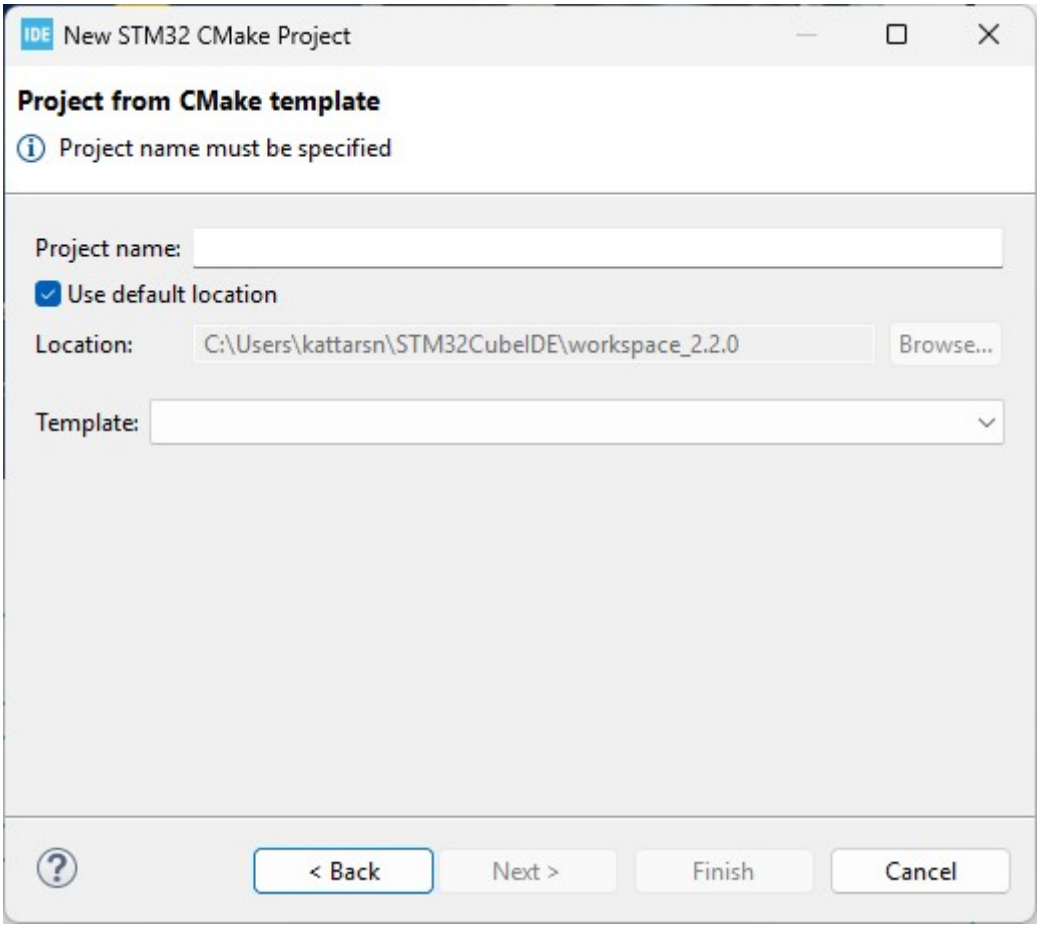


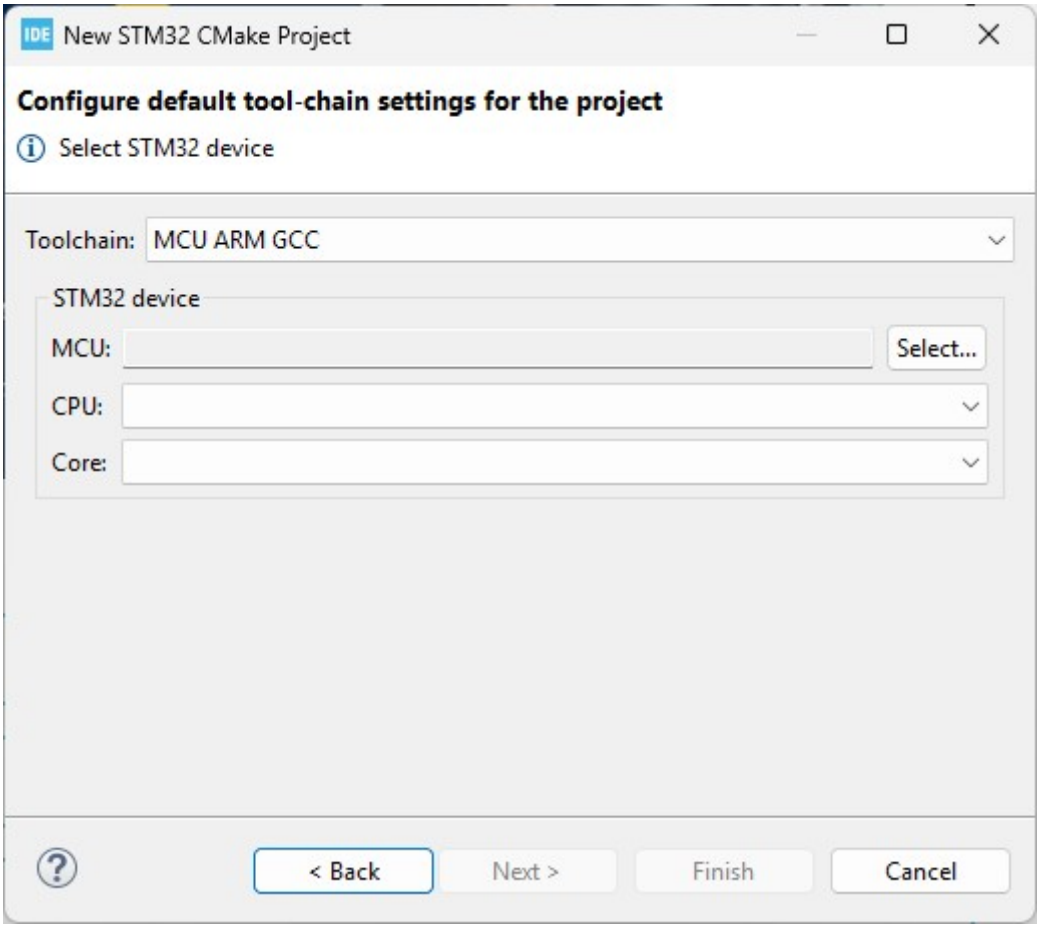
Start from scratch – STM32CubeIDE empty CMake project creation

To create an empty CMake project, proceed as follows:

1. Select the **[File]> [STM32 Project Create/Import]>[Create New STM32 Project]** command.
2. Select STM32 CMake project as illustrated in Figure 7.
3. Follow the wizard displayed to enter the following information as illustrated in Figure 8:
   - Project name.
   - Select template such as Executable or Static Library.
   - Select the required MCU.
4. Click on [Finish]. Illustrated in Figure 9.


Create / Import STM32 Project
Select
Wizards:
type filter text
Create New STM32 Project
STM32CubeIDE Empty Project
C++ Project
C Project
STM32 CMake Project
Import STM32 Project
STM32CubeMX/STM32CubeIDE Project
Makefile Project
Ac6 System Workbench for STM32 Project
Atollic TrueSTUDIO Project
STM32CubeMX2 Project
STM32 CMake Project
Debug via elf-file import
< Back
Next >
Finish
Cancel


New STM32 CMake Project
Project from CMake template
Project name must be specified
Project name:
Use default location
Location:
C:\Users\kattarsn\STM32CubeIDE\workspace_2.2.0
Browse...
Template:
< Back
Next >
Finish
Cancel



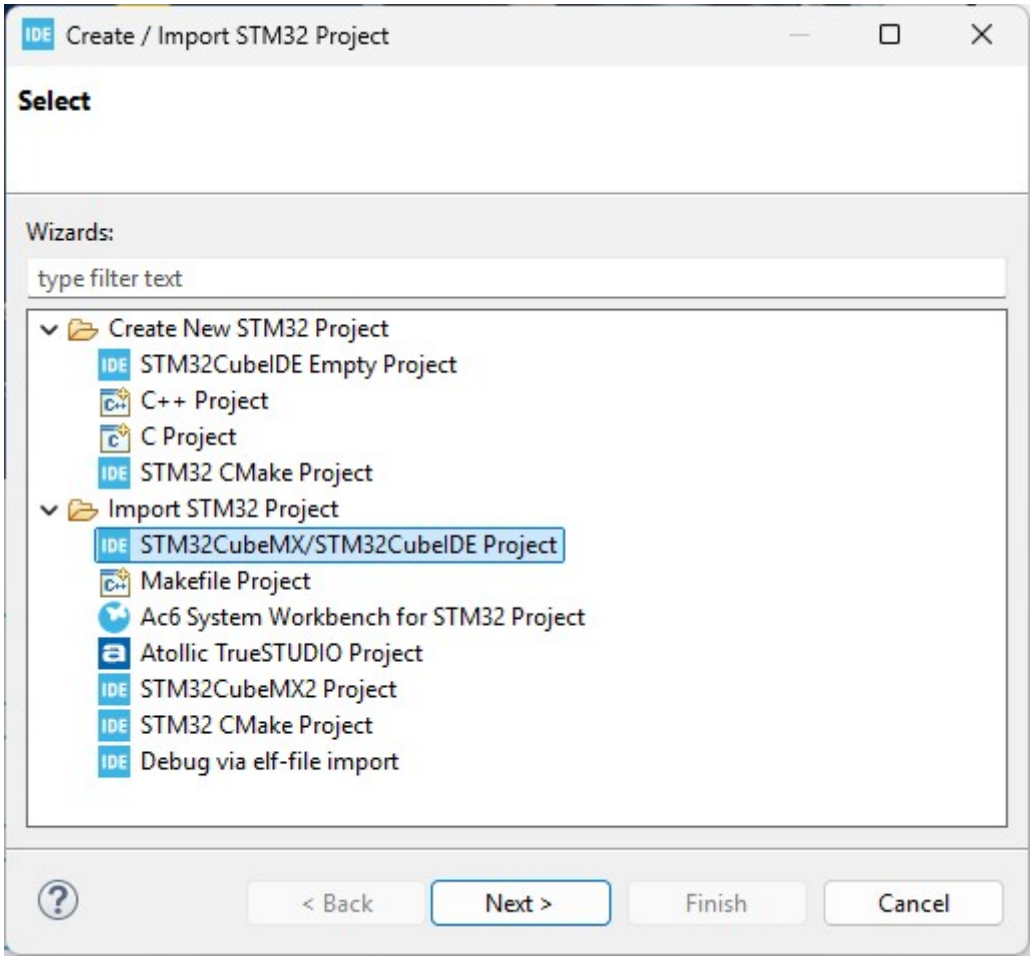
3.3.3 Start from existing project with STM32CubeIDE

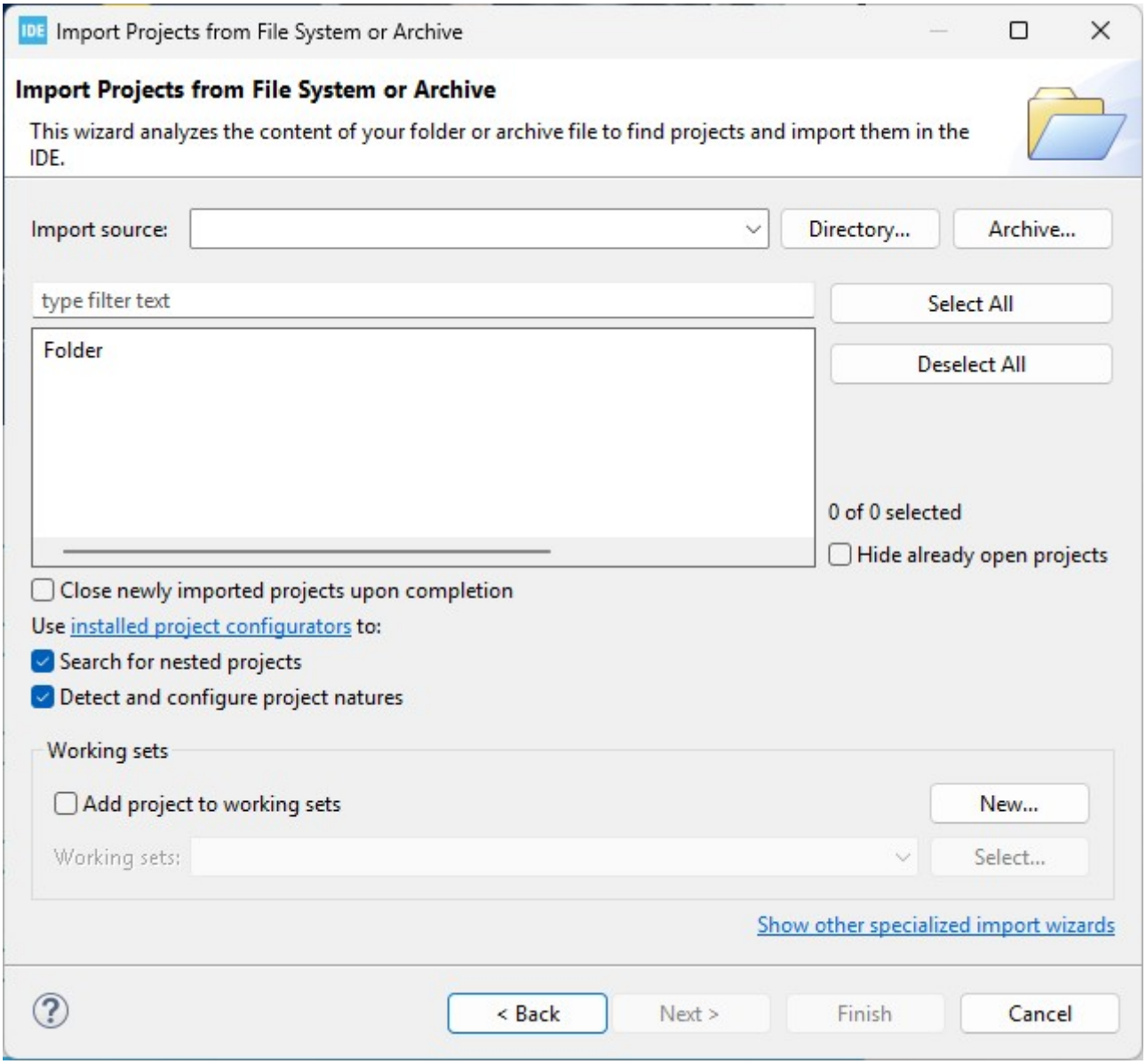
STM32CubeIDE allows users to import projects generated by STM32CubeMX or STM32CubeMX2.

Start from existing project: Import STM32CubeMX or STM32CubeMX2 project

To import an STM32CubeMX project use the STM32 project wizard and follow the steps below:

1. Select the [[File]> [STM32 Project Create/Import]>[Import STM32 Project]] menu command. The wizard is illustrated in Figure 10
2. Select STM32CubeMX or STM32CubeMX2 project
3. Follow the wizard instructions displayed to enter the requested data:
   - Project name
   - STM32CubeMX directory location. Illustrated in Figure 11.
4. Click on [Finish].


Create / Import STM32 Project
Select
Wizards:
type filter text
Create New STM32 Project
STM32CubeIDE Empty Project
C++ Project
C Project
STM32 CMake Project
Import STM32 Project
STM32CubeMX/STM32CubeIDE Project
Makefile Project
Ac6 System Workbench for STM32 Project
Atollic TrueSTUDIO Project
STM32CubeMX2 Project
STM32 CMake Project
Debug via elf-file import
< Back
Next >
Finish
Cancel

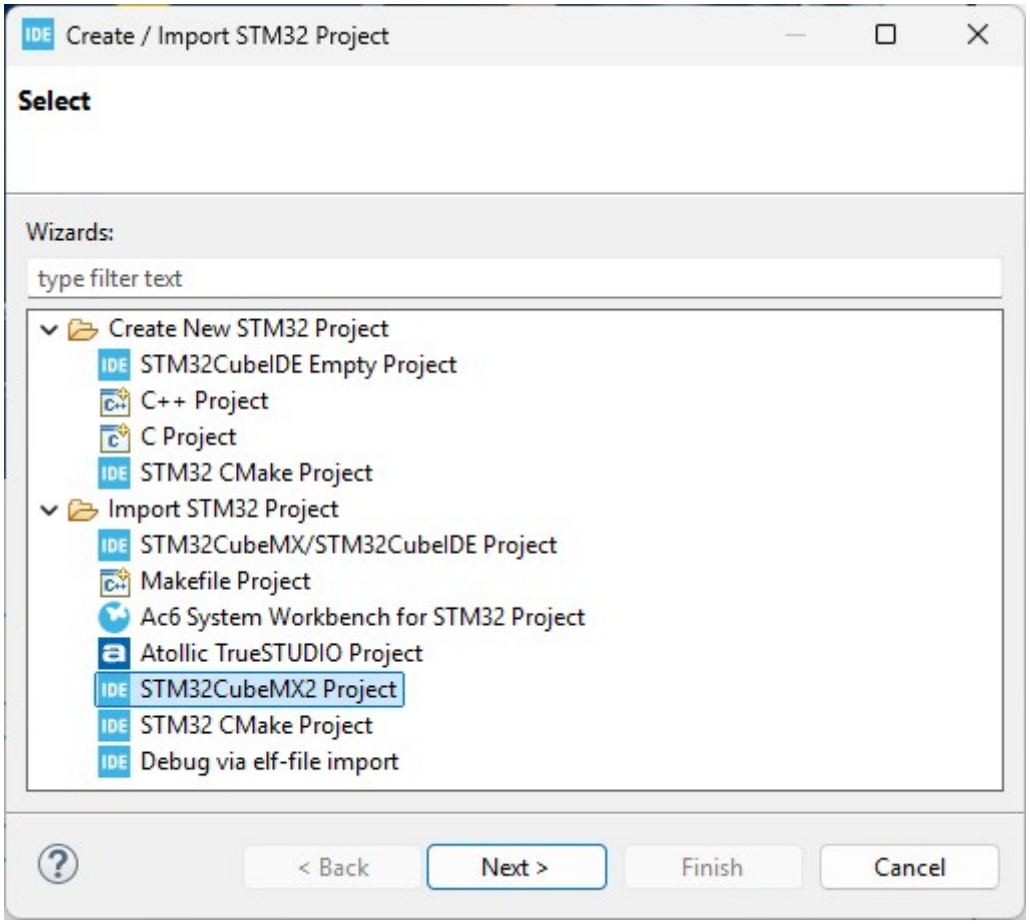


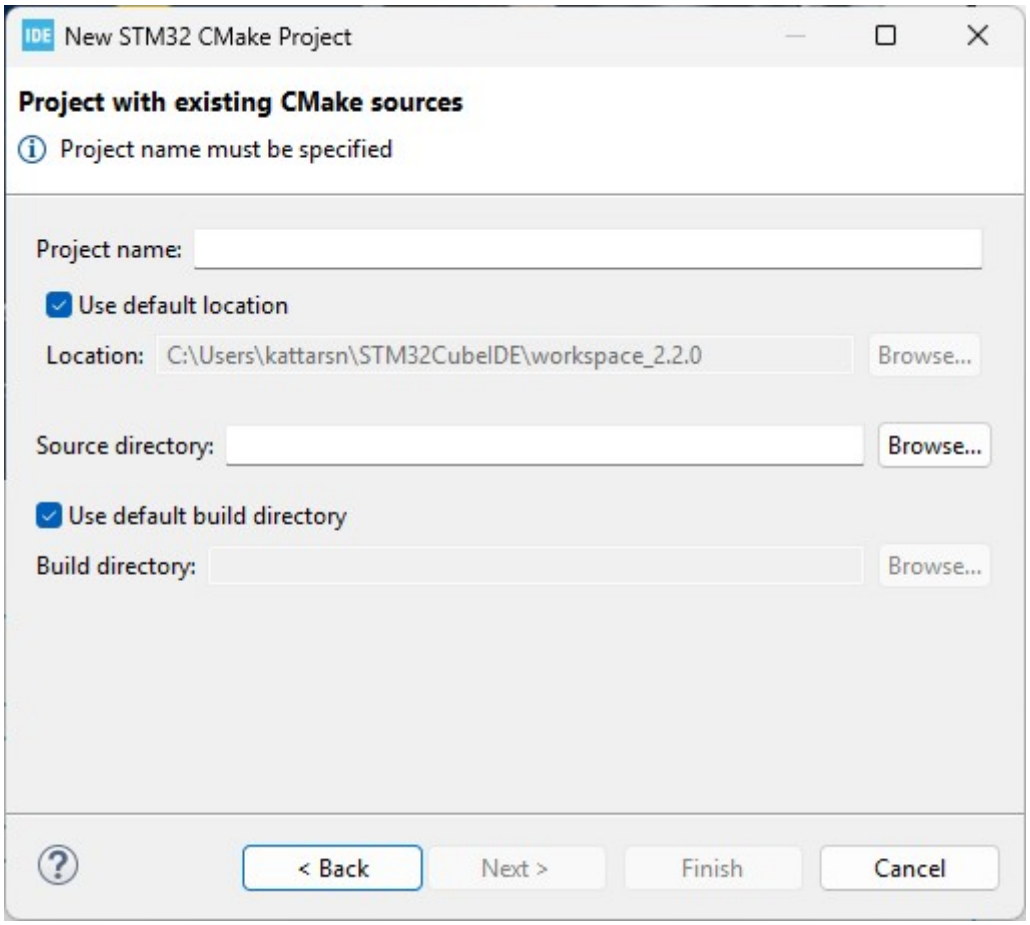
New projects can also be created through [[**File**]>[**Import STM32 Project**]> **STM32CubeMX/STM32CubeIDE**] menu command.

Start from existing project: Import STM32CubeMX2 or STM32CubeIDE project

To import an STM32CubeMX2 project use the STM32 project wizard and follow the steps below:

1. Select the [[**File**]> [**STM32 Project Create/Import**]> [**Import STM32 Project**]] menu command. Illustrated in Figure 12.
2. Select STM32CubeMX2 project.
3. Follow the wizard displayed to enter the requested data, as illustrated in Figure 13:
   - Project name
   - STM32CubeMX2 directory.
4. Click on [**Finish**].


Create / Import STM32 Project
Select
Wizards:
type filter text
Create New STM32 Project
STM32CubeIDE Empty Project
C++ Project
C Project
STM32 CMake Project
Import STM32 Project
STM32CubeMX/STM32CubeIDE Project
Makefile Project
Ac6 System Workbench for STM32 Project
Atollic TrueSTUDIO Project
STM32CubeMX2 Project
STM32 CMake Project
Debug via elf-file import
< Back
Next >
Finish
Cancel


New STM32 CMake Project
Project with existing CMake sources
Project name must be specified
Project name:
Use default location
Location:
C:\Users\kattarsn\STM32CubeIDE\workspace_2.2.0
Browse...
Source directory:
Browse...
Use default build directory
Build directory:
Browse...
< Back
Next >
Finish
Cancel

New projects can be created as well through [**[File]>[Import STM32 Project]> STM32CubeMX2**] project menu command.

Project importing can also be initiated through the added icon in the toolbar illustrated in Figure 14.



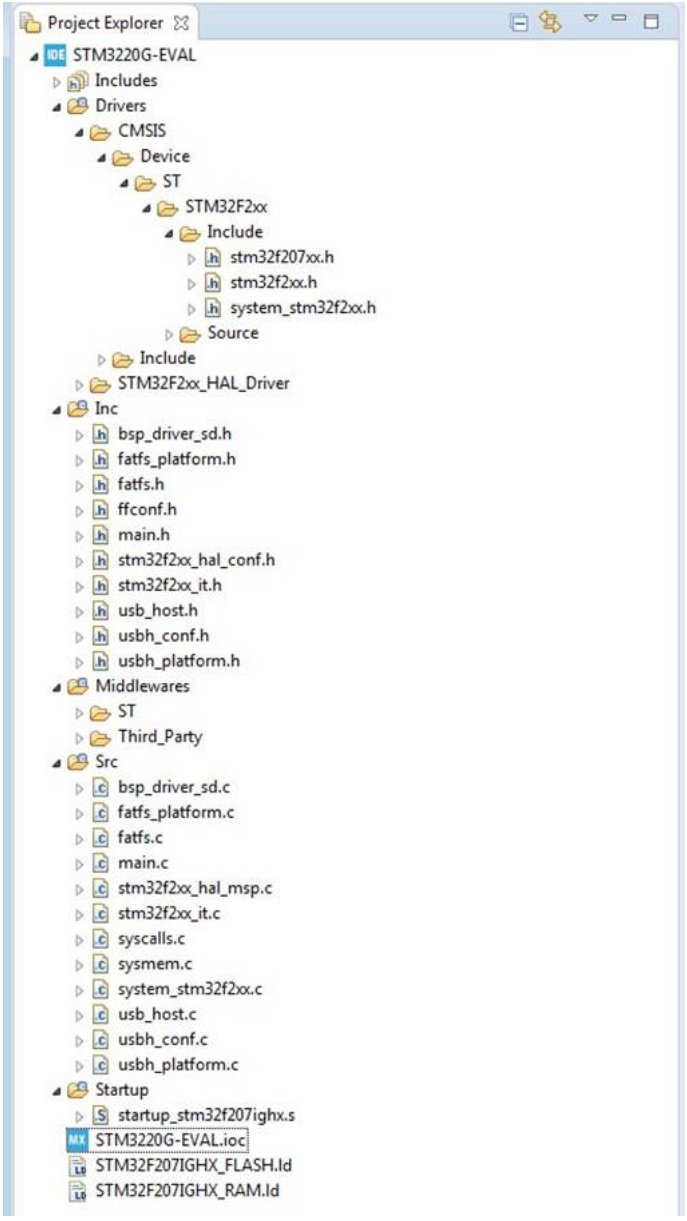
Please refer to the following links for an initiation with STM32CubeIDE

- Getting Started with STM32Cube Ecosystem
- Introduction to STM32CubeIDE

3.4 Project information

Once the information is entered in the wizard, it is possible to work in the project itself:

- The Project Explorer docking view is populated with the selected project. In the Project Explorer window, the subfolders can be expanded to any desired sublevel.
- The kind of project created by the Project Wizard depends on the device family. In any case, the project contains all the necessary elements of a complete embedded application, ready to program into the memory of the part and run.
- When several projects exist in a workspace, it is possible to close some of them. The actions controlled by the icons such as run, or launch debug session then apply to the open projects.
- Projects can be closed or opened by first right-clicking on the project name in the Project Explorer docking view and then selecting [Open Project] or [Close Project].



Controls

All control icons have tool tips activated by hovering the mouse over the icon. The icons are shortcuts for the same functions that can be activated from the main menu bar. These icons control specific functions that pertain to code editing, building, and project management; they are unique to the C/C++ perspective.





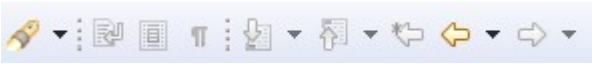
Use this icon to create new C source code module, header files, or new object such as project, library, or repository ([**File**]>[**New**] from the main menu options).



Use this icon to build your project.



Use this icon to launch a specific debug configuration, or configure a debug configuration by clicking on the arrow (Functions can be activated in the [**Run**] option in the main menu).



The flashlight icon launches various search utilities; the arrows navigate among recently visited places in the project ([**Search**] and [**Navigate**] in the main menu options).

3.5 Debugging

Once the project is built without errors, use the debeug icon to launch the program code into the part together with the debugger. The first time a debug session is launched, STM32CubeIDE builds the project and then shows the debug launch configuration menu. This gives the user the opportunity to verify settings and make changes if desired.

Important: If the Serial Wire Viewer (SWV) is to be used during debugging, this must be explicitly enabled in the dialogue box:

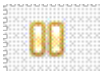
1. Click on the [Debugger] tab to reach the SWV settings and enable SWV (SWV is not enabled by default).
2. If changes are made to the default settings in the launch configuration properties dialog box, they must be saved by clicking on the [**Apply**] button.

When ready to launch the debug session, click on [**OK**]. Several things happen “under the hood” including the launch of the debugger driver and GDB server, as well as the programming of the application into the part, and any other actions called for in the startup script. At this point, the application is normally halted at the first line of main().

The main control icons of the debugger are as follows:



Resume full speed execution of the application on the target (greyed out when running).



Halt execution (greyed out when halted).



Step into a function, over a function, or out of a function.



Toggle between C and instruction stepping.



Reset the chip and restart execution.



Terminate the debug session.



Terminate and relaunch the debug session.

Basic debugging techniques

- To set a breakpoint, click on the blue horizontal bar next to the statement line number.
- To view memory locations, use the Variables (variables in focus or global variables), Memory, or Expressions window.

Configuring and using debugging tools, especially with SWV, is frequently more complicated than using the IDE for C/C++ code development. More details on this topic are provided by associated documents available on the STM32CubeIDE web page, which must be consulted for anything beyond the most basic operations.

4 Installation Guide

4.1 Introduction

This installation guide for STM32CubeIDE gives directions on how to install software on each of the operating systems it supports. It is primarily intended for software developers or system administrators who are about to install the STM32CubeIDE product.

This installation guide covers the following topics:

- System requirements
- Important information
- STM32CubeIDE installation (Windows ®)
- STM32CubeIDE installation (Linux ®)
- STM32CubeIDE installation (macOS ®)
- Update an STM32CubeIDE installation
- Uninstall STM32CubeIDE (Windows ®)
- Uninstall STM32CubeIDE (Linux ®)
- Uninstall STM32CubeIDE (macOS ®)

4.2 System requirements

STM32CubeIDE
is tested and verified on the Microsoft ® Windows ®, Linux ®, and macOS ® versions listed in this chapter.

Important: Only 64-bit OS versions are supported.

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

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.

4.2.1 Microsoft® Windows®

- Microsoft® Windows® 10
- Microsoft® Windows® 11

  Note: Microsoft and Windows are trademarks of the Microsoft group of companies.

4.2.2 Linux®

- Ubuntu® LTS 24.04, and Ubuntu® LTS 26.04
- Fedora® 43, and Fedora® 44.

  Note: Linux® is a registered trademark of Linus Torvalds.

  Ubuntu® is a registered trademark of Canonical Ltd.

  Fedora® is a trademark of Red Hat, Inc.

4.2.3 MacOS®

- macOS® 15 (Sequoia)
- macOS® 26 (Tahoe)

Note: macOS® is a trademark of Apple Inc., registered in the U.S. and other countries and regions.

All other trademarks are the property of their respective owners.

4.2.4 Hardware requirements

The following hardware requirements apply:

- 2 Gbytes of RAM minimum. 4 Gbytes of RAM recommended
- 6 Gbytes of free hard-disk space for non STM32 MPU OpenSTLinux Distribution developers, 15 Gbytes for STM32 MPU OpenSTLinux Distribution usage

4.3 Important information

This chapter contains important information regarding the installation of STM32CubeIDE.

4.3.1 Product installer

The latest version of the STM32CubeIDE installer can be downloaded from the STMicroelectronics web site at www.st.com.

4.3.2 Installing from USB memory

It is not recommended to launch the STM32CubeIDE installer directly from a USB memory. Instead, copy the executable installation file from the USB memory to the local hard-disk drive of the computer and execute the installation from the hard-disk drive.

If an installation from a USB memory is still preferred, make sure that the USB memory is not write-protected and that there is at least 6 GBytes of free memory beyond the space occupied by the installation executable. The extra space is required for temporary files during the installation.

Caution: *Do not remove the USB memory from the computer until the installation process is completely finished, or the installation will fail.*

4.3.3 Product upgrades

It is possible to install new versions of STM32CubeIDE in parallel with older versions.

4.4 STM32CubeIDE installation (Windows®)

This section describes how to install the STM32CubeIDE product on Microsoft® Windows®.

Note: The installation is done through a product installer. Make sure that the user account, from which the installer is launched, has administrative privileges.

Proceed as follows:

1. Launch the product installer (`.exe` file)

`stm32cubeide_VERSION-Win-ARCHITECTURE.exe`

where:

– `VERSION` is the actual product version and build date. Example: `2.2.0_2026_20190221_1309`

– `ARCHITECTURE` is the architecture of the target host computer to run STM32CubeIDE. Example: `x86_64`

If, when launching the product installer, it reports an attempt to install a version that is already installed:

a. Launch the register dialog by typing regedit.exe in the search bar

b. Remove registry key

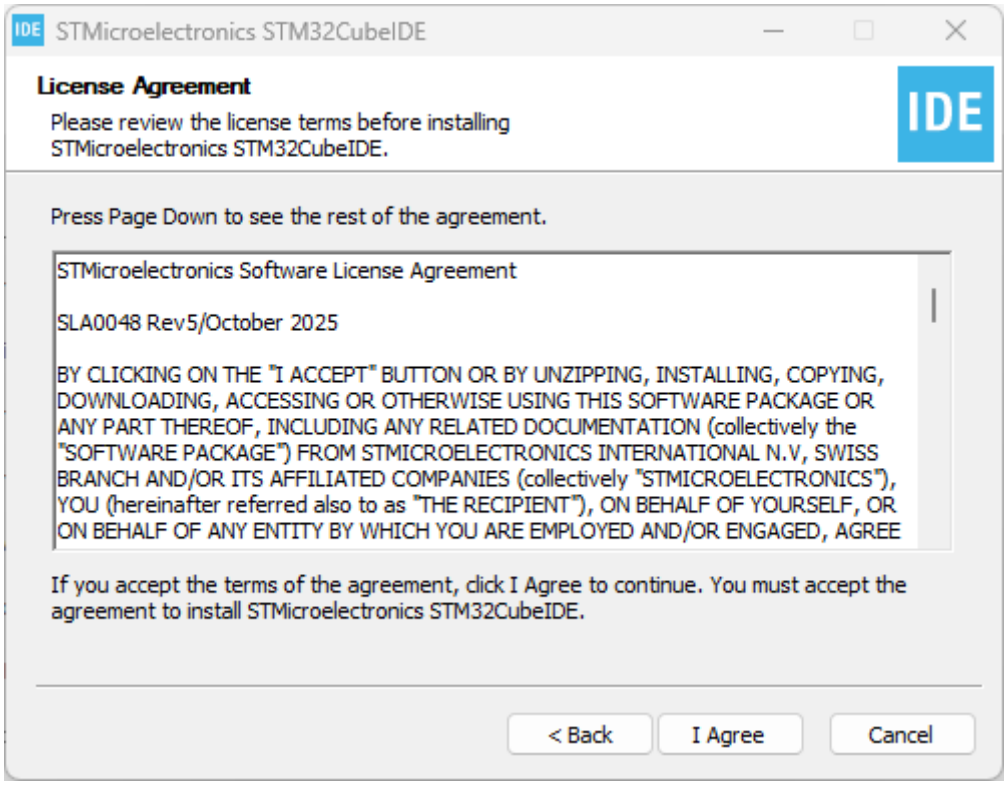
`HKEY_LOCAL_MACHINE\\SOFTWARE\\WOW6432Node\\STMicroelectronics\\STM32CubeIDE`

2. During the installation process, the operating system may display a dialog stating: `"Do you want to allow this app to make changes to your device?"` with info `"Verified publisher: STMicroelectronics Software AB"`. Accept ([**YES**]) to let the installer continue.

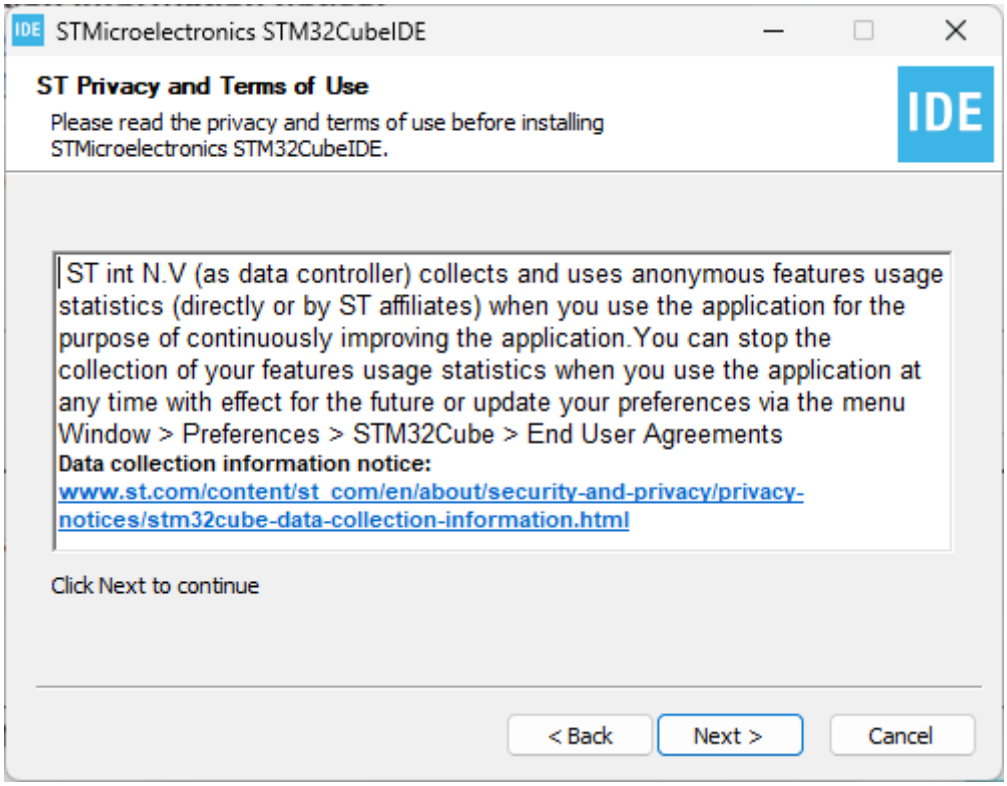
3. Wait for the installer **Welcome** dialog to be displayed and click on [Next >].



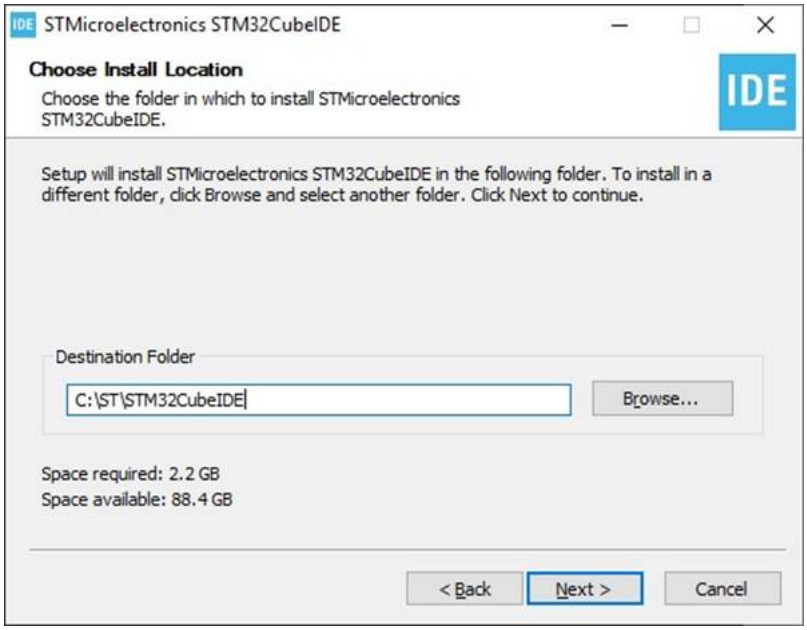
4. Read the license agreement. Click [**I Agree**] to accept its terms, or [**Cancel**] to abort the installation. If you accept the agreement, the installation wizard continues.



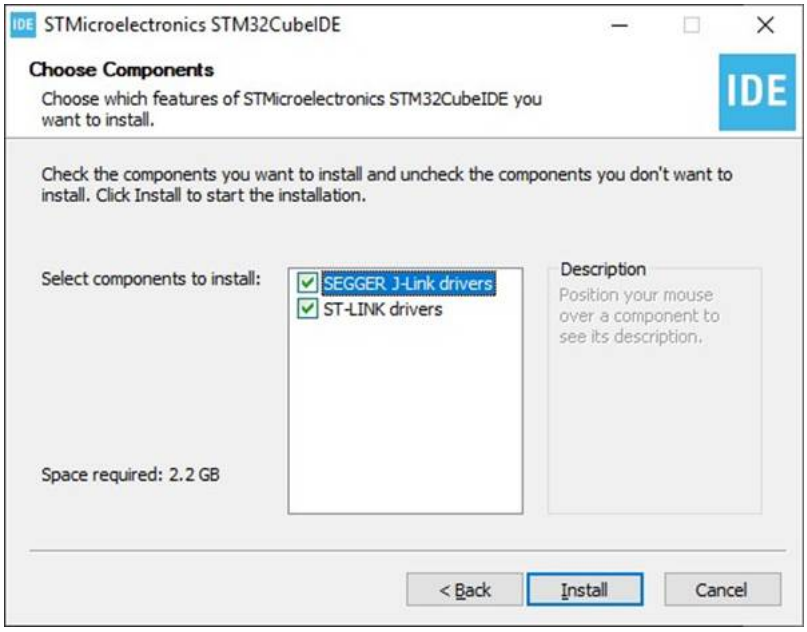
5. Review the privacy notice and terms of use, then click [**Next >**].



6. Select the installation location. Choose a short path to avoid Windows path-length limitations.



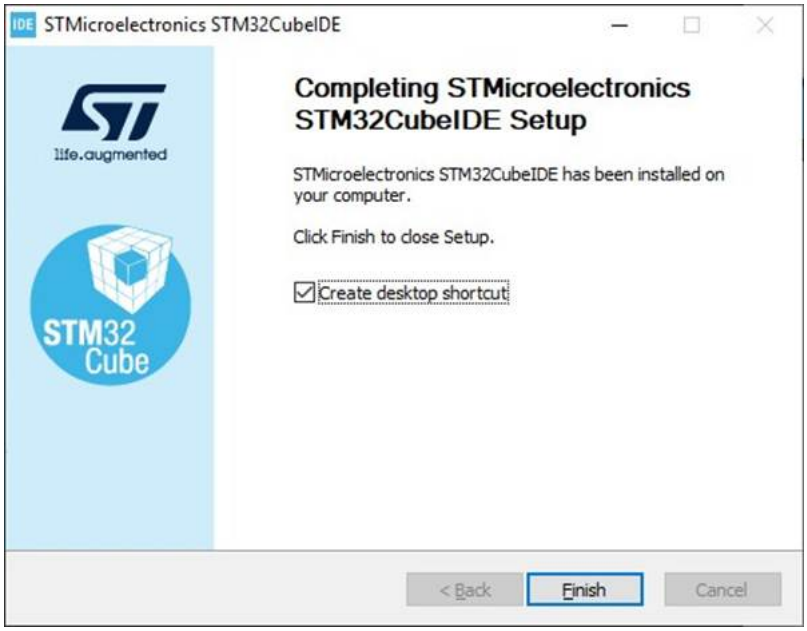
7. Wait for the Choose Components dialog to be displayed. Select the GDB Server components to be installed together with STM32CubeIDE.A server is needed for each type of JTAG probe used for debugging with STM32CubeIDE.



8. Click on [Install] to start the installation. The drivers that were selected are installed in parallel with this installation of STM32CubeIDEfrom here on.

docs/Assets/Installation_Guide/image0033.jpg

9. Click on [Next] to continue to the final step of the installation process. That is a Confirmation dialog informining the user that the installation is finished. Once the user clicks on [Finish], the installation process in complete.



4.5 STM32CubeIDE installation (Linux®)

This section describes how to install the STM32CubeIDE product on Linux®.

Note: The installation is done through a product installer. Linux® root privileges are required to complete

the installation.

The installer comes in different bundles to suit the various Linux® distributions. The bundles are named according to:

`stm32cubeide_VERSION-Lin-PACKAGE-ARCHITECTURE.sh` where:

-`VERSION` is the actual product version and build date. Example: `2.2.0_2026_20190221_1309`

-`ARCHITECTURE` is the architecture of the target host computer to run STM32CubeIDE. Example: `x86_64`

-PACKAGE is the Linux® package type to be installed. The supported packages are:

– RPM for Fedora®/CentOS™, for example: `Lin-RPM-x86_64.sh`

– Deb for Ubuntu®, for example: `Lin-Deb-x86_64.sh`

– No package to add for generic Linux®, for example: `Lin-x86_64.sh` Proceed as follows:

1. Navigate to the location of the installer file with a command console on the host computer.
2. Enter the following command in the console window:

   `sudo sh./st-stm32cubeide_VERSION_ARCHITECHURE.PACKAGE` where: `-VERSION`, `ARCHITECTURE` and `PACKAGE` must be entered after the selected Linux® package.
3. Follow the further instructions provided through the console window.

Manual installation (RPM/Deb)

For RPM-based distributions (Red Hat®, CentOS™, SUSE®, Fedora®): `sudo sh./stm32cubeide_2.2.0_xxxx-Lin-Deb-x86_64.sh`

For Debian-based distributions (Debian®, Ubuntu®):

```
sudo sh./stm32cubeide_2.2.0_xxxx-Lin-Deb-x86_64.sh
```

4.6 STM32CubeIDE installation (macOS®)

This section describes how to install the STM32CubeIDE product on macOS®.

Note: The installation is done through a product installer. Make sure that the user account, from which the installer is launched, has administrative privileges.

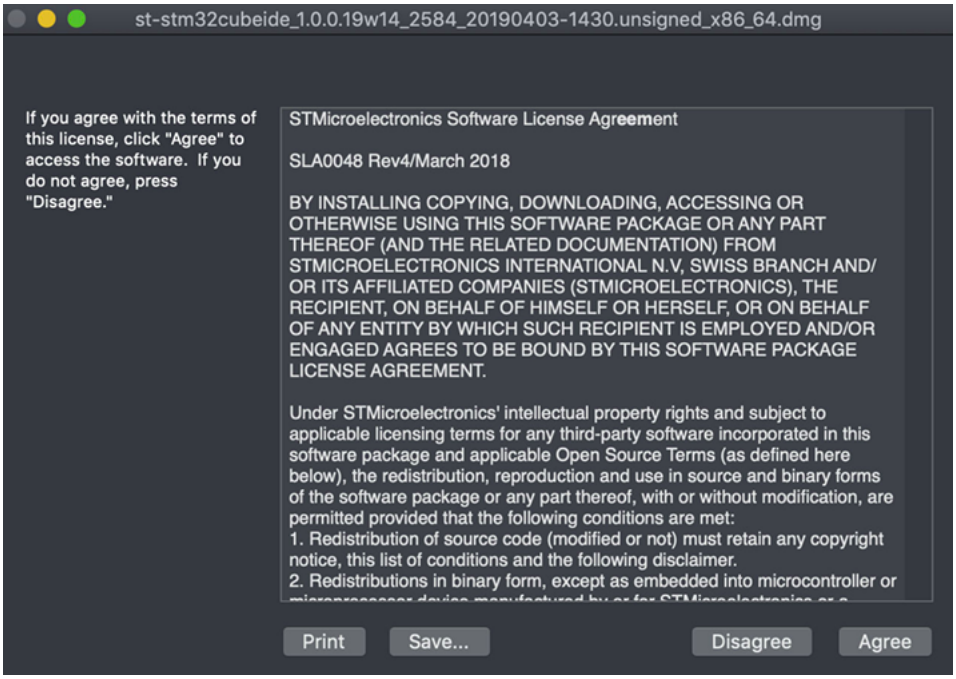
Proceed as follows:

1. Launch the product installer (`.dmg` file) `stm32cubeide_VERSION-Mac-ARCHITECTURE.dmg` where:

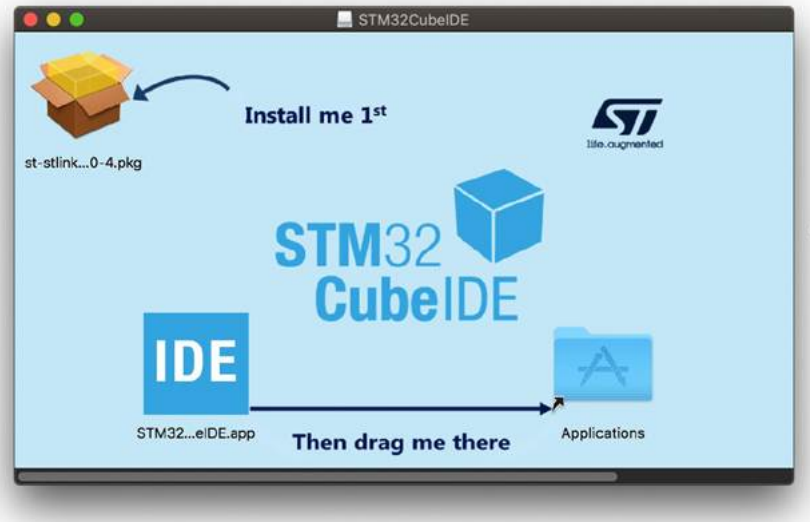
   –`VERSION` is the actual product version and build date.

   Example:`2.2.0_2026_20190221_13091309`

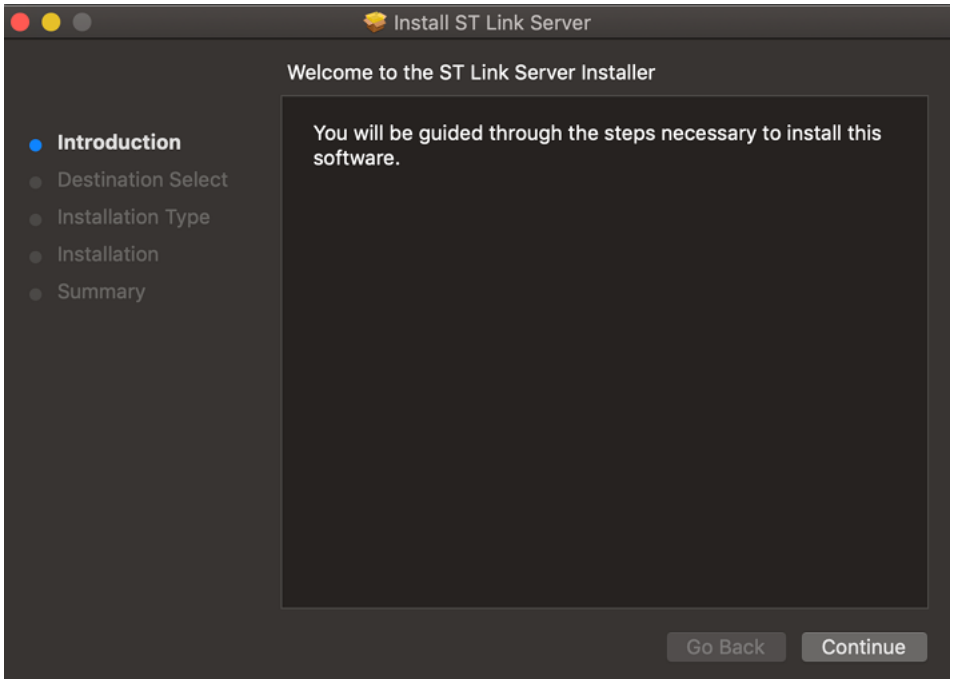
   –`ARCHITECTURE` is the architecture of the target host computer to run STM32CubeIDE. Example: `x86_64 or aarch64`
2. Read the license agreement. Click on [Agree] to accept the terms of the agreement, or [Disagree] to abort the installation. If the agreement is accepted, the installation wizard continues.



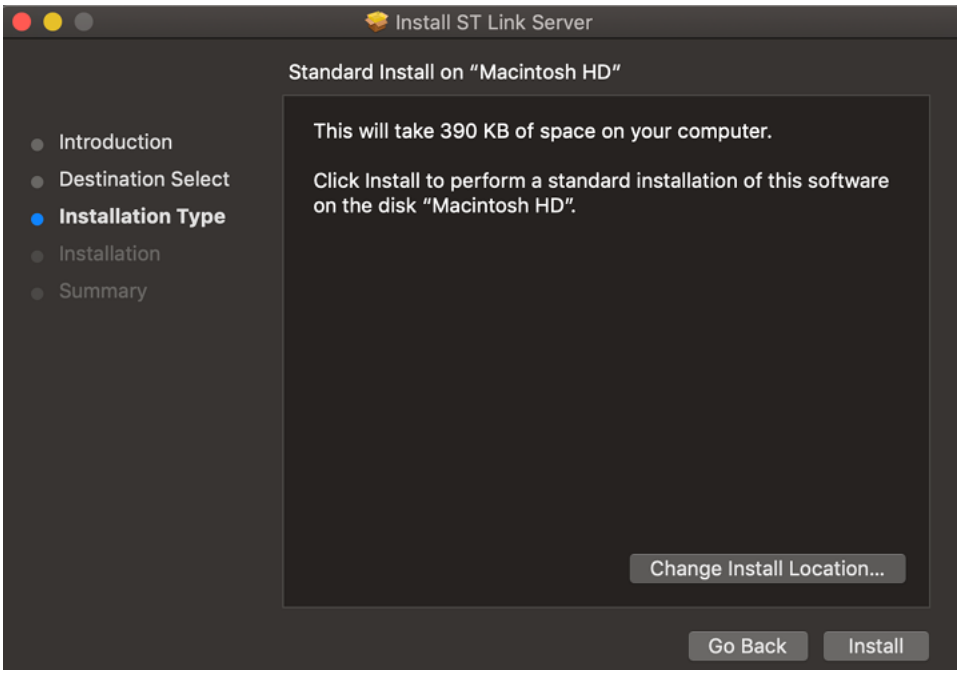
3. Wait for the installation welcome page to appear.
4. Double click on the `.pkg` file indicated with the curved arrow and text stating “Install me 1st”.



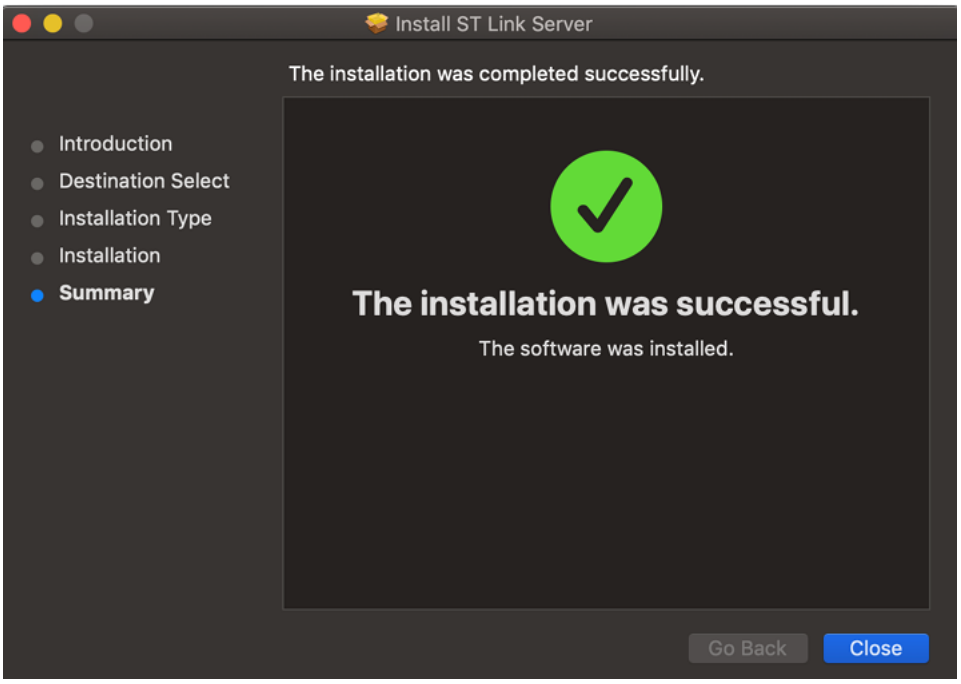
5. This installation is required and installs the ST Link Server. Click on the [Continue] button.



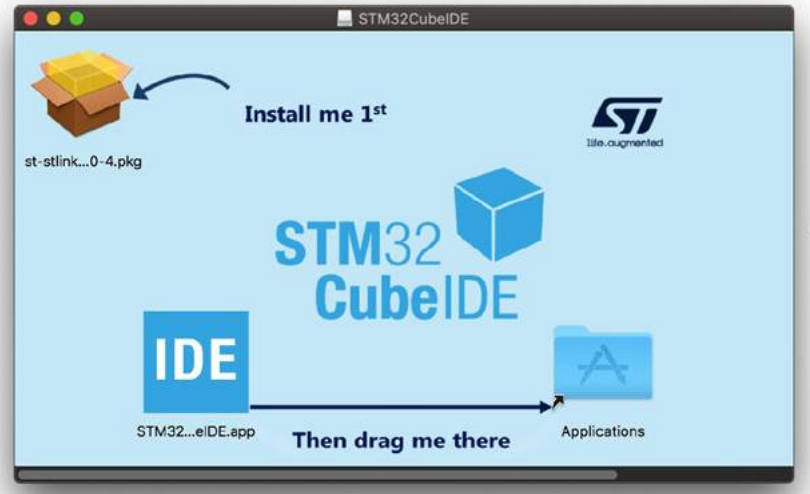
6. In this dialog, select the location for the installation of the ST Link Server.
7. Click on [Install]. Log in to authorize the installation if prompted to do so.
8. If a warning is displayed stating “This package is incompatible with this version of macOS and may fail to install”, click on [Install Anyway].



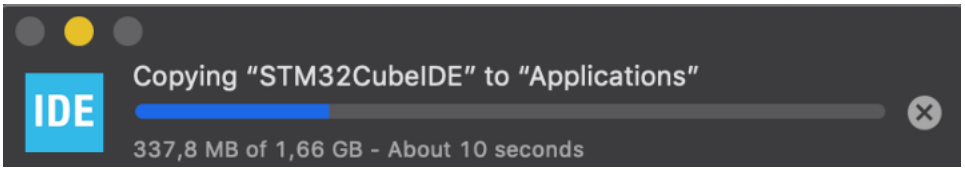
9. Click on [Close] and continue the installation of the STM32CubeIDE product.



10.After successfully installing ST Link Server, drag the STM32CubeIDE icon to the Applications folder as indicated by the straight arrow.



11. Wait for the installation to finish. When done, it is possible to launch STM32CubeIDEfrom the Launchpad by clicking on the IDE icon.



4.6.1 Rosetta® installation on M1-based computers

To install STM32CubeIDE on an M1-based computer running macOS®, the user is prompted to install Rosetta® for the STM32CubeIDE installation to proceed.

After accepting to install Rosetta®, the user credentials must be entered (see Figure 14), after which

Rosetta® installation starts (see Figure 15).



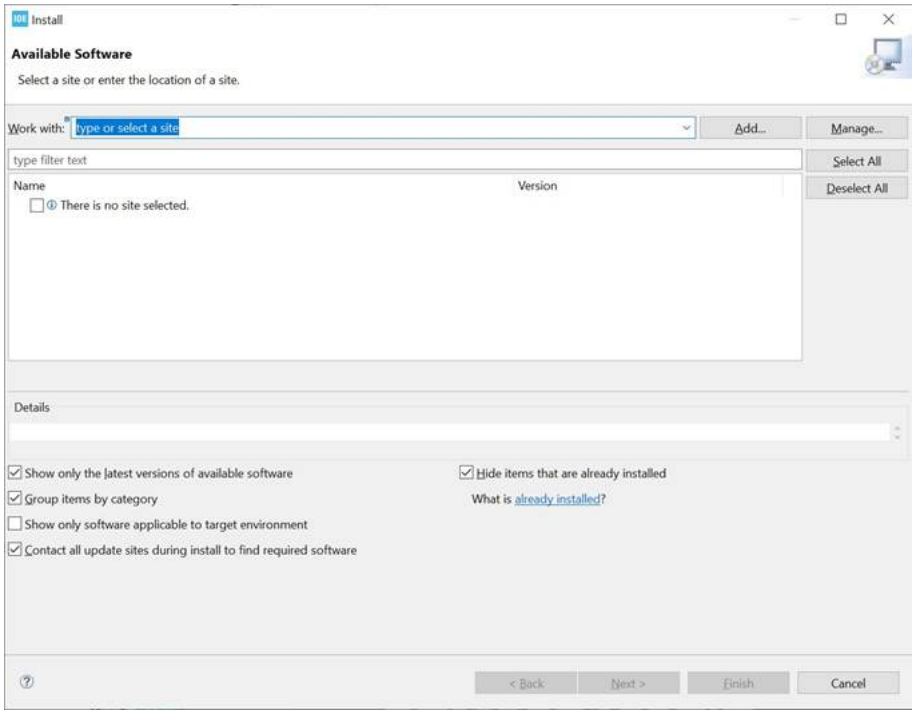
4.7 Update an STM32CubeIDE installation

The Eclipse ® update mechanism permits the quick update of available patches when STM32CubeIDE is already installed. To use this mechanism:

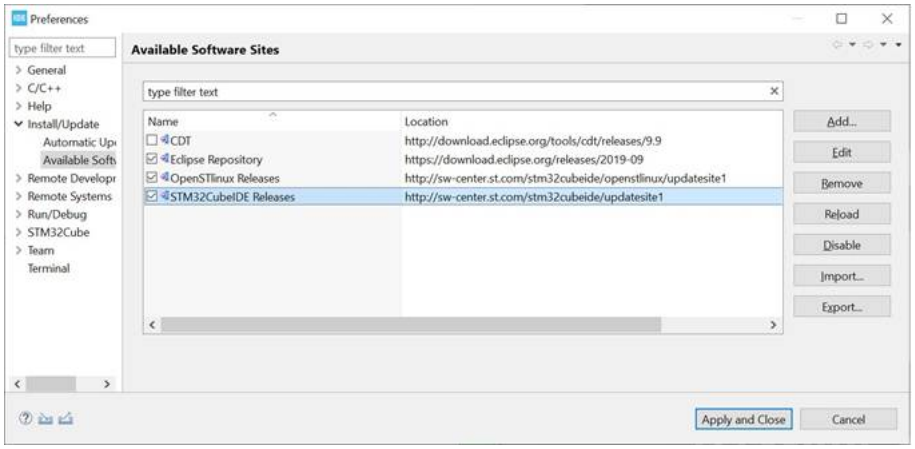
1. Launch STM32CubeIDE
2. Update the tool by [Help]>[Check for updates...]
3. Restart STM32CubeIDE
   – Either automatically if STM32CubeIDE proposes it
   – By exiting STM32CubeIDE and restarting it otherwise

By default, the update is done with the last version. If another version is needed, follow the steps below:

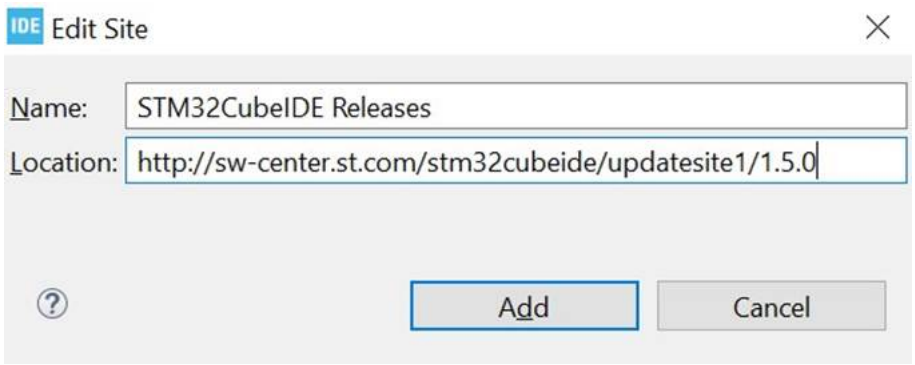
1. Open the Install dialog box as shown in Figure 16 by [Help]>[Install new software...]



2. Click on the [Manage...] button
3. Select [STM32CubeIDE Releases] and click on [Edit] as shown in Figure 17



4. Add the desired version number at the end of the url in the [Location] field as shown in Figure 18



Note: The update mechanism has the same requirements as the installation in term of administration rights. If the installation has been made with administration rights, the update must be made also with the same administration rights.

4.8 Uninstall STM32CubeIDE (Windows®)

To uninstall STM32CubeIDE, under the installation folder:

1. Launch the product uninstaller (`uninstall.exe`)
2. Wait for the uninstaller dialog box and click on [Uninstall]

Note: In Windows ®, the removal of the installation directory instead of using the uninstall procedure prevents further installation. In this case, delete registry key

`HKEY_LOCAL_MACHINE\\SOFTWARE\\WOW6432Node\\STMicroelectronics\\STM32CubeIDE\\VersionToS uppress` using `regedit.exe`.

4.9 Uninstall STM32CubeIDE (Linux®)

The uninstallation of STM32CubeIDE depends on the distribution. Uninstall STM32CubeIDE according to the step below corresponding to the distribution used:

- For any distribution or if the distribution is not known:

  `sudo /opt/st/stm32cubeide_xxxx/uninstall.sh`

- For RPM-based distributions (such as Red Hat®, CentOS™, SUSE®, Fedora® or others):

  `sudo rpm -e st-stm32cubeide_xxxx st-stlink-udev-rules st-stlink-server seggerjlink-udev-rules`

- For Debian-based distributions (such as Debian®, Ubuntu® or others):

`sudo apt-get remove st-stm32cubeide-xxxx st-stlink-udev-rules st-stlink-server segger-jlink-udev-rules`

4.10 Uninstall STM32CubeIDE (macOS®)

To uninstall STM32CubeIDE:

1. Locate the version of STM32CubeIDE to uninstall in the Applications folder in the Finder
2. Drag the STM32CubeIDE app to uninstall to the trash
3. To delete the app permantently, choose [Finder]>[Empty Trash]

5 Software License Agreement (SLA0048)

5.1 Software Package License Agreement

BY CLICKING ON THE “I ACCEPT” BUTTON OR BY UNZIPPING, INSTALLING, COPYING, DOWNLOADING, ACCESSING OR OTHERWISE USING THIS SOFTWARE PACKAGE OR ANY PART THEREOF, INCLUDING ANY RELATED DOCUMENTATION (collectively the “SOFTWARE PACKAGE”) FROM STMICROELECTRONICS INTERNATIONAL N.V, SWISS BRANCH AND/OR ITS AFFILIATED COMPANIES (collectively “STMICROELECTRONICS”), YOU (hereinafter referred also to as “THE RECIPIENT”), ON BEHALF OF YOURSELF, OR ON BEHALF OF ANY ENTITY BY WHICH YOU ARE EMPLOYED AND/OR ENGAGED, AGREE TO BE BOUND BY THIS AGREEMENT.

Under STMICROELECTRONICS’ intellectual property rights and subject to applicable licensing terms for any third-party software incorporated in the SOFTWARE PACKAGE and applicable Open Source Terms (as defined here below), the redistribution, reproduction and use in source and binary forms of the SOFTWARE PACKAGE or any part thereof, with or without modification, are permitted provided that the following conditions are met:

1. Redistribution of source code (modified or not) must retain any copyright notice accompanying the SOFTWARE PACKAGE, this list of conditions and the disclaimer below.
2. Redistributions in binary form, except as embedded into a processing unit device manufactured by or for STMicroelectronics or a software update for any such device, must reproduce the accompanying copyright notice, this list of conditions and the below disclaimer in capital type, in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of other contributors to the SOFTWARE PACKAGE may be used to endorse or promote products derived from the SOFTWARE PACKAGE or part thereof without specific written permission.
4. The SOFTWARE PACKAGE or any part thereof, including modifications and/or derivative works of the SOFTWARE PACKAGE, must be used and execute solely and exclusively on or in combination with a processing unit device manufactured by or for STMicroelectronics.
5. No use, reproduction or redistribution of the SOFTWARE PACKAGE partially or totally may be done in any manner that would subject the SOFTWARE PACKAGE to any Open Source Terms (as defined below).
6. Some portion of the SOFTWARE PACKAGE may contain software subject to Open Source Terms (as defined below) applicable for each such portion (“Open Source Software”), as further specified in the SOFTWARE PACKAGE. Such Open Source Software is supplied under the applicable Open Source Terms and is not subject to the terms and conditions of this AGREEMENT. “Open Source Terms” shall mean any open source license which requires as part of distribution of software that the source code of such software is distributed therewith or otherwise made available, or open source license that substantially complies with the Open Source definition specified at www.opensource.org and any other comparable open source license such as for example GNU General Public License (GPL), Eclipse Public License (EPL), Apache Software License, BSD license and MIT license.
7. The SOFTWARE PACKAGE may also include third-party software as expressly specified in the SOFTWARE PACKAGE subject to specific license terms from such third parties. Such third-party software is supplied under such specific license terms and is not subject to the terms and conditions of this AGREEMENT. By installing, copying, downloading, accessing or otherwise using the SOFTWARE PACKAGE, the RECIPIENT agrees to be bound by such license terms with regard to such third-party software.
8. STMicroelectronics has no obligation to provide any maintenance, support or updates for the SOFTWARE PACKAGE.

9. The SOFTWARE PACKAGE is and will remain the exclusive property of STMicroelectronics and its licensors. The RECIPIENT will not take any action that jeopardizes STMicroelectronics and its licensors' proprietary rights or acquire any rights in the SOFTWARE PACKAGE, except the limited rights specified hereunder.
10. The RECIPIENT shall comply with all applicable laws and regulations affecting the use of the SOFTWARE PACKAGE or any part thereof including any applicable export control law or regulation.
11. Redistribution and use of the SOFTWARE PACKAGE partially or any part thereof other than as permitted under this AGREEMENT is void and will automatically terminate RECIPIENT's rights under this AGREEMENT.
12. The RECIPIENT shall be solely liable to determine and verify that the SOFTWARE PACKAGE is fit for the RECIPIENT intended use, environment or application and comply with all regulatory, safety and security related requirements concerning any use.

5.2 DISCLAIMER

THE SOFTWARE PACKAGE IS PROVIDED BY STMICROELECTRONICS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS, IMPLIED OR STATUTORY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS ARE DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT SHALL STMICROELECTRONICS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE SOFTWARE PACKAGE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

EXCEPT AS EXPRESSLY PERMITTED HEREUNDER AND SUBJECT TO THE APPLICABLE LICENSING TERMS FOR ANY THIRD-PARTY SOFTWARE INCORPORATED IN THE SOFTWARE PACKAGE AND OPEN SOURCE TERMS AS APPLICABLE, NO LICENSE OR OTHER RIGHTS, WHETHER EXPRESS OR IMPLIED, ARE GRANTED UNDER ANY PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS OF STMICROELECTRONICS OR ANY THIRD PARTY.

If you want, I can also provide a "strict legal-preservation" variant that keeps every original character and only fixes layout.

6 FAQ

6.1 Authentication issue

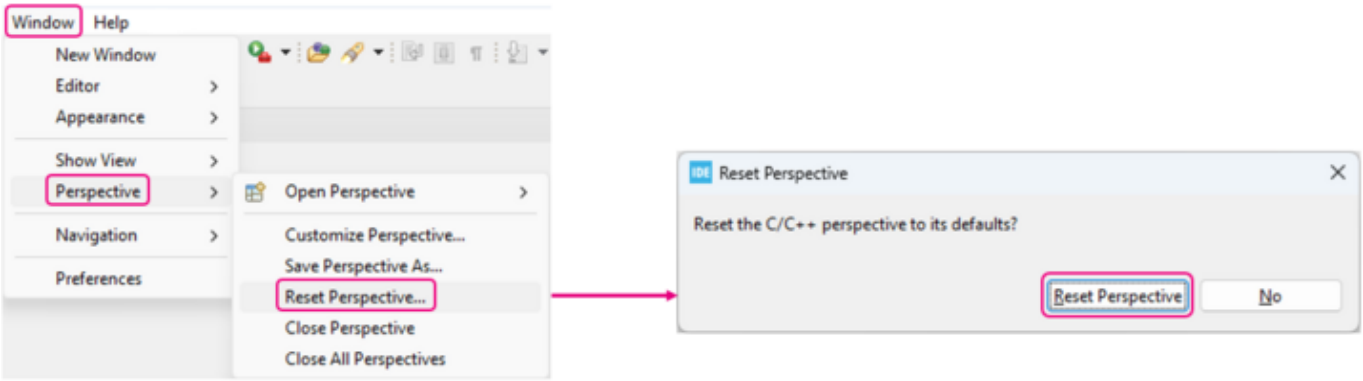
For STM32CubeIDE versions prior to 2.0.0: If you have encountered an authentication problem while using STM32CubeIDE, do not worry. You simply need to set your connection parameters. For more details, see this link : https://community.st.com/t5/stm32-mcus/how-to-set-the-network-connection-parameters-for-stm32cubeide/ta-p/585694

6.2 How to refresh the code after an update within STM32CubeMX standalone tool

If the new version of STM32CubeIDE 2.0.0 is used, imported projects from STM32CubeMX need to be refreshed according to the following instructions: right-click on the imported project, and select the :guilabel:Refresh entry in the menu. Perspectives ————————

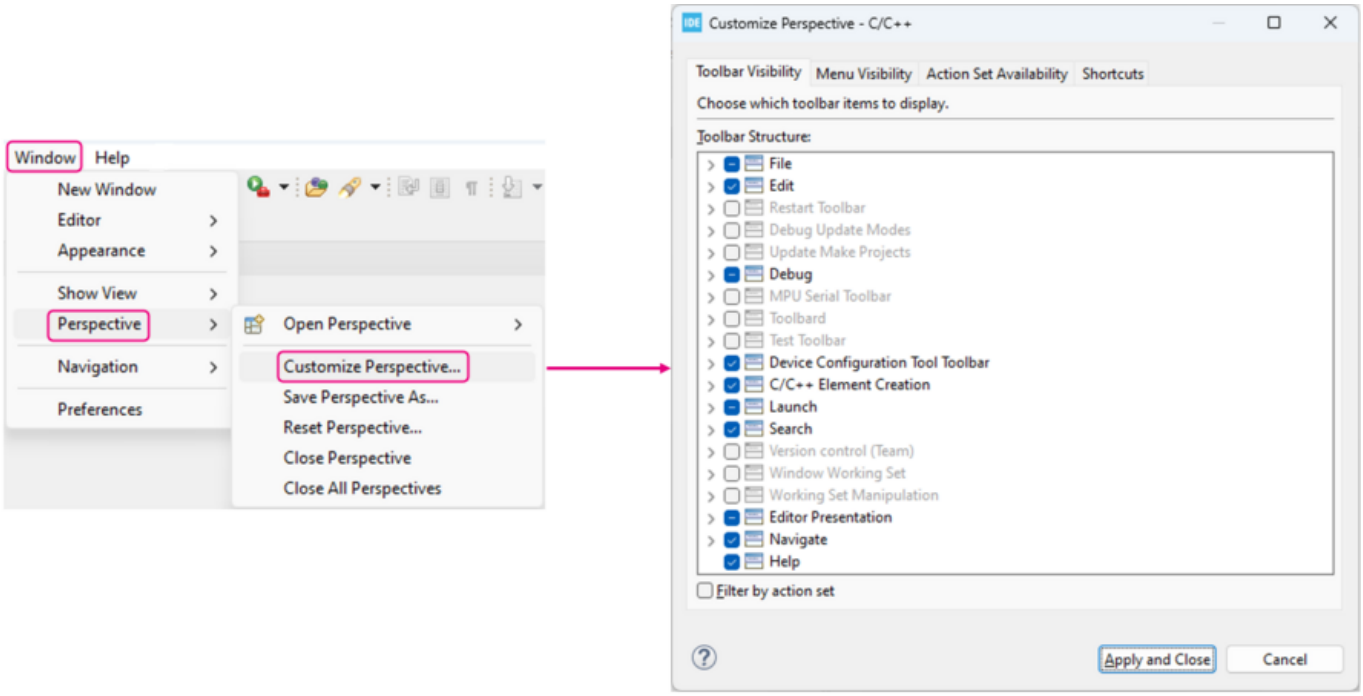
6.2.1 Reset perspectives

The user can reset the perspective back to the default configuration using the menu *Window* *Perspective* *Reset Perspective...*



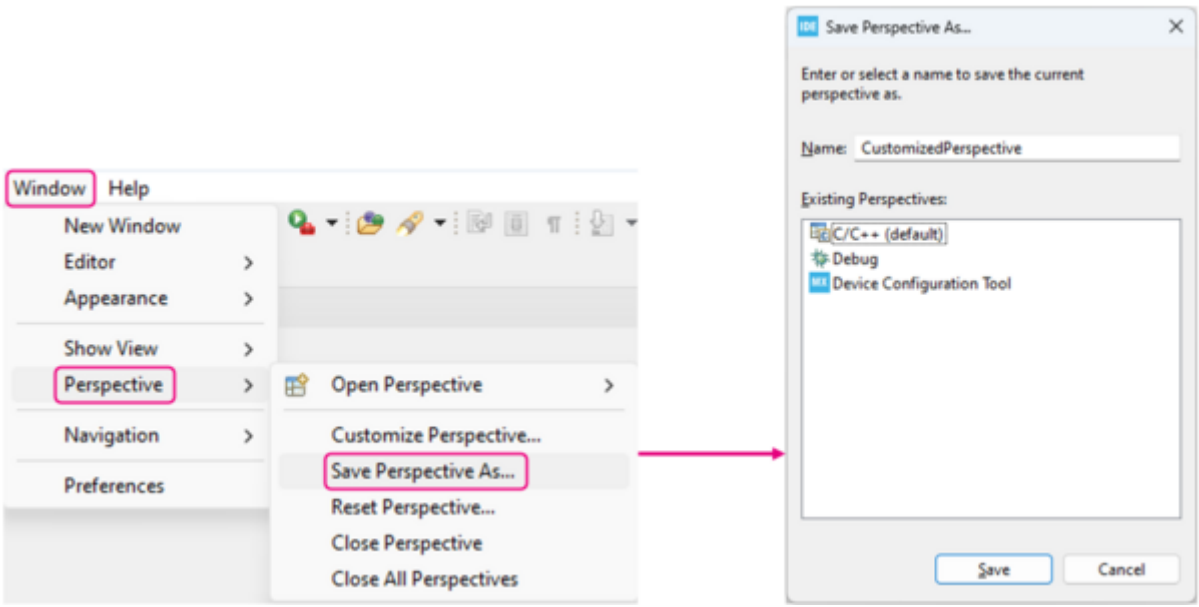
6.2.2 Customize perspectives

The user can customize the perspectives using the menu :menuselection:Window –> Perspective –> Customize Perspective... and choosing toolbars and menu visibility, action sets, and shortcuts availability.



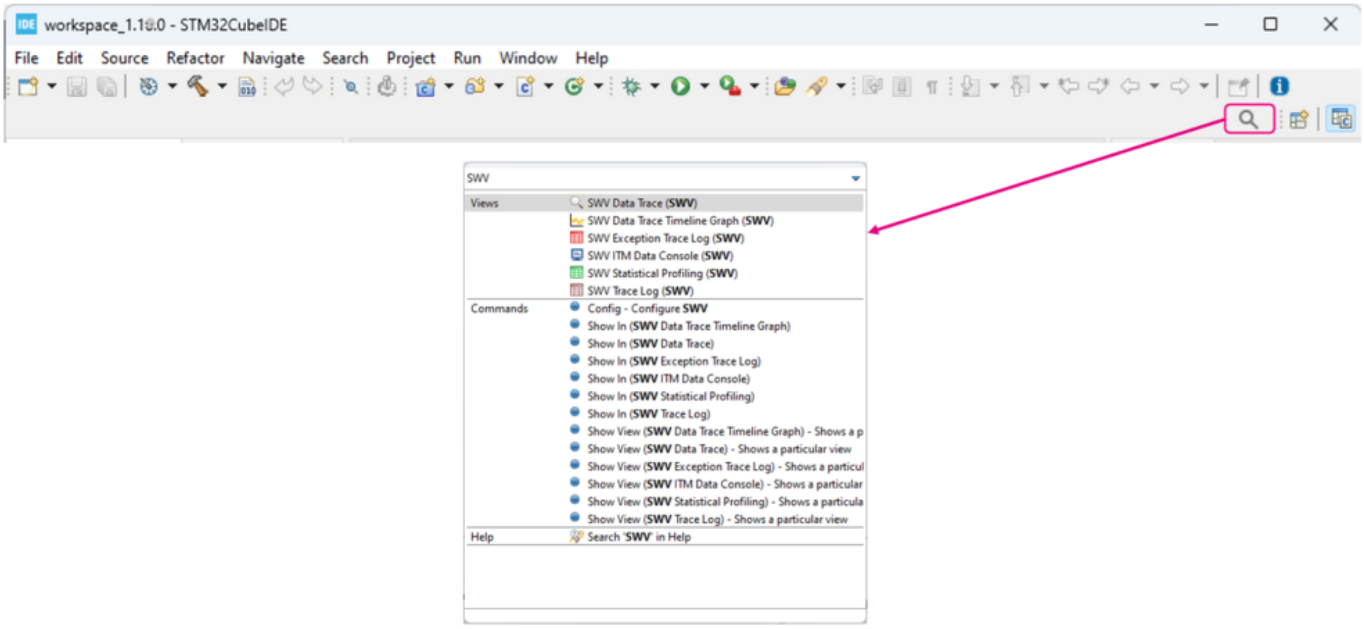
6.2.3 Create an own perspective

The user can create a customized perspective after selecting or moving some views and using the menu *Window* *Perspective* *Save Perspective As...*, and then choosing a name to save it.



6.3 Quick access to views, menus, preferences

A quick-access menu is available to search for views, menus, and preferences by clicking on the magnifying glass at the top right and typing the name of the view in the dialog.



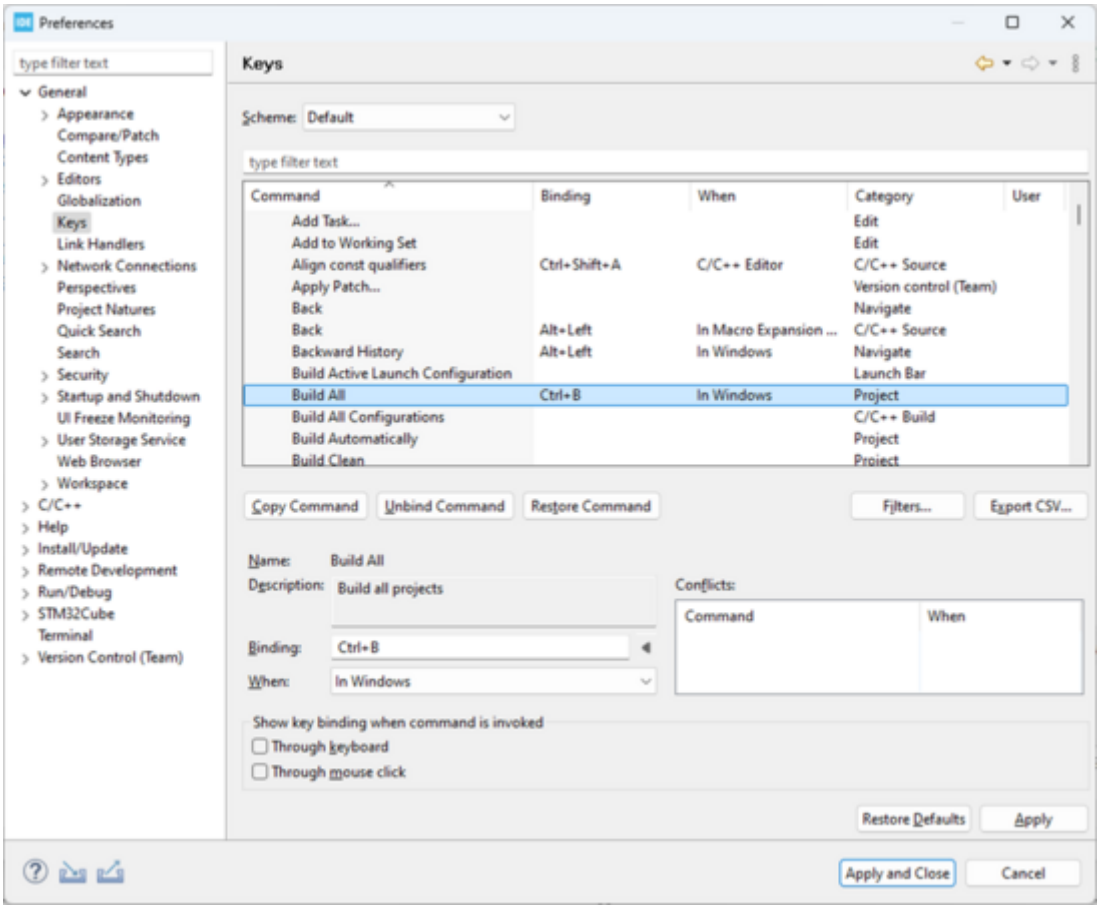
6.4 Key bindings

Select the menu *Window* *Preferences* then select the category *General* *Keys*.

Choose between three different key binding schemes: Eclipse, GNU Emacs, Microsoft Visual Studio.

Use the search filter to find a command and learn or configure its specific key binding.

For example, use `Ctrl+B` to Build All:



6.5 Shortcuts

6.5.1 Shortcuts to debug

Use shortcuts to run, debug, and control code execution.

- `Ctrl+F11` to run the last application launched.
- `F11` to debug the last application launched.
- `F5` to step into.
- `F6` to step over, i.e. go to next line.
- `F7` to step return to caller.
- `F8` to resume until next breakpoint.

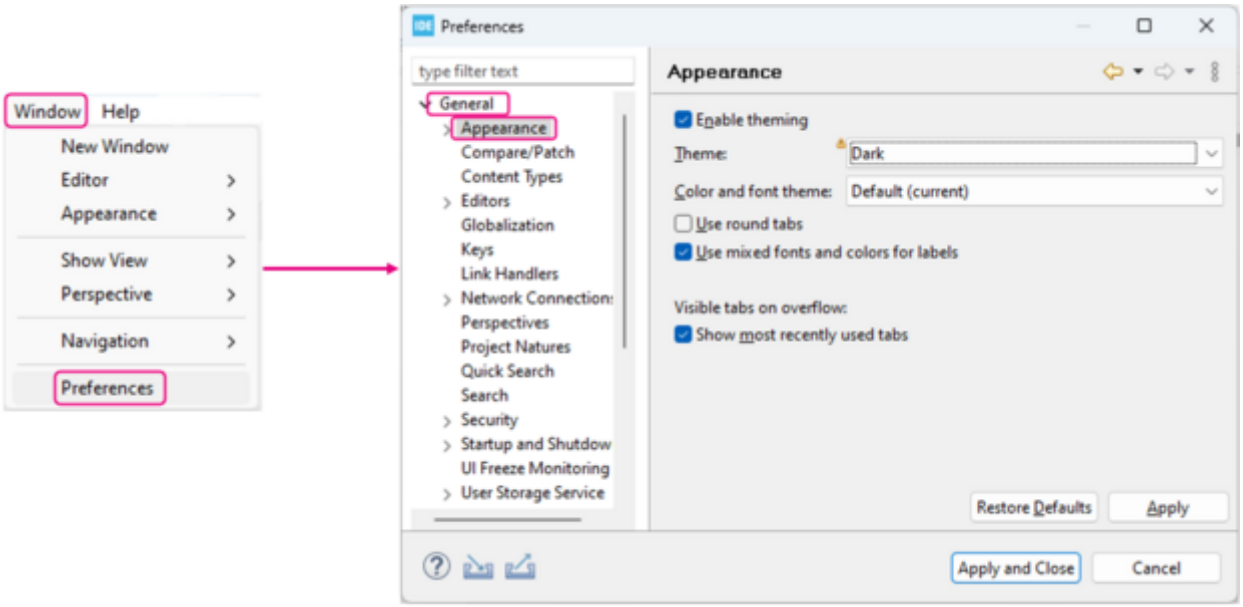
6.5.2 Shortcuts to code

Use shortcuts to code.

- `Ctrl+Space` to auto complete.
- `Ctrl+H` to search workspace resources with useful filters.
- `Alt+Shift+A` to select blocks. It is useful to modify several lines at a time.
- `Ctrl+O` to open the quick outline.
- `Ctrl+Shift+R` to open resources.
- `Ctrl+Shift+T` to open elements.

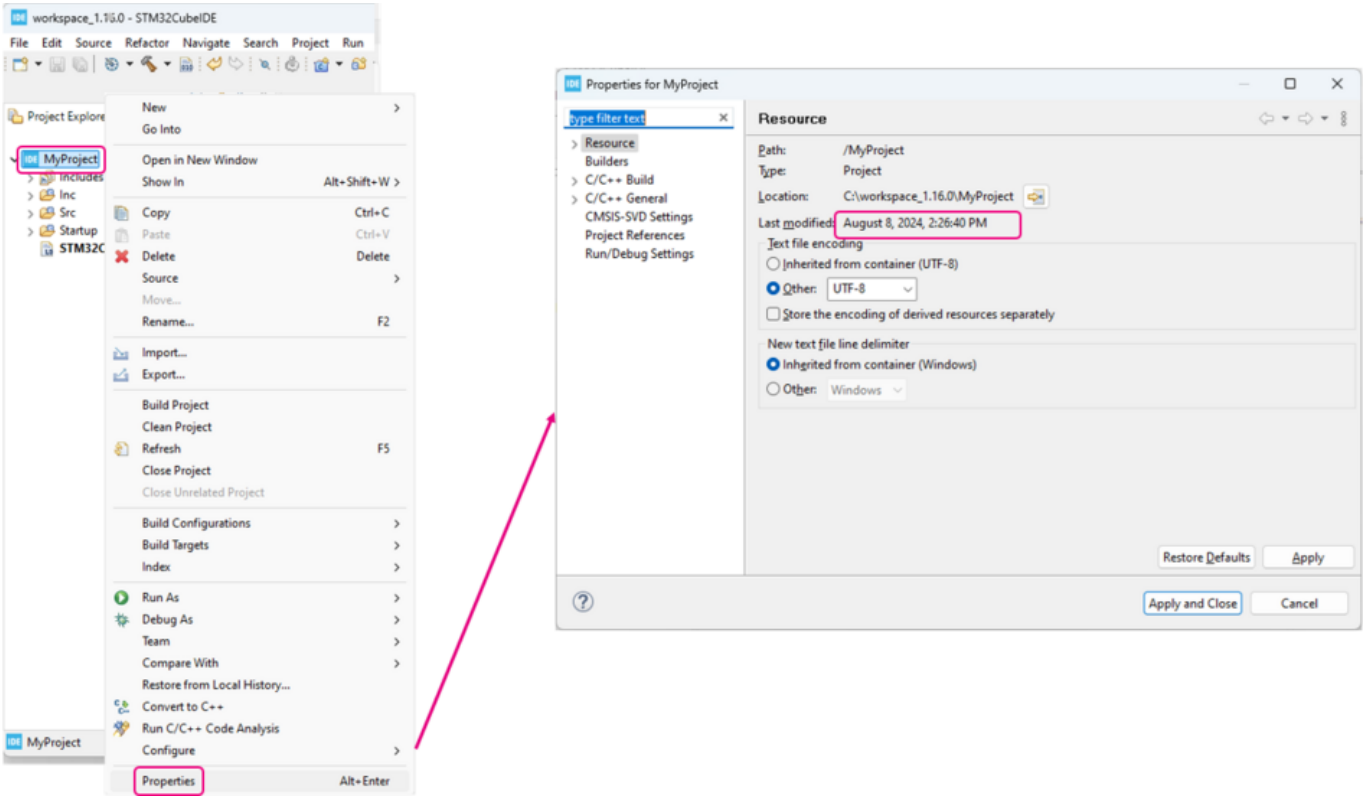
6.6 Dark theme

The dark theme is a very popular preference. A restart is needed for the theme change to come into effect. Select the menu *Window* *Preferences* then select the category *General* *Appearance* and turn the theme to dark.



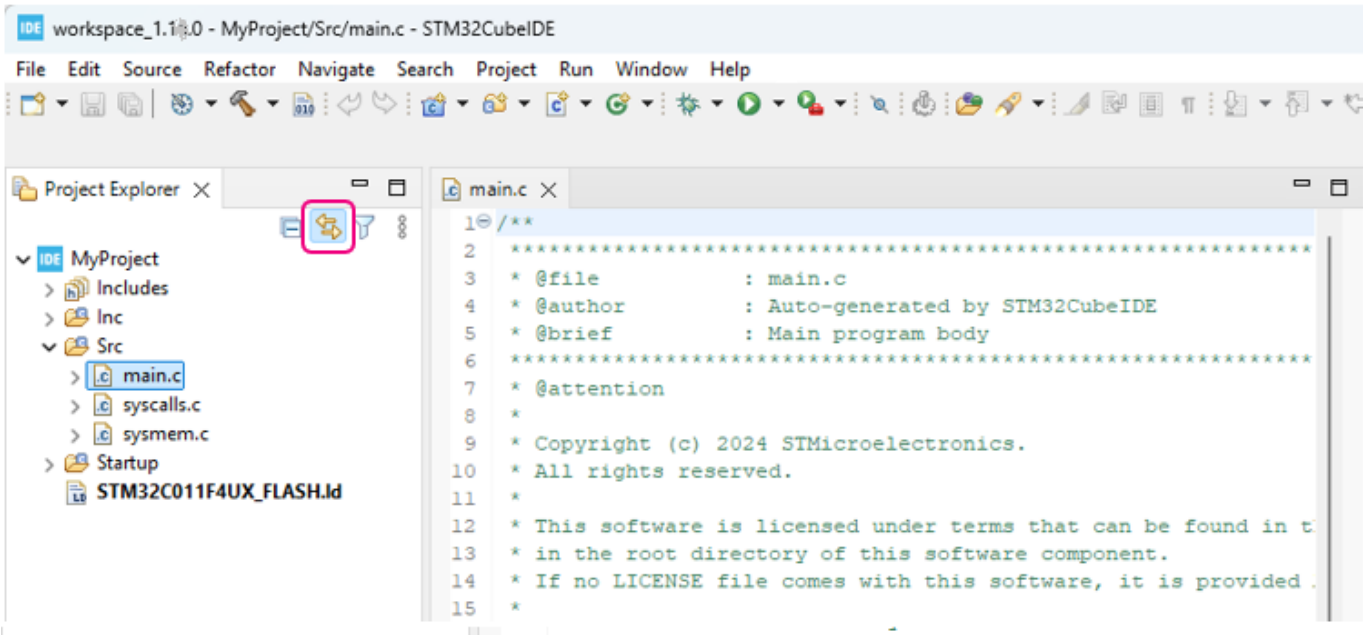
6.7 Workspace path

STM32CubeIDE is always launched inside a workspace. A workspace contains projects. It is easy to open a file explorer inside the workspace folder. Select a project inside the Project Explorer, right-click to open the menu and then select *Properties*.



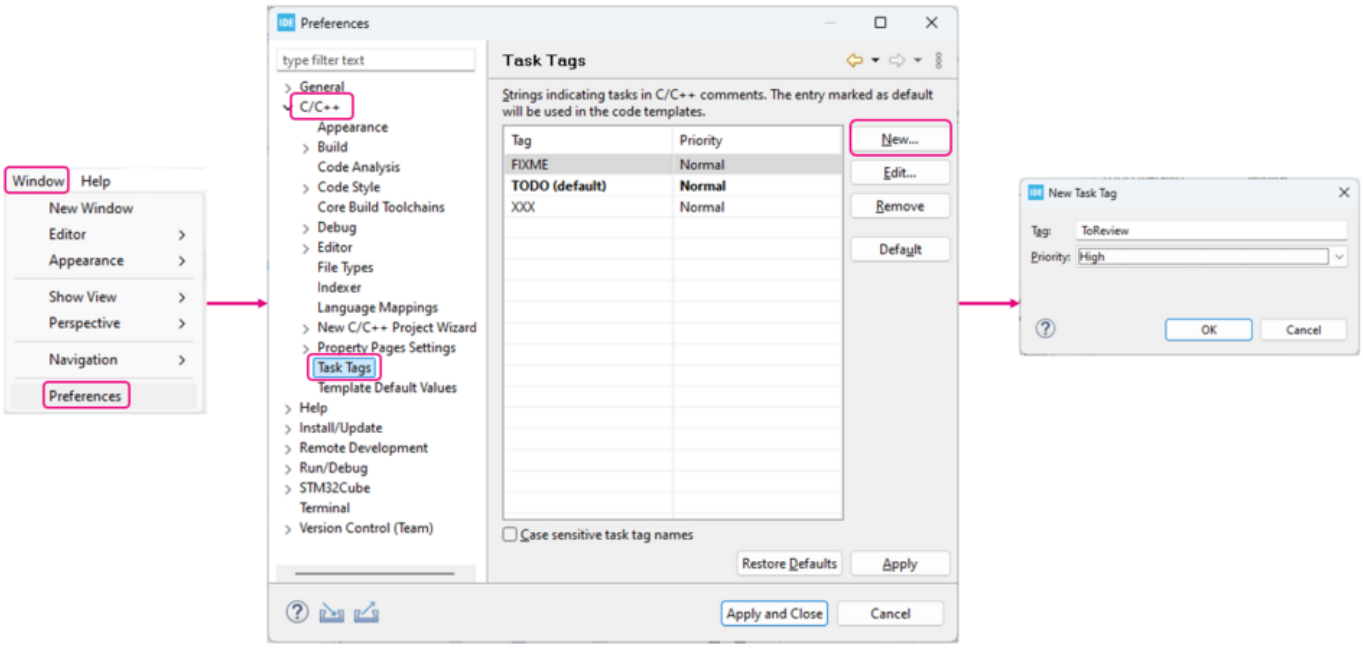
6.8 Link with editor

To know where a file is located, activate the link with editor feature. By default, this feature is toggled off. The link with editor feature is very convenient to find a file inside a large folder hierarchy. It is recommended to toggle it on only when needed.

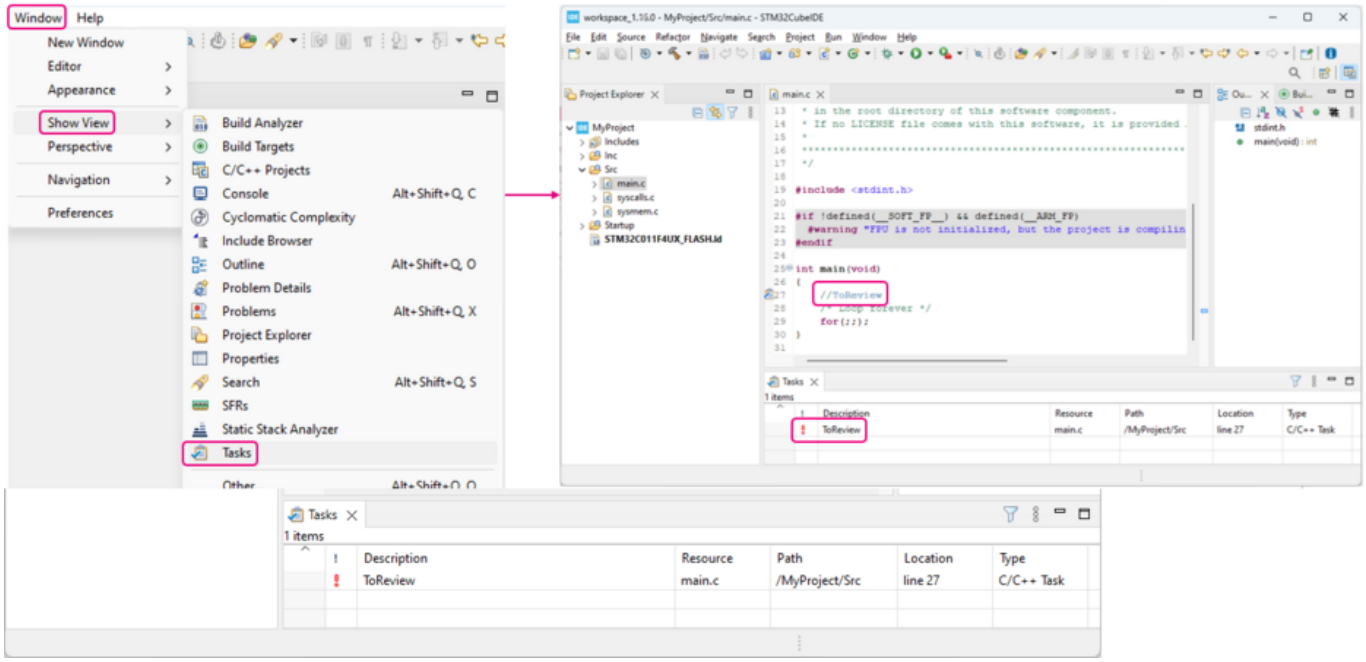


6.9 Task tags

Define your own task tags to keep track of your selections. Select the menu *Window* *Preferences* then select the category *C/C++* *Task Tags* and click on the *New* button. Enter a tag name and a priority (High/Normal/Low). Tags may contain any character string.

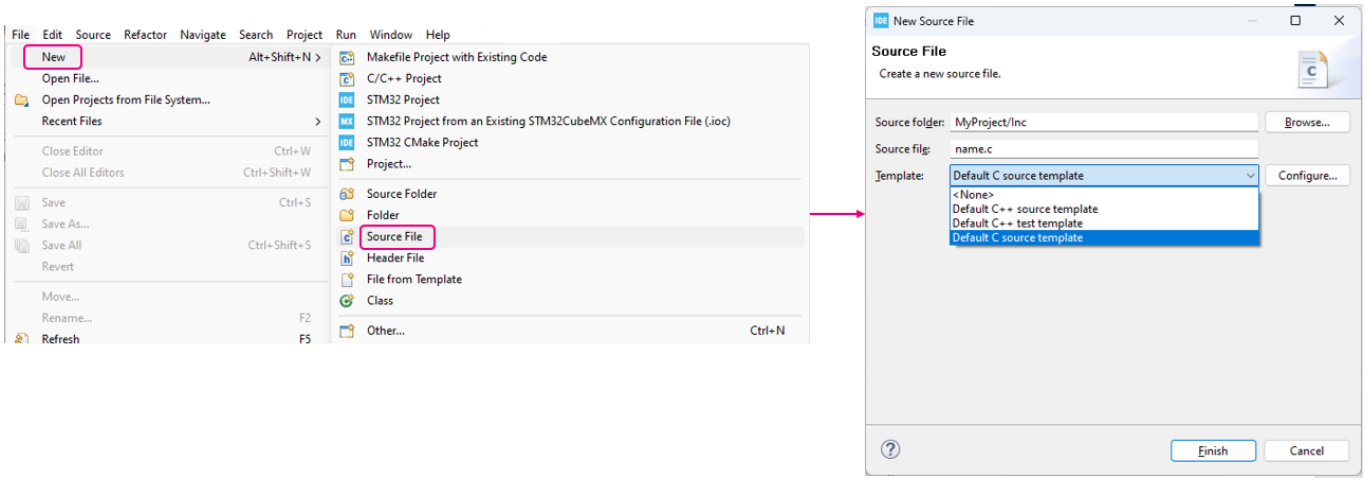


Add several tags into the code. Open the *Tasks* view: *Window* *Show views* *Tasks* to access and navigate your list of tags.



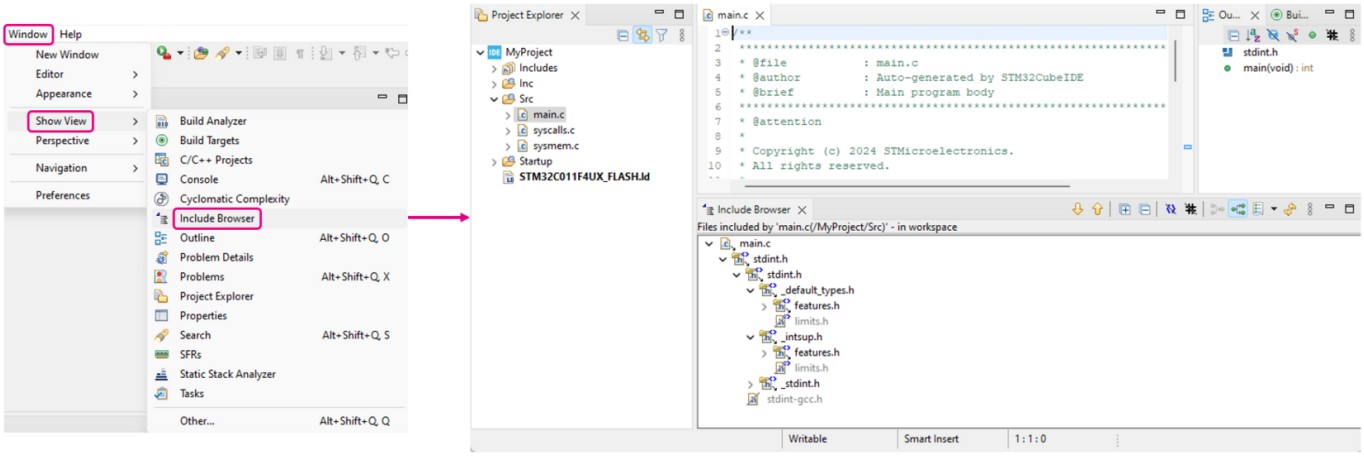
6.10 Code templates

To simplify and standardize the source files, usage of code templates is very useful. Select the menu *File* *New* *Source File* and select the template, or configure a new default template, or modify an existing one by clicking *Configure...*.



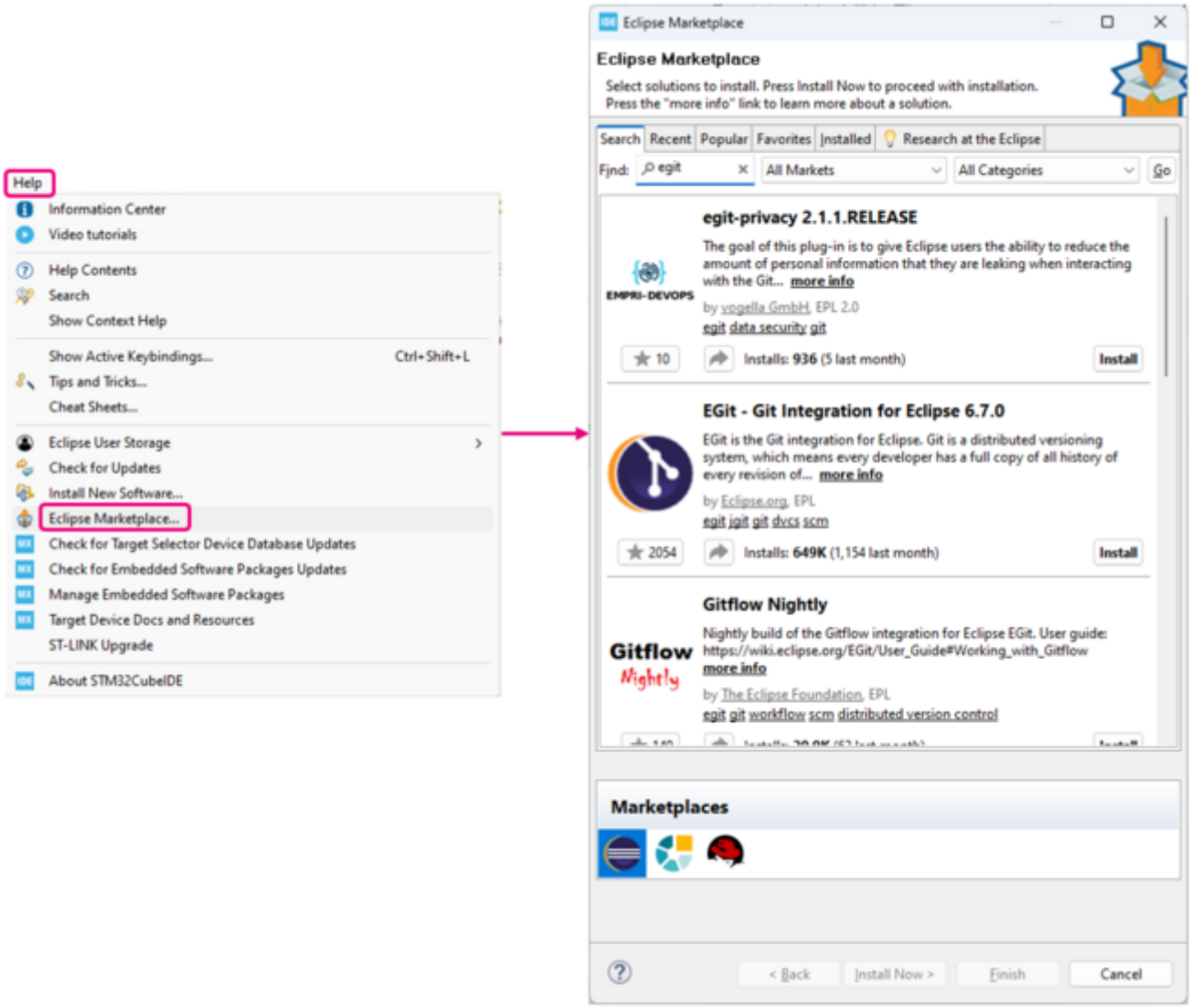
6.11 Include browser

Open the *Include Browser* view with the menu *Window* *Show View* *Include Browser* to visualize include dependencies between files. Drag and drop a file into the view.



6.12 Eclipse Marketplace

The Eclipse Marketplace is an interface for browsing and installing additional tools or features. Open the menu *Help* *Eclipse Marketplace*. Search for a keyword, for example `egit`, for related solutions.



6.13 Cyclic redundancy check - An error detection method

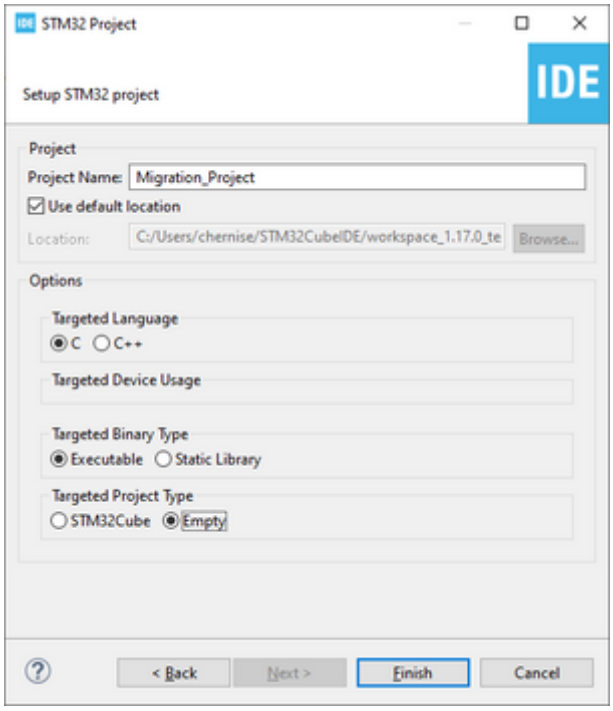
Cyclic redundancy check (CRC) is an error detection method used for digital data, based on binary division. The CRC algorithm generates a fixed-length checksum code.

The post below describes how to implement the CRC mechanism in your project: CRC computation in STM32 MCUs and post-build creation

The following GitHub repository contains an example of how to generate the CRC for your projects: ethanhuan-ginst/STM32CubeIDE-Workshop-2019 GitHub

6.14 Project migration from IAR Embedded Workbench® for Arm or Keil® MDK-ARM to STM32CubeIDE

Create an empty project in STM32CubeIDE based on the same MCU as the EWARM or MDK-ARM project.

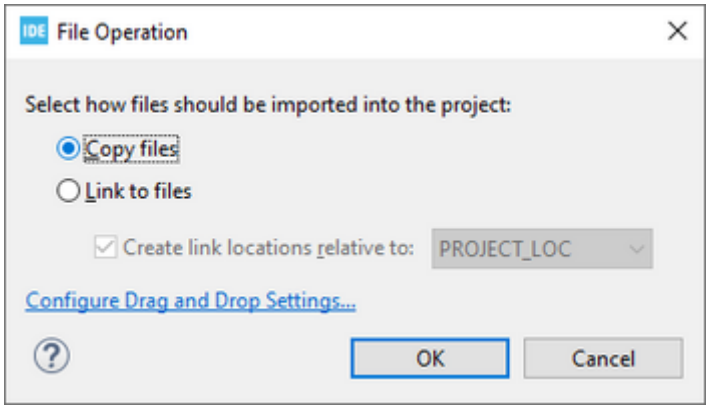


The empty project contains the basic necessary items to make the MCU work:

- Startup file
- Linker file

Manually import source files from the original project by dragging and dropping them into the STM32CubeIDE workspace. During this operation, a pop-up message appears. Select the adequate option for your application:

- `Copy files`: This creates a copy of the file under the project folder.
- `Link to files`: This creates a link between the STM32CubeIDE project in the workspace and the original project folder.

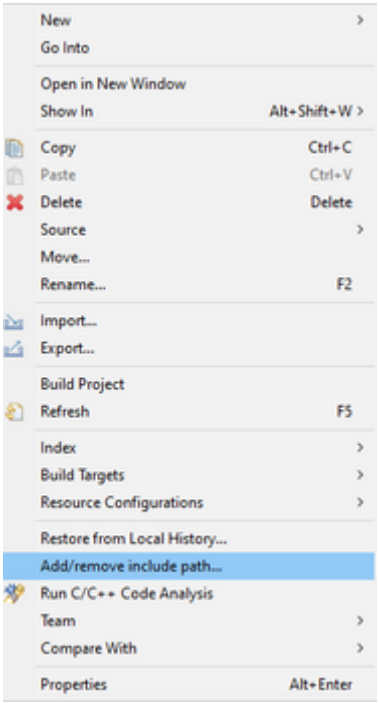


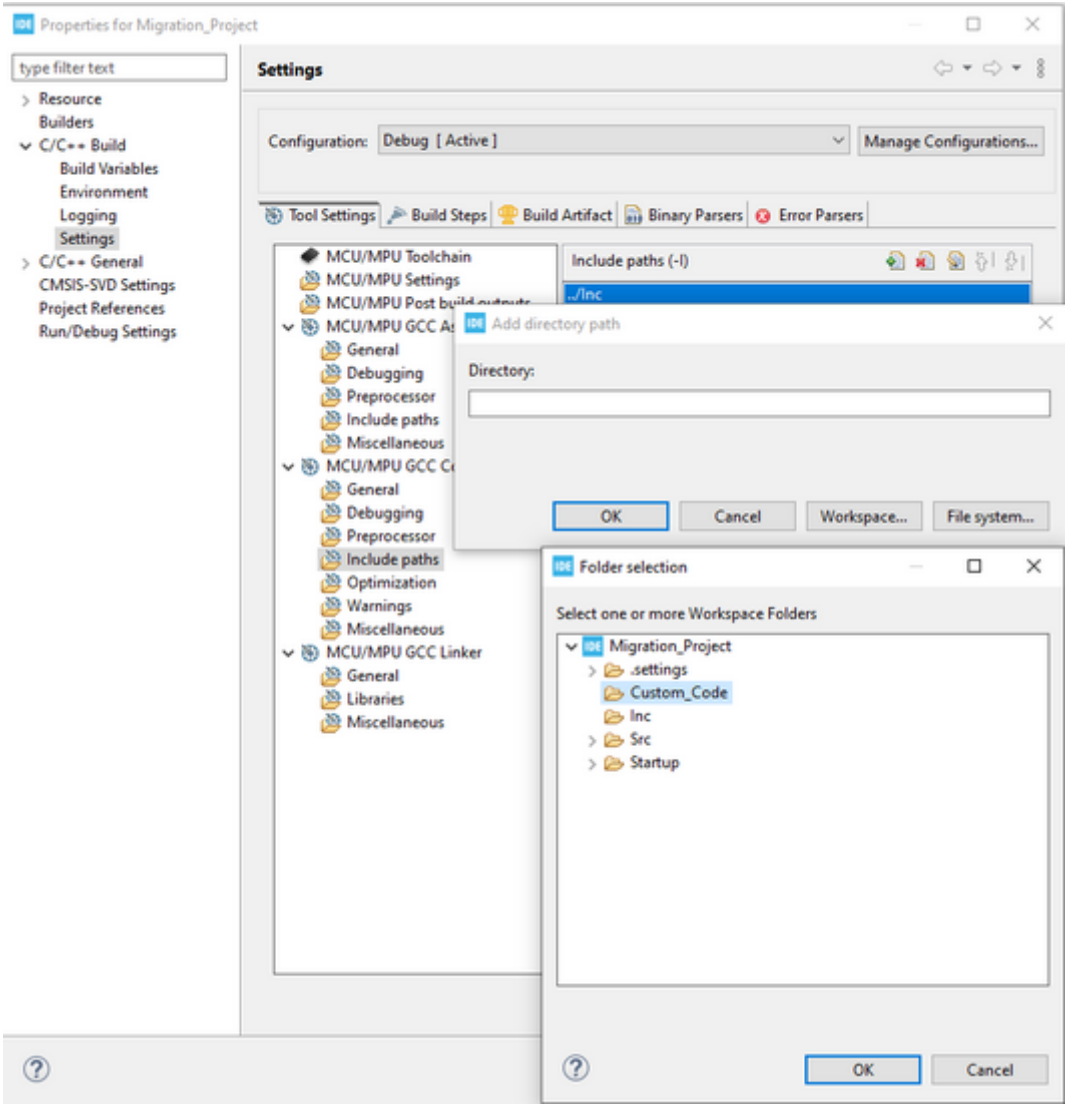
Set the include paths from STM32CubeIDE settings:

- Option 1: Right-click on the folder inside the workspace and select *Add/remove include path...*
- Option 2: Under *Properties C/C++ Build Settings MCU/MPU GCC Compiler Include paths*, add the include files folder.

The remaining options (such as compilation and linker) are described in the STM32CubeIDE user guide.

At the end, the user obtains an STM32CubeIDE project.


New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Refresh F5
Index
Build Targets
Resource Configurations
Restore from Local History...
Add/remove include path...
Run C/C++ Code Analysis
Team
Compare With
Properties Alt+Enter


Properties for Migration_Project
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
C/C++ General
CMSIS-SVD Settings
Project References
Run/Debug Settings
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
MCU/MPU Toolchain
MCU/MPU Settings
Include paths (-I)
../Inc
General
Debugging
Preprocessor
Include paths
Miscellaneous
Optimization
Warnings
MCU/MPU GCC Linker
Libraries
Add directory path
Directory:
OK
Cancel
Workspace...
File system...
Folder selection
Select one or more Workspace Folders
Migration_Project
.settings
Custom_Code
Inc
Src
Startup
OK
Cancel

6.15 The link time optimization (LTO)

The link time optimization (LTO) is a feature that requests the linker to optimize generated code. To apply LTO, add the `-flto` option to compiler (CFLAGS) and linker (LDFLAGS) options in project properties.

Add the flag `-flto` to the MCU GCC Compiler Miscellaneous:

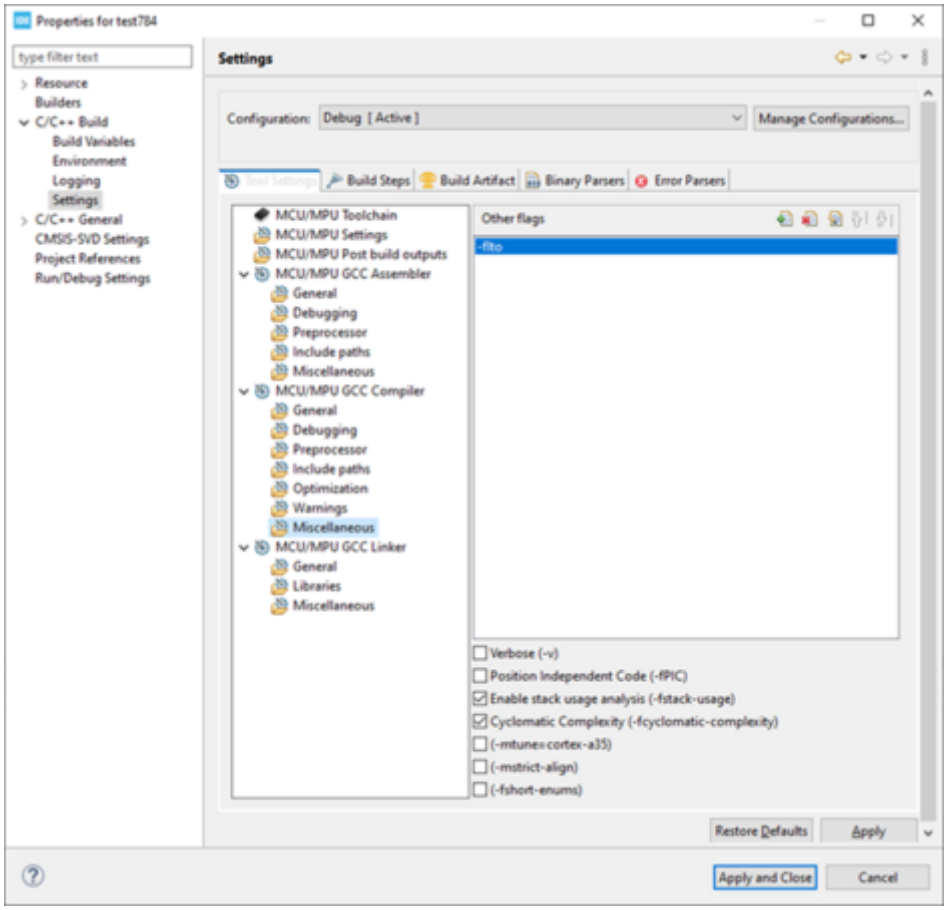
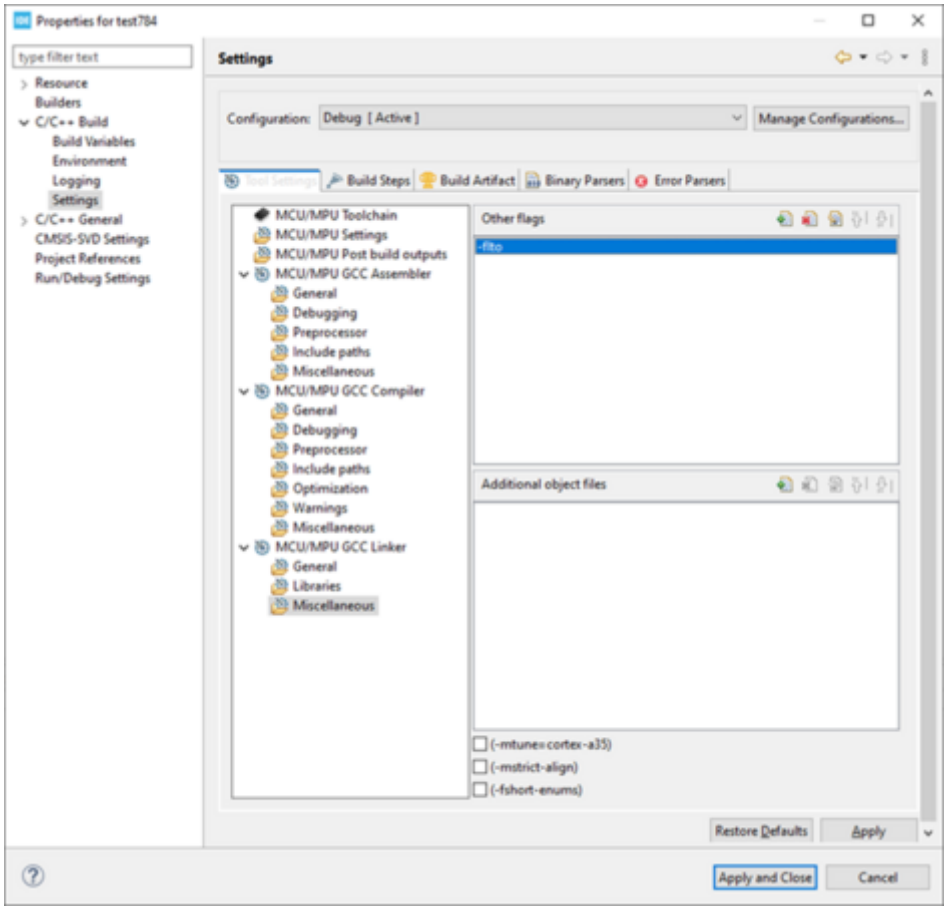


Fig. 1: LTO compiler flag

Add the flag `-flto` to the MCU GCC linker Miscellaneous:



6.16 Memory management

If the load address is wrong in ELF file.

If the user changes the origin of a memory area with a non-aligned address in the linker file and enables conversion to Hex (`-O hex`), for example:

```
From
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 24K
  FLASH  (rx)     : ORIGIN = 0x8000000,    LENGTH = 128K
}
to
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 24K
  FLASH  (rx)     : ORIGIN = 0x8000800,    LENGTH = 128K
}
```

The user has to change the maximum page size by adding the flag `-n` which turns off page alignment (`-Wl,-n` on the GCC driver command line).

GNU `ld` manual full description:

- `-n`
- `--nmagic` Turn off page alignment of sections, and disable linking against shared libraries. If the output format

supports Unix style magic numbers, mark the output as NMAGIC.

6.17 Debug an already-running target

STM32CubeIDE does not provide a dedicated one-click command to attach to a running target. Configure a debug session to load the application symbols without reloading the program or resetting the target.

Before you start:

- Start the target and ensure the application is running.
- Use the ELF file that matches the firmware running on the target. Matching symbols are needed for accurate source-level debugging.

Configure the debug session:

1. In Project Explorer, right-click the project and select *Debug As* *Debug Configurations…*.
2. Select the debug configuration for your STM32 project, or create one.
3. Configure the application loading option to **Load symbols only**.
4. In the **Debugger** tab, set **Reset behavior** to **None**.
5. In the **Startup** tab, select the loaded file, clear **Download**, and keep **Load symbols** selected.
6. Apply the changes and start the debug session.

The debugger connects using the selected symbols without reprogramming or resetting the target. To return to the usual debug workflow, restore the normal application-loading and reset settings in the configuration.

7 Community

7.1 Support link

If you need assistance or have questions about STM32CubeIDE, support is available. Access a wide range of resources and get help from the STMicroelectronics support team. For more information and support, visit the STMicroelectronics Support Home.

7.2 Community link

Your feedback is valuable and helps improve STM32CubeIDE. If you have suggestions, issues, or comments, share them with the community. To provide feedback and join discussions, visit the STM32CubeIDE Community.

7.3 Knowledge articles link

For information on STM32 microcontroller programming, troubleshooting, ecosystem tools, and product details, visit the STM32 MCUs Knowledge Base.

8 Previous documents