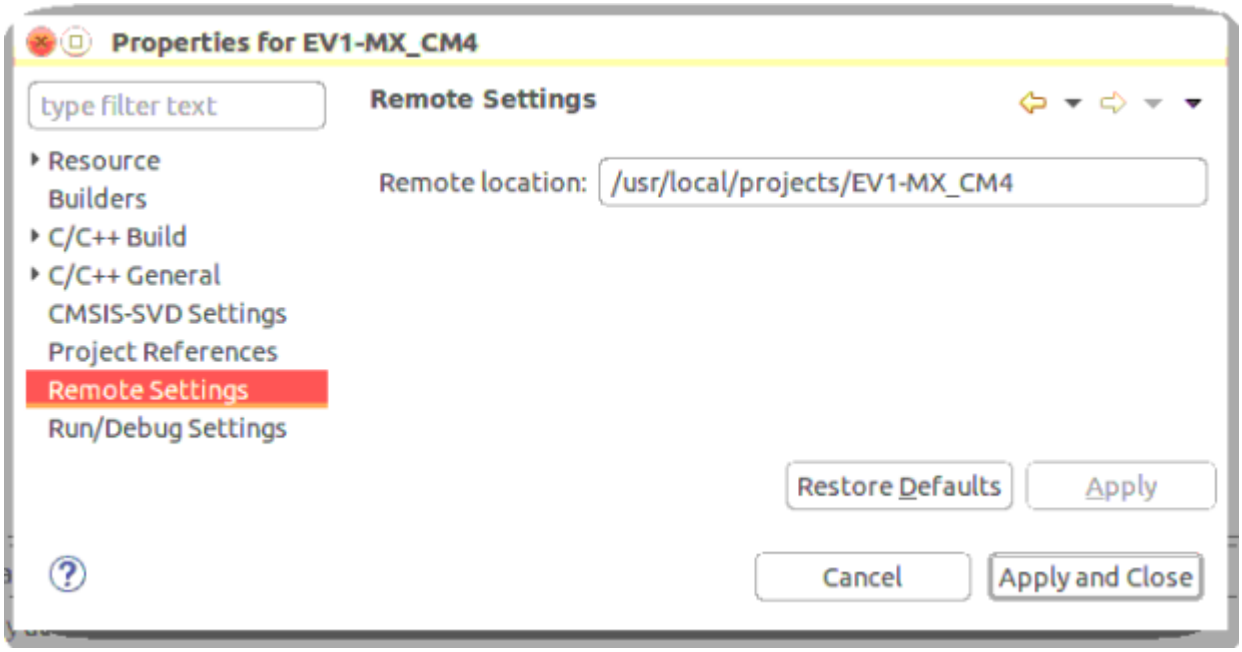
File:CubeIDE-Dbg-RemotePath.png

Contents

Stable: 09.11.2020 - 21:28 / Revision: 31.08.2020 - 15:20

File:CubeIDE-Dbg-RemotePath.png

- File
- File history
- File usage
- Metadata



No higher resolution available.
CubeIDE-Dbg-RemotePath.png (617 × 322 pixels, file size: 42 KB, MIME type: image/png)
A quality version of this page, approved on *9 November 2020*, was based off this revision.

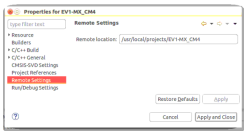
Source not applicable (screen capture)

Remote Path property

File:CubeIDE-Dbg-RemotePath.png

File history

Click on a date/time to view the file as it appeared at that time.

	Date/Time	Thumbnail	Dimensions	User	Comment
current	14:49, 27 July 2020		617 × 322 (42 KB)	Nathalie Sangouard (talk | contribs)	Merge upload

- You cannot overwrite this file.

File:CubeIDE-Dbg-RemotePath.png

File usage

The following 2 pages link to this file:

- STM32CubeIDE
- Draft:User:Ludovic Rattin

Metadata

This file contains additional information, probably added from the digital camera or scanner used to create or digitize it. If the file has been modified from its original state, some details may not fully reflect the modified file.

Horizontal resolution 47.24 dpc

Vertical resolution 47.24 dpc

Stable: 19.04.2021 - 08:24 / Revision: 15.02.2021 - 16:11

A quality version of this page, approved on *19 April 2021*, was based off this revision.

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

For more information about STM32CubeIDE please refer to its user manual.

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 to 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 on Arm® Cortex®-M

File:CubeIDE-Dbg-RemotePath.png

2 How to install STM32CubeIDE

It 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 1.5.0**
	• 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 1.5.0
		• Download the all-in-one Windows installer from my.st.com
		  • *Windows Installer - STM32CubeIDE-Win*

Installation guide	• Please refer to *STM32CubeIDE Installation guide (UM2563)* available on my.st.com.	

User manual	• When the installation is over, please see additional information about the 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 Release Notes *STM32CubeIDE release v1.5.0* from my.st.com	

Minor releases may then be available from update site, please 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 if you are moving from SW4STM32 to STM32CubeIDE or if you are starting a new project with STM32CubeIDE.

3.1 How to get started with STM32CubeIDE from scratch

How to get started with STM32CubeIDE from scratch.

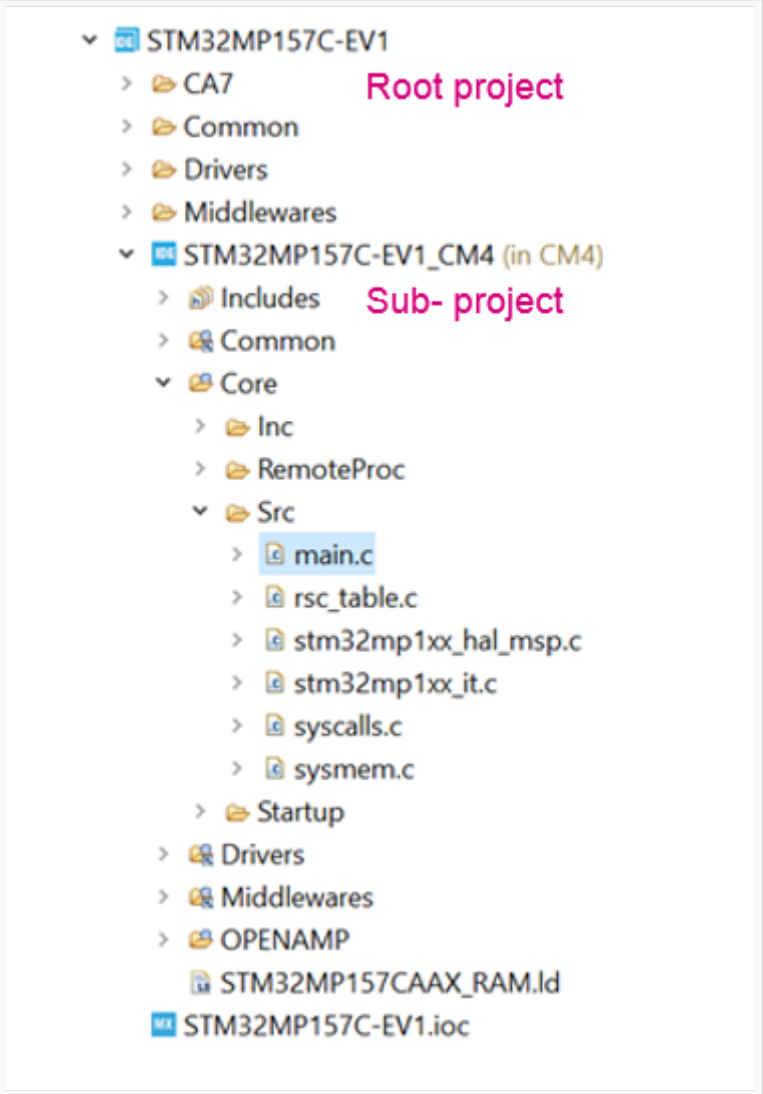
3.2 How to move from SW4STM32 to STM32CubeIDE

How to move from SW4STM32 to STM32CubeIDE.

4 Project structure

A hierarchical project structure is created in same time as 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 STM32 MCU projects, for each core. A hierarchical structure example is shown below.



Hierarchical project structure

File:CubeIDE-Dbg-RemotePath.png

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

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

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 Cortex-M is started. Firmware is loaded via JTAG/SWD into its dedicated RAM.

This mode is not the default one, it has to be set via the *"Debug Configuration"* menu, in *"Debugger"* tab , *"thru JTGA/SWD link (Engineering mode)"* option selected.

Warning

initialization normally done in Cortex-A like (clock tree set-up, ...) have to be handled by Arm Cortex-M.

Debugging in Engineering Mode in STM32 MPU device is very similar to a standard STM32 MCU debug in term of functionality, expect that here Arm Cortex-M has only a dedicated memory, no flash memory type.

5.2 Production mode

Production mode targets a debug close to 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).

Board, aka named as "target", is booted in Production mode from flash memory (e.g. SD card) ; it is connected to the Host:

- via Ethernet network, using Ethernet cable or dedicated USB cable
- and via USB cable to ST-Link probe, giving access to JTAG/SWD and Linux Console.

ST-Link automatically brings support for Cortex-A Linux console via VCP (Virtual COM Port). This enables 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 Target Status widget before creating Debug Configuration. This principle is depicted in *Debug Configuration* screenshot below.

Network connection can be set up by 2 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.

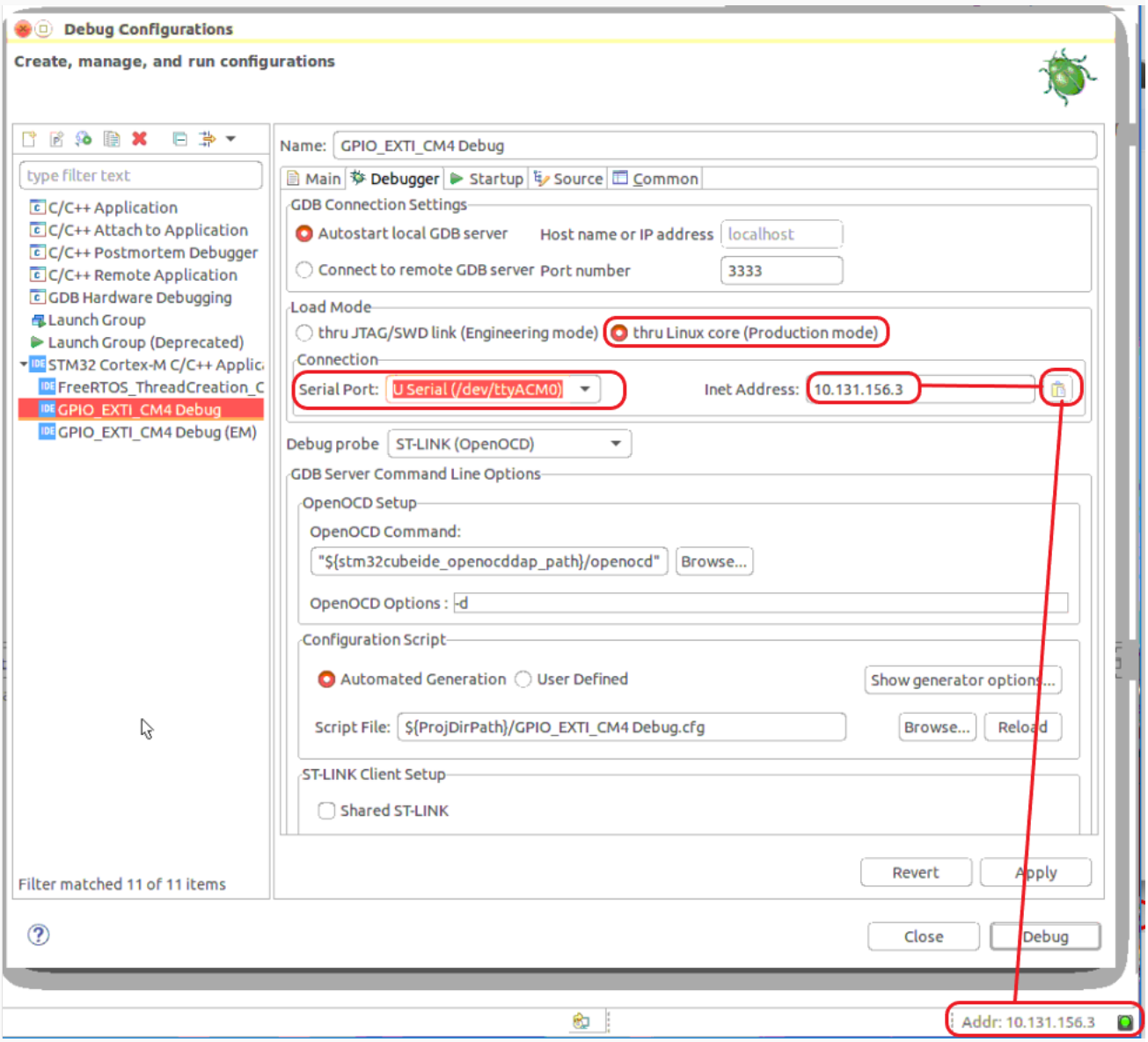
Debug is performed thanks to the following workflow:

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. The firmware is loaded to Cortex-M core thanks to the “remoteproc” framework.

3. Finally the Debug session is started via 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.



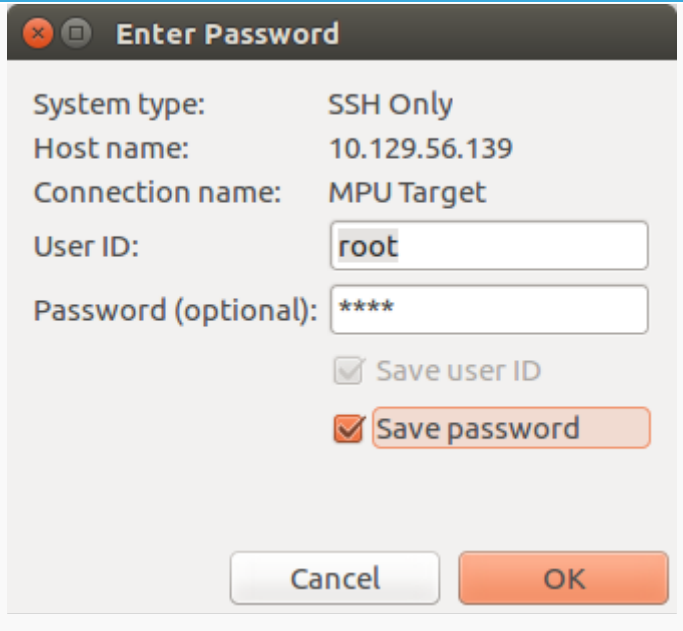
Debug configuration, production mode activated.

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

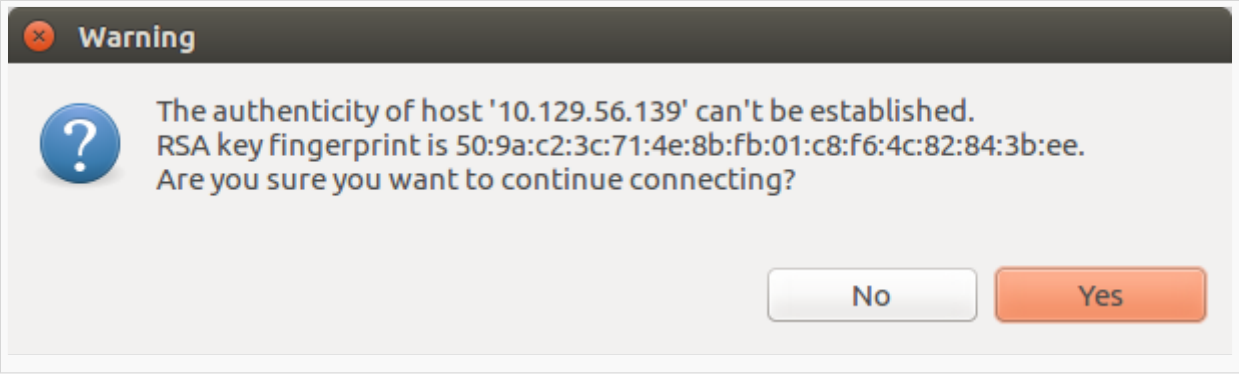
At debug launch, some specific pop-up appear:

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

File:CubeIDE-Dbg-RemotePath.png



- The RSA key has to be approved.

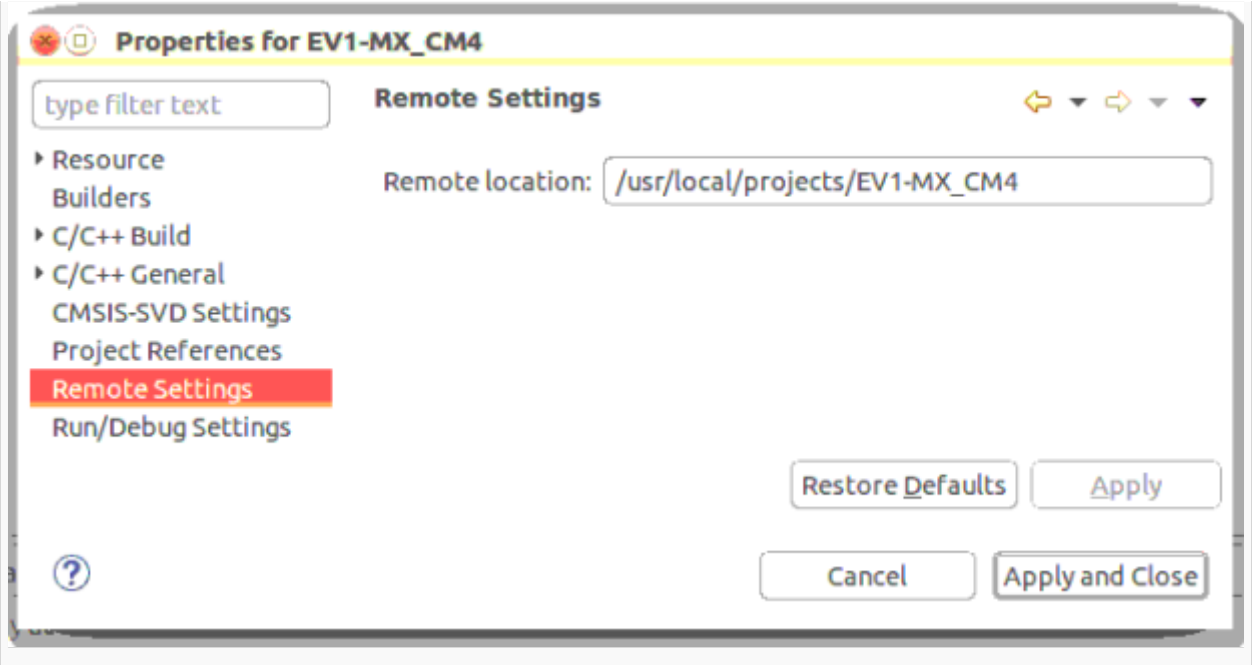


Defining an Arm Cortex-M project in an STM32 MPU context also means to define where are downloaded on Arm Cortex-A Linux file system:

- 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 thanks to the Remote Settings properties item.

File:CubeIDE-Dbg-RemotePath.png



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

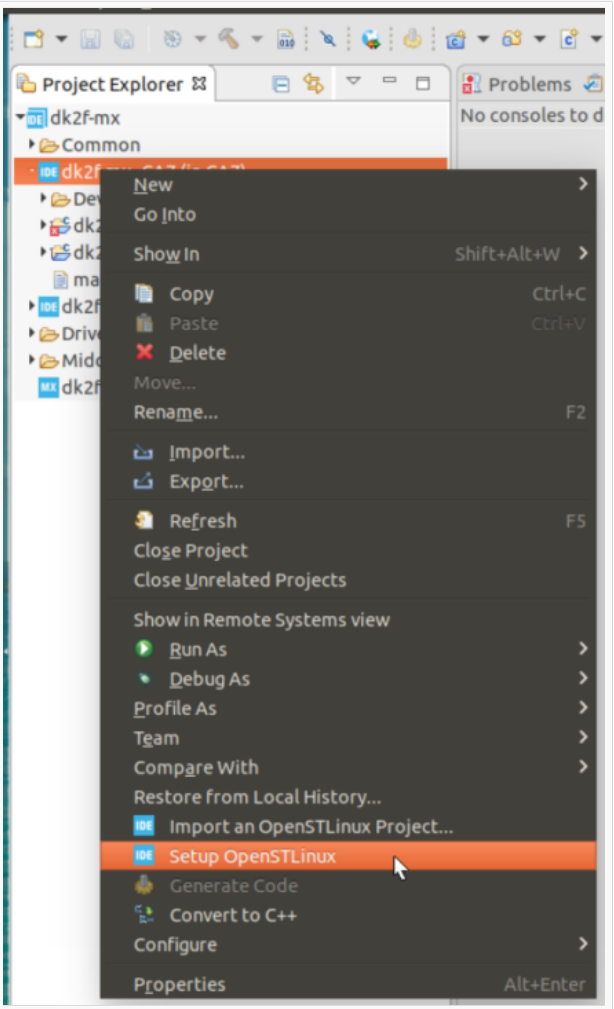
5.3 ST-Link sharing support

From STM32CubeIDE v1.2.0, it is possible to connect to ST-Link probe several application, in parallel of openOCD. This support relies onto '**ST-Link Server**'.
It is enable 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 SDK. Inside STM32CubeIDE, this support means two new Eclipse plugins (SDK & Sources) to be installed, directly from embedded CA7 project menu context:

- Setup OpenSTLinux
- Import an OpenSTLinux Project...



OpenSTLinux Contextual Menu

Those Eclipse plugins embed official packages from OpenSTLinux 2.0.

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

6.1 How to install Yocto SDK in STM32CubeIDE

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

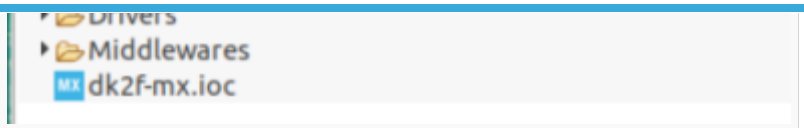
File:CubeIDE-Dbg-RemotePath.png

6.2 How to manage OpenSTLinux projects

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

Regarding Project Structure, those OpenSTLinux projects enrich the Cortex-A part as depicted hereafter. Note that this support is starting from OpenSTLinux v2.0.

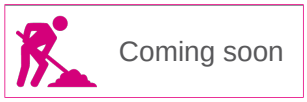
File:CubeIDE-Dbg-RemotePath.png



Hierarchical Project Structure with OpenSTLinux projects

File:CubeIDE-Dbg-RemotePath.png

7 FAQ



How to copy and paste in the STM32CubeIDE console?

Operating System

GNU dedugger, a portable debugger that runs on many Unix-like systems

(Software)Integrated development/design/debugging environment

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

debug and test protocol, named from the Joint Test Action Group that developed it

Serial Wire Debug

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

also known as

SD memory card (https://www.sdcard.org)

spelling for older versions of STLink, ST in-circuit debugger and programmer for the STM8 and STM32 microcontroller families

Virtual COM Port. ST-Link Console support

Dynamic Host Configuration Protocol (See https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol for more details)

USB On-The-Go (Capability/type of USB port, acting primarily as USB device, to also act as USB host. Also known as USB OTG.)

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)