STM32CubeIDE

Stable: 23/11/2021 - 10:29 | Revision: 30/11/2021 - 10:29

A quality version of this page, approved on *23 November 2021*, was based off this revision.

This article explains some of the basics of STM32CubeIDE support for STM32 MPU. STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem.

For more information about STM32CubeIDE, refer to its user guide.

Contents

1 STM32CubeIDE purpose

STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors. It is based on the Eclipse®/CDT framework, GCC toolchain for the development and GDB for the debugging.

It allows the integration of the hundreds of existing plugins that complete the features of the Eclipse® IDE. STM32CubeIDE integrates all STM32CubeMX functionalities to offer all-in-one tool experience, and save installation and development time.

With STM32CubeIDE, you can

- select the appropriate STM32 device corresponding to your needs
- configure the device using STM32CubeMX
- develop and debug applications on top of Arm® Cortex®-M

2 How to install STM32CubeIDE

STM32 MPU support inside STM32CubeIDE is available on Linux® and Windows® host PCs, but **it is NOT on macOS®.**

	STM32CubeIDE for Linux® host PC	STM32CubeIDE for Windows® host PC

Download	**Version 1.6.1**
	• Download the preferred all-in-one Linux installer from my.st.com
	  • *Generic Linux Installer - STM32CubeIDE-Lnx*
	  • *RPM Linux Installer - STM32CubeIDE-RPM*
	  • *Debian Linux Installer - STM32CubeIDE-DEB*
		Version 1.6.1
		• Download the all-in-one Windows installer from my.st.com
		  • *Windows Installer - STM32CubeIDE-Win*

Installation guide	• Refer to *STM32CubeIDE installation guide (UM2563)* available on my.st.com.	

User manual	• When the installation is completed, see additional information about STM32CubeIDE in my.st.com:
	  • *STM32CubeIDE quick start guide (UM2553)*
	  • *Getting started with projects based on the STM32MP1 Series in STM32CubeIDE (AN5360)*	

Detailed release note	• Details about the content of this tool version are available in the *STM32CubeIDE release v1.6.1* release note from my.st.com	

Minor releases may be available from the update site. Check chapter 10 in (UM2609) for more information on how to update STM32CubeIDE.

3 How to get started with STM32CubeIDE

This section links to two different *how to* articles depending on whether you are migrating from SW4STM32 to STM32CubeIDE or starting a new project with STM32CubeIDE.

3.1 How to get started with STM32CubeIDE from scratch

How to get started with STM32CubeIDE from scratch.

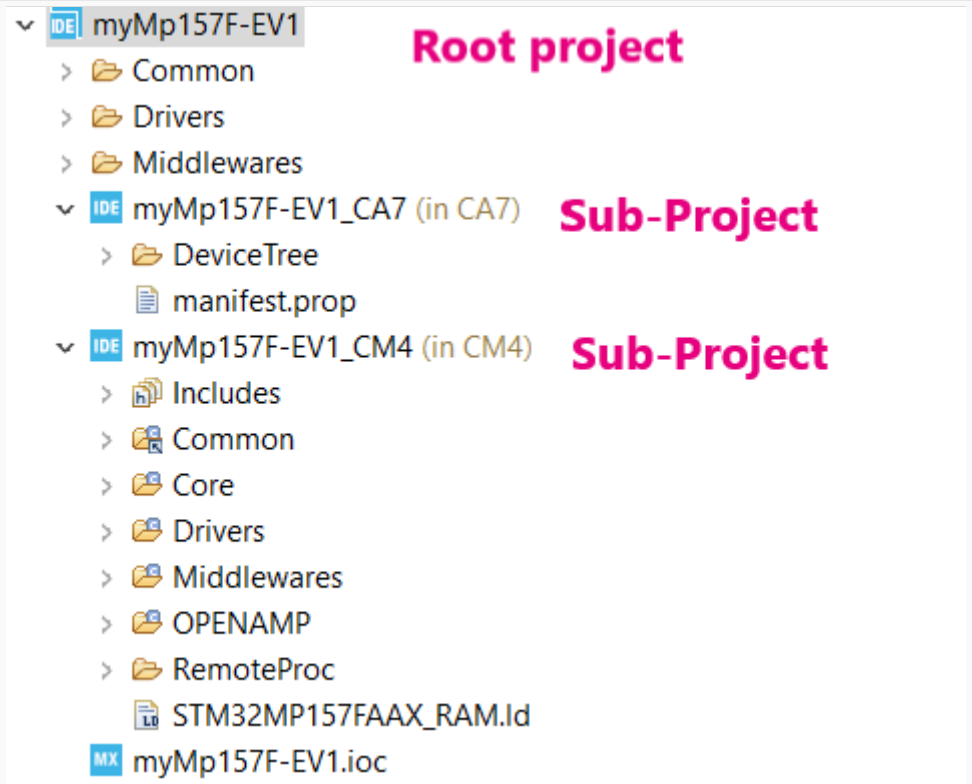
3.2 How to migrate from SW4STM32 to STM32CubeIDE

How to move from SW4STM32 to STM32CubeIDE.

4 Project structure

A hierarchical project structure is created together with the creation of an STM32 MPU project.

The project structure for single-core projects is flat. On the contrary, in a multi-core project, the hierarchical project structure is used. When the user creates or imports an STM32 MPU project, its structure is made of one root project together with sub-projects, referred to as STM32 MCU projects, for each core. A hierarchical structure example is shown below.



Hierarchical project structure

5 Arm® Cortex®-M debug on STM32 MPU device

Two modes are used to debug Arm® Cortex®-M firmware on STM32 MPU devices.

5.1 Engineering mode

Very powerful to debug preliminary Arm® Cortex®-M, the engineering mode implies a specific boot mode: the Engineering Boot Mode where only the Cortex®-M is started. Firmware is loaded via JTAG/SWD into its dedicated RAM.

This mode is not the default one. It must be set via the *"Debug Configuration"* menu, in the *"Debugger"* tab , with *"through JTGA /SWD link (Engineering mode)"* option selected.

Warning

The initialization normally done in the Cortex®-A (such as clock tree setup or others) must be handled by the Arm® Cortex®-M.

Debugging in engineering mode in an STM32 MPU device is very similar to a standard STM32 MCU debug in terms of functionality, except that in this case the Arm® Cortex®-M core has only one dedicated memory, no Flash memory type.

5.2 Production mode

Production mode targets a debug close to the final product. It means to have an STM32 MPU board up and running on the Cortex®-A (Linux®) and a Cortex®-M firmware to debug (usually an STM32CubeIDE project).

The board, also named "target", is booted in the production mode from the Flash memory (the SD card); it is connected to the Host:

- via the Ethernet network, using an Ethernet cable or dedicated USB cable
- and via a USB cable to the ST-LINK probe, giving access to the JTAG/SWD and Linux® console

ST-LINK automatically brings support for Cortex®-A Linux® console via VCP (Virtual COM port). This enables the Target Status widget, visible on the bottom-right of STM32CubeIDE, allowing target IP address discovery. For production mode setup, it is recommended to get the target IP address discovered by the Target Status widget before creating the debug configuration. This principle is depicted in the *Debug Configuration* screenshot below.

Network connection can be set up in two ways:

- Ethernet
  - managed network: IP address attributed by DHCP server
  - unmanaged network: IP address to be manually configured
- USB
  - using the dedicated USB OTG connection for Ethernet point to point.

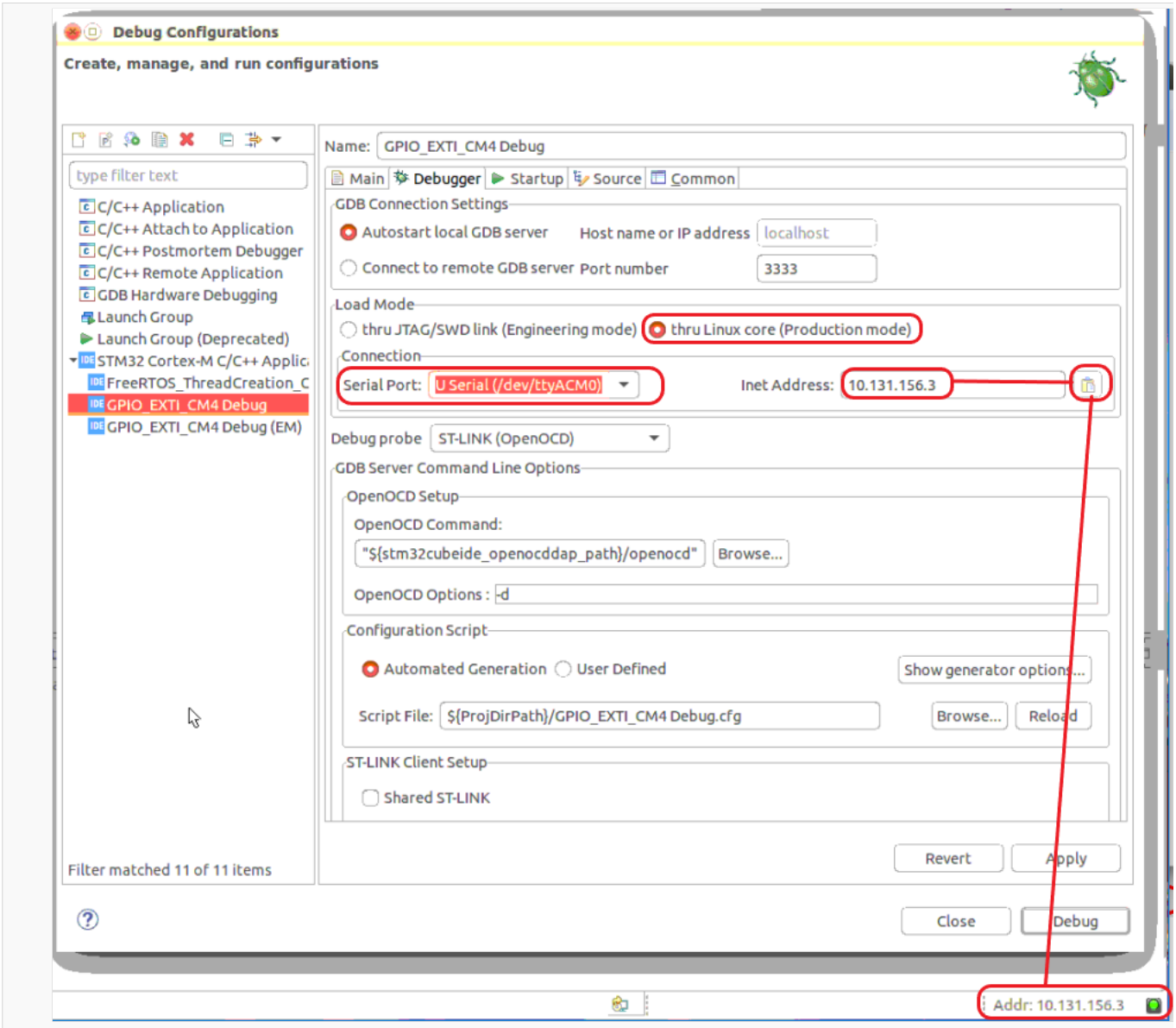
Debug is performed though the following workflow:

1. The firmware built binary is transferred to the target using the network (SSH protocol), more precisely to the Cortex®-A Linux® file system.

2. Firmware is loaded to Cortex®-M core using the “remoteproc” framework.

3. Finally the debug session is started via the JTAG/SWD connection

The production mode is the default one when you create a new debug configuration. It is automatically set via the *"Debug Configuration"* menu.

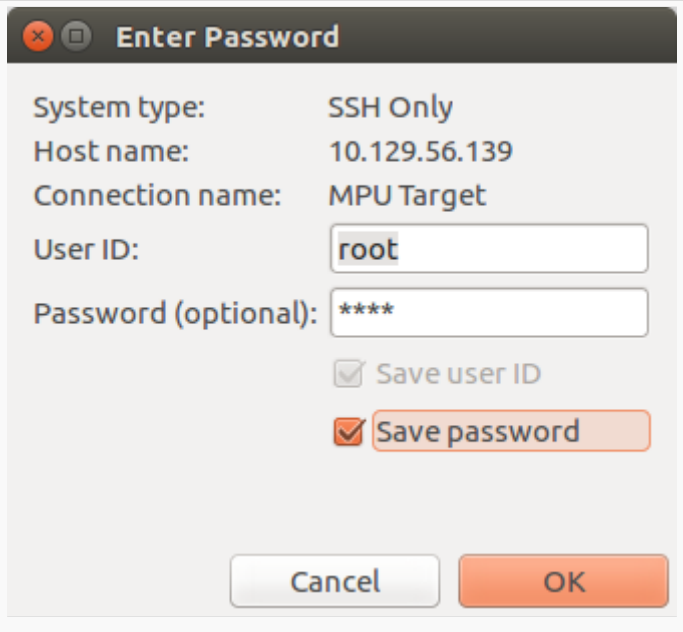


Debug configuration, production mode activated.

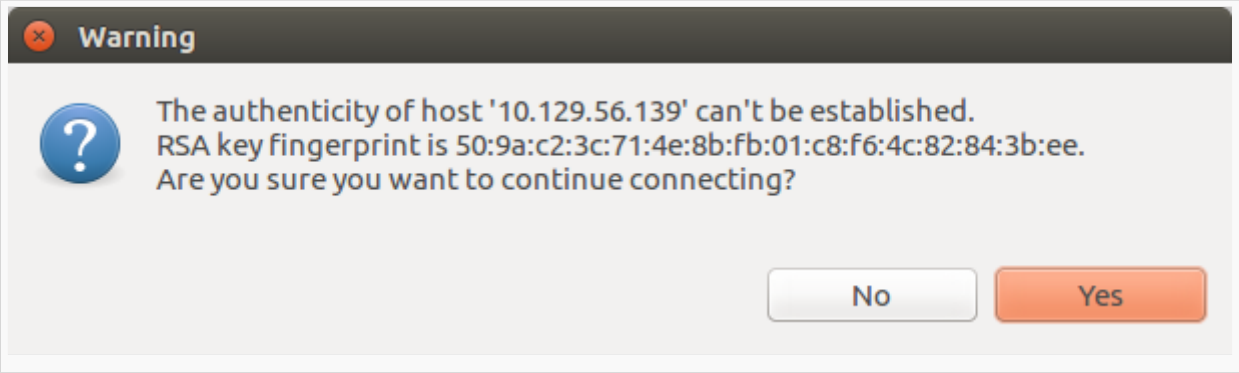
More information on how to use STM32CubeIDE target status are given in article How to use the Target Status widget in STM32CubeIDE.

At debug launch, some specific pop-up appear:

- The SSH Password must be completed: the default one is **root.**



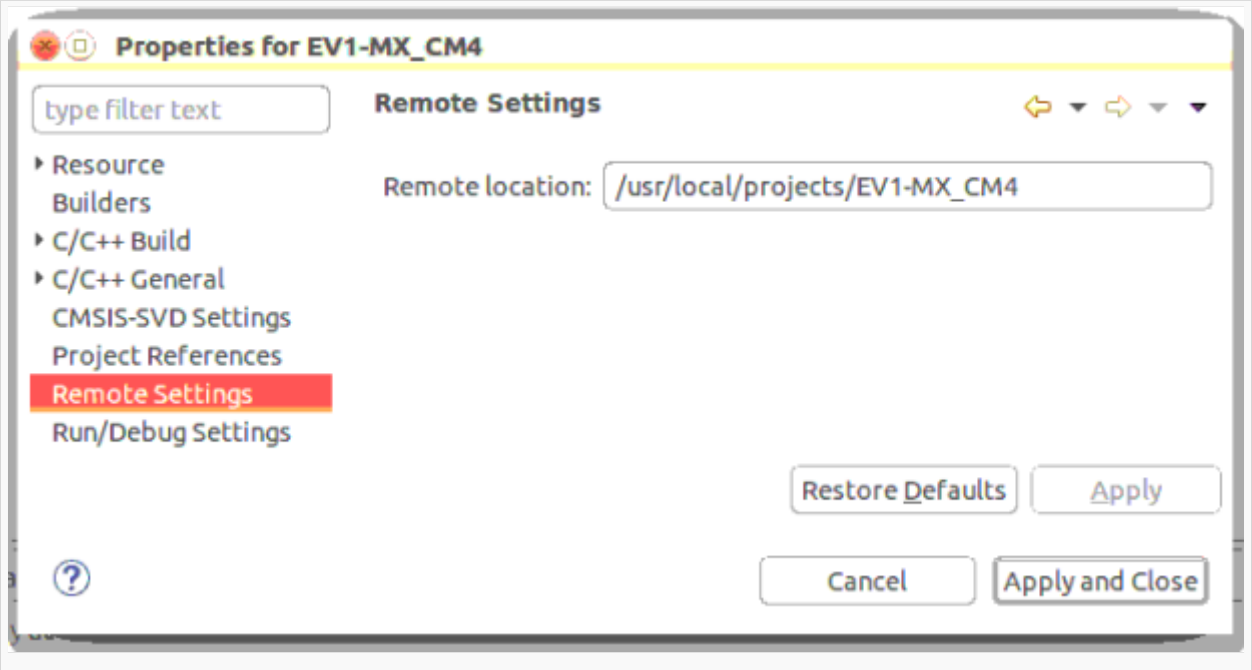
- The RSA key must be approved.



Defining an Arm® Cortex®-M project in an STM32 MPU context also means to define where are downloaded in the Arm® Cortex®-A Linux® file system:

- the Cortex®-M firmware: **<ProjectName>.elf**
- the load/unload script **fw_cortex_m4.sh** used by the STM32CubeIDE if present it can be customized
- and a **README**, informing a user having a direct connection onto the target.

This is the purpose of "Remote Path Setting" property which is linked to the project. Its default value at creation is /usr/local /project/<ProjectName>, but can also be changed using the *Remote Settings* properties item.



Inside project structure,directory "<ProjectName>_CM4/Core/RemoteProc/" defines the tree downloaded when debugging in the production mode.

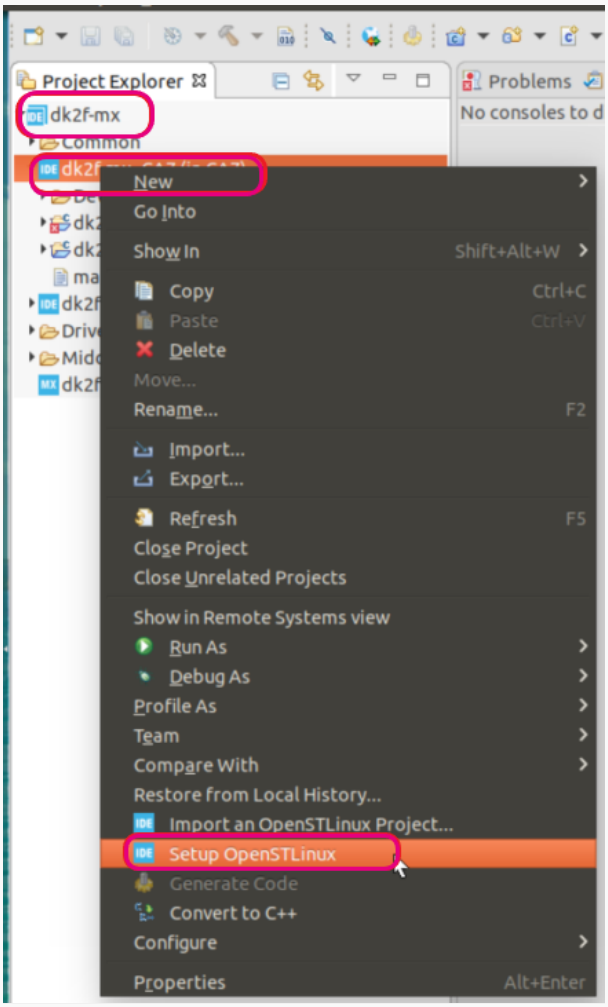
5.3 ST-LINK sharing support

From STM32CubeIDE v1.2.0, it is possible to connect to an ST-LINK probe several applications, in parallel of openOCD. This support relies onto '**ST-Link Server**'.
It is enabled by checking *Shared ST-Link* inside *"Debug Configuration > Debugger > ST-Link Client Setup"*.

6 OpenSTLinux project support - Cortex®-A

From release 1.4.0 on **Linux® host ONLY**, STM32CubeIDE supports OpenSTLinux projects and its associated Yocto Project® SDK. Inside STM32CubeIDE, this support means two new Eclipse® plugins (SDK & Sources) to be installed, directly from the embedded CA7 project menu context:

- *Setup OpenSTLinux*
- *Import an OpenSTLinux Project...*



OpenSTLinux Contextual Menu

These Eclipse® plugins embed official packages from OpenSTLinux 2.0.

Note that a minimum disk space of 5 GBytes is needed under /tmp during the installation phase.

6.1 How to install Yocto Project® SDK in STM32CubeIDE

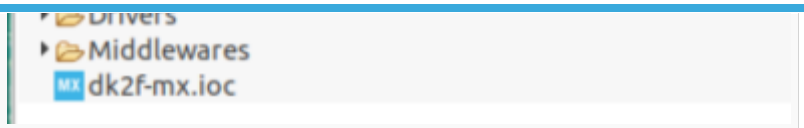
We focus in the article How to install the Yocto Project SDK in STM32CubeIDE on various ways to install the Yocto Project® SDK or point to an already existing SDK installation.

6.2 How to manage OpenSTLinux projects

All projects from OpenSTLinux can be imported inside STM32CubeIDE are available for compilation with Yocto Project® SDK.

Regarding project structure, these OpenSTLinux projects enrich the Cortex®-A part as depicted hereafter. Note that this support is starting from OpenSTLinux v2.0.


dk2f-mx
Common
dk2f-mx_CA7 (in CA7)
DeviceTree
Device Trees from CubeMX
dk2f-mx_CA7_linux-5.4.31 (in linux-5.4.31)
Build Targets
Linux project
Binaries
Includes
build
linux-5.4.31
dk2f-mx_CA7_optee_os-3.9.0 (in optee_os-3.9.0)
Build Targets
Optee project
Includes
optee-os-stm32mp-3.9.0.r1
Makefile.sdk
dk2f-mx_CA7_tf-a-2.2 (in tf-a-2.2)
Build Targets
TF-A project
Binaries
Includes
boot_msg
build
flashlayout
tf-a-stm32mp-2.2.r1
Makefile.sdk
dk2f-mx_CA7_tf-a-ssp-2.2 (in tf-a-ssp-2.2)
Build Targets
TF-A SSP project
Binaries
Includes
build
tf-a-stm32mp-ssp-2.2.r1
Makefile.sdk
dk2f-mx_CA7_u-boot-2020.01 (in u-boot-2020.01)
Build Targets
U-boot project
Includes
u-boot-stm32mp-2020.01
Makefile.sdk
manifest.prop
dk2f-mx_CM4 (in CM4)
Binaries
Cortex-M4 project
Includes
Common
Core
Drivers
Middlewares
OPENAMP
Debug
RemoteProc
dk2f-mx_CM4 Debug.cfg
STM32MP157FAAX_RAM.ld



Hierarchical Project Structure with OpenSTLinux projects

6.3 How to debug a user space application

Release 1.6.0 provides support for creating, building and debugging projects aiming to run into the Linux® user space of STM32 MPUs. This includes executable, static and shared library.

7 Tips & tricks

- How to use the Target Status widget in STM32CubeIDE
- How to set up proxy and P2P Ethernet connection with STM32CubeIDE
- How to use the RSE Perspective with STM32CubeIDE
- How to setup target password in STM32CubeIDE
- How to copy and paste in the STM32CubeIDE console
- How to debug with Serial Wire Viewer tracing on STM32MP15