AN5360

Application note

Getting started with projects based on the STM32MP1 Series in STM32CubeIDE

Introduction

This application note describes how to get started with projects based on the STM32MP1 Series in STMicroelectronics STM32CubeIDE integrated development environment.

1 General information

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

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

1.1 Prerequisites

The following tools are prerequisites for understanding the tutorial in this document and developping an application based on the STM32MP1 Series:

- STM32CubeIDE 1.1.0 or newer
- STM32Cube_FW_MP 1.1.0 or newer
- STM32CubeMX 5.4.0 or newer

Users are advised to keep updated with the documentation evolution of the STM32MP1 Series at www.st.com/en/microcontrollers-microprocessors/stm32mp1-series.

1.2 The use cases in this document

In the STM32CubeIDE context, users have different ways to explore and get started with the development of projects based on the STM32MP1 Series. From the list below, select the description that best fits the use case considered and refer to the corresponding section in this application note:

- I already have an SW4STM32 project with an `ioc` file:
  Refer to Section 2.2 Import an SW4STM32 project with an ioc file
- I already have an SW4STM32 project without an `ioc` file:
  Refer to Section 2.3 Import an SW4STM32 project without an ioc file
- I want to learn with and explore example projects:
  Refer to Section 2.5 Import a project from the STM32CubeMP1 MCU Package
- I want to start a first STM32MP1 project:
  - Empty project – No STM32CubeMX support for maximum flexibility.
    Refer to Section 2.4 Create an empty project based on the template in the STM32CubeMP1 MCU Package
  - STM32CubeMP1 project – STM32CubeMX-managed project.
    Refer to Section 2.1 Create a new STM32 project

1.3 Specific features of the STM32MP1 Series

The way the target is booted is important. Boot pins are set by the user on STMicroelectronics boards by means of switches. For the STM32MP157C-EV1 Evaluation board, related information is provided in the *Boot options* section of the user manual (UM2535). More generally, information is also available from STMicroelectronics MPU wiki at wiki.st.com/stm32mpu in the *Boot related switches* section of the board being used.

Two boot modes are considered:

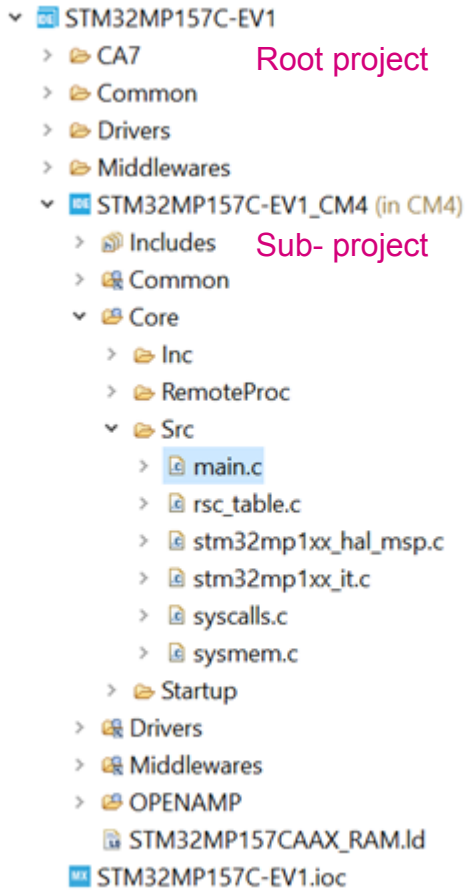
- **Production boot mode**: Linux® usually boots on an SD card, but is also capable to boot through an on-board NAND or NOR. The Cortex®-M4 `elf` is downloaded through the network and loaded by the OpenAMP framework. It is possible to debug the application via JTAG/SWD by attaching to a running target.
- **Engineering boot mode**: The Cortex®-A7 is effectively disabled and the application is downloaded directly to the Cortex®-M4 through JTAG/SWD. Using this mode, the application is debugged like for any standard Cortex®-M4 device.

Additional consequences of the choice between production and engineering modes are dealt with further in Section 3.1 Debug modes and Section 3.2 Target status.

1.3.1 STM32MP1 project structure

When an STM32MP1 project is created, its structure is automatically made hierarchical. 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 STM32MP1 project, it consists of one root project together with sub-projects, referred to as MCU projects, for each core. A hierarchical structure example is shown in Figure 1.

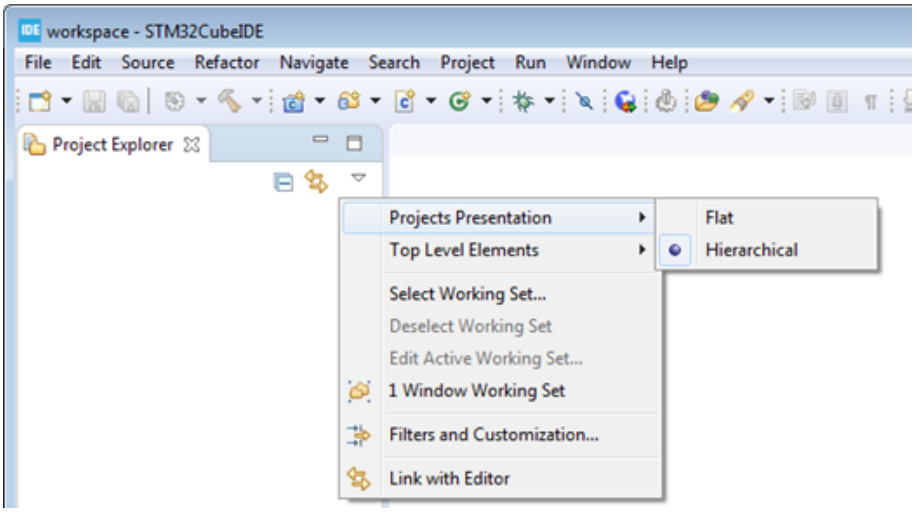
Figure 1. Hierarchical project structure



The root project is a simple container that allows sharing common code between the cores. The root project can contain neither build nor debug configurations. However, the MCU projects are real CDT projects that can contain both build and debug configurations.

If the project is not shown in a hierarchical structure, this can be changed as shown in Figure 2.

Figure 2. Setting the project hierarchical view



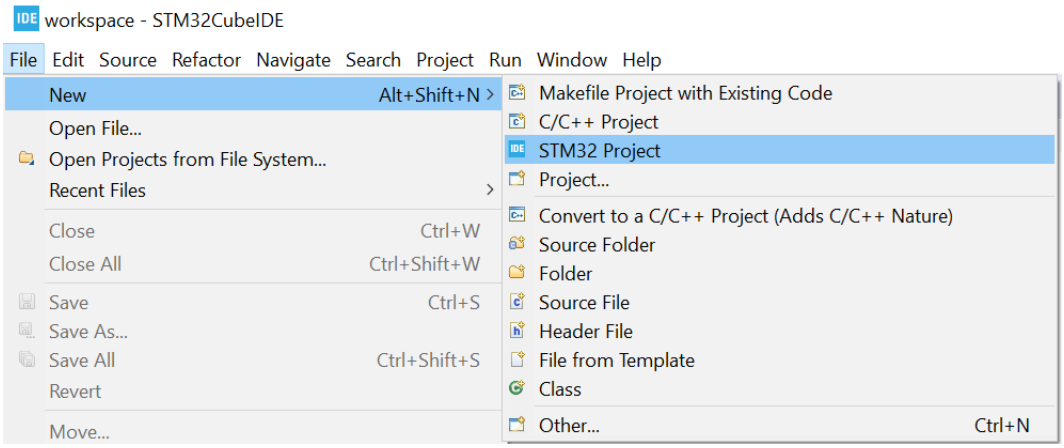
2 Create and import projects

This chapter describes how to create or import projects for the STM32MP1 Series.

2.1 Create a new STM32 project

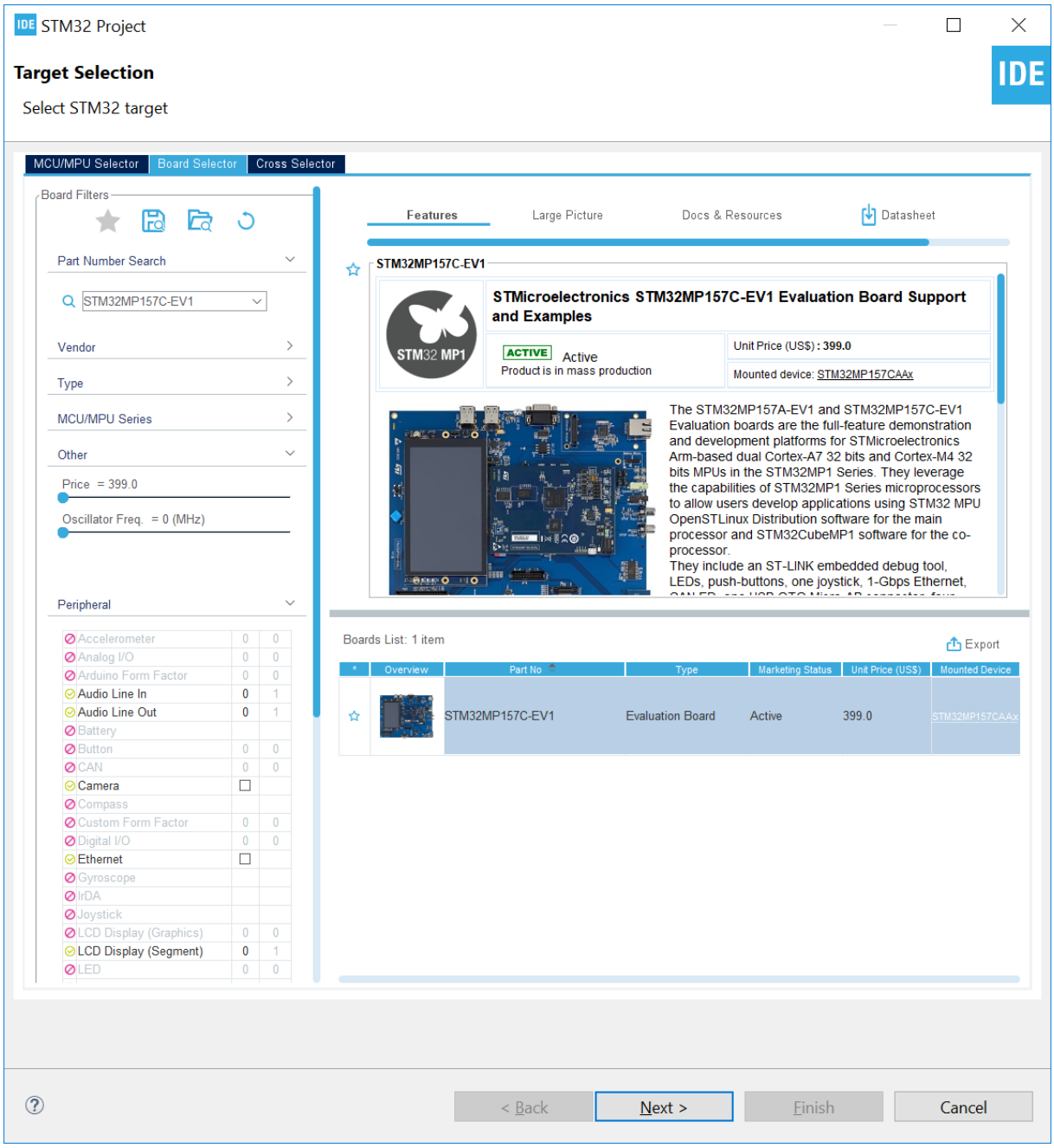
To start a new project, go to [**File**]>[**New**]>[**STM32 Project**] as shown in Figure 3.

Figure 3. New STM32 project



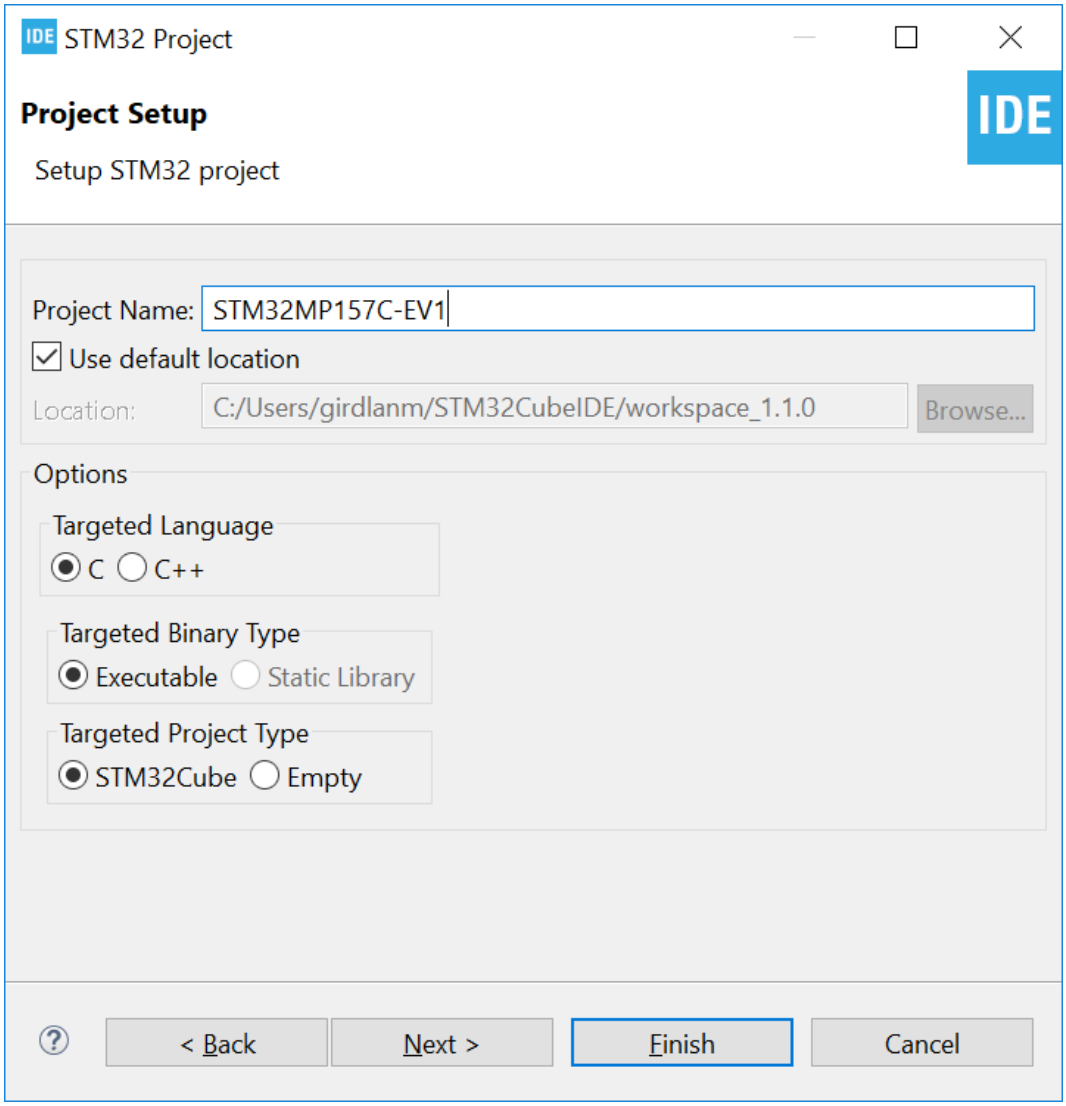
Select the desired MCU or board. In the example illustrated in Figure 4, the selected board is the STM32MP157C-EV1. Click on [**Next >**].

Figure 4. Target selection



After the target selection comes the project setup step shown in Figure 5. The *Targeted Project Type* setting determines whether the project gets generated by STM32CubeMX or not. An *Empty* project is a skeleton of a project that needs building upon while *STM32Cube* indicates an STM32CubeMX-managed project.

Figure 5. Projet setup

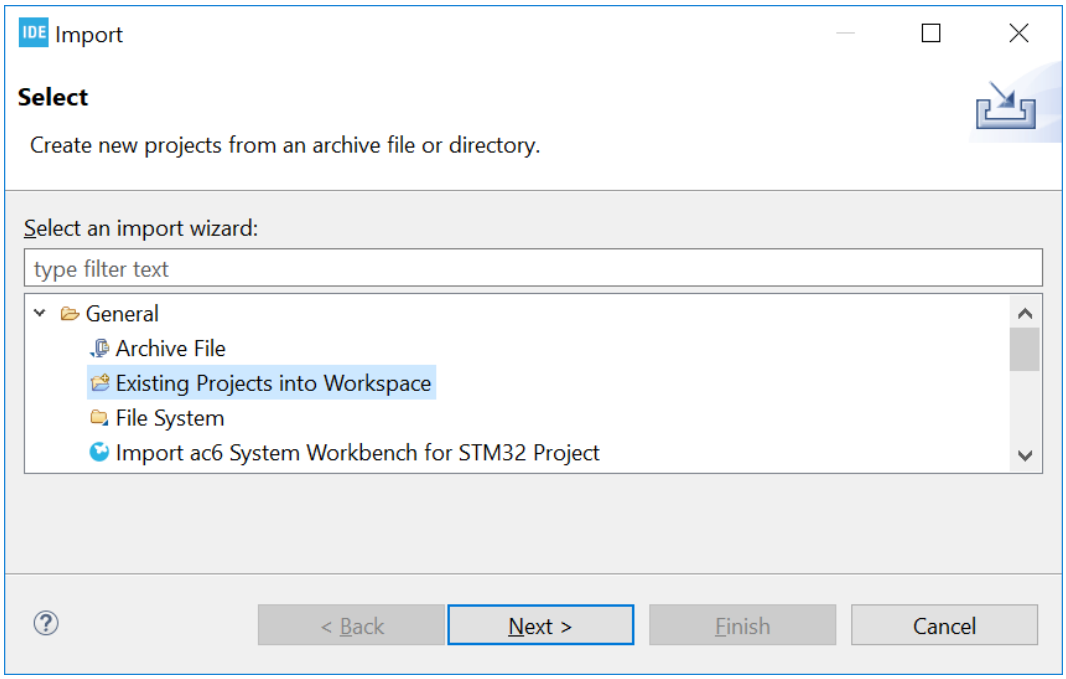


2.2 Import an SW4STM32 project with an ioc file

If the project already contains an ioc file, the easiest way to import the project into a working STM32CubeIDE environment is to copy it and open the copy through STM32CubeMX stand alone, then, in the *Project Manager*, change the *Toolchain / IDE* to STM32CubeIDE and regenerate the project.

After the project is regenerated, go to [**File**]>[**Import…**] and choose to import it as an *Existing projects into workspace* as shown in Figure 6.

Figure 6. Import an existing projet with an ioc file

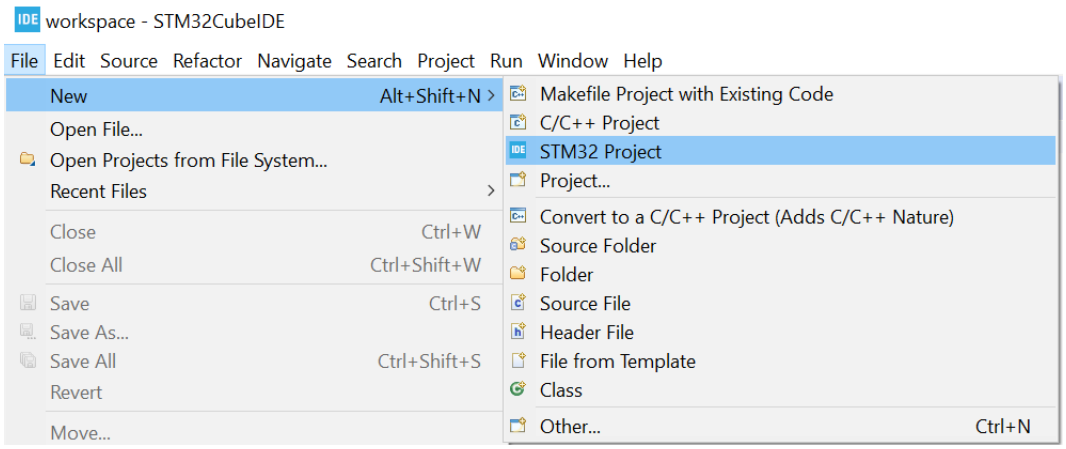


Then copy the code inside the different `/* USER CODE BEGIN */` blocks that exist in the project into the new STM32CubeIDE environment.

2.3 Import an SW4STM32 project without an `ioc` file

To make sure the project gets a hierarchical structure, the recommended way is to go to [**File**]>[**New**]>[**STM32 Project**] as shown in Figure 7.

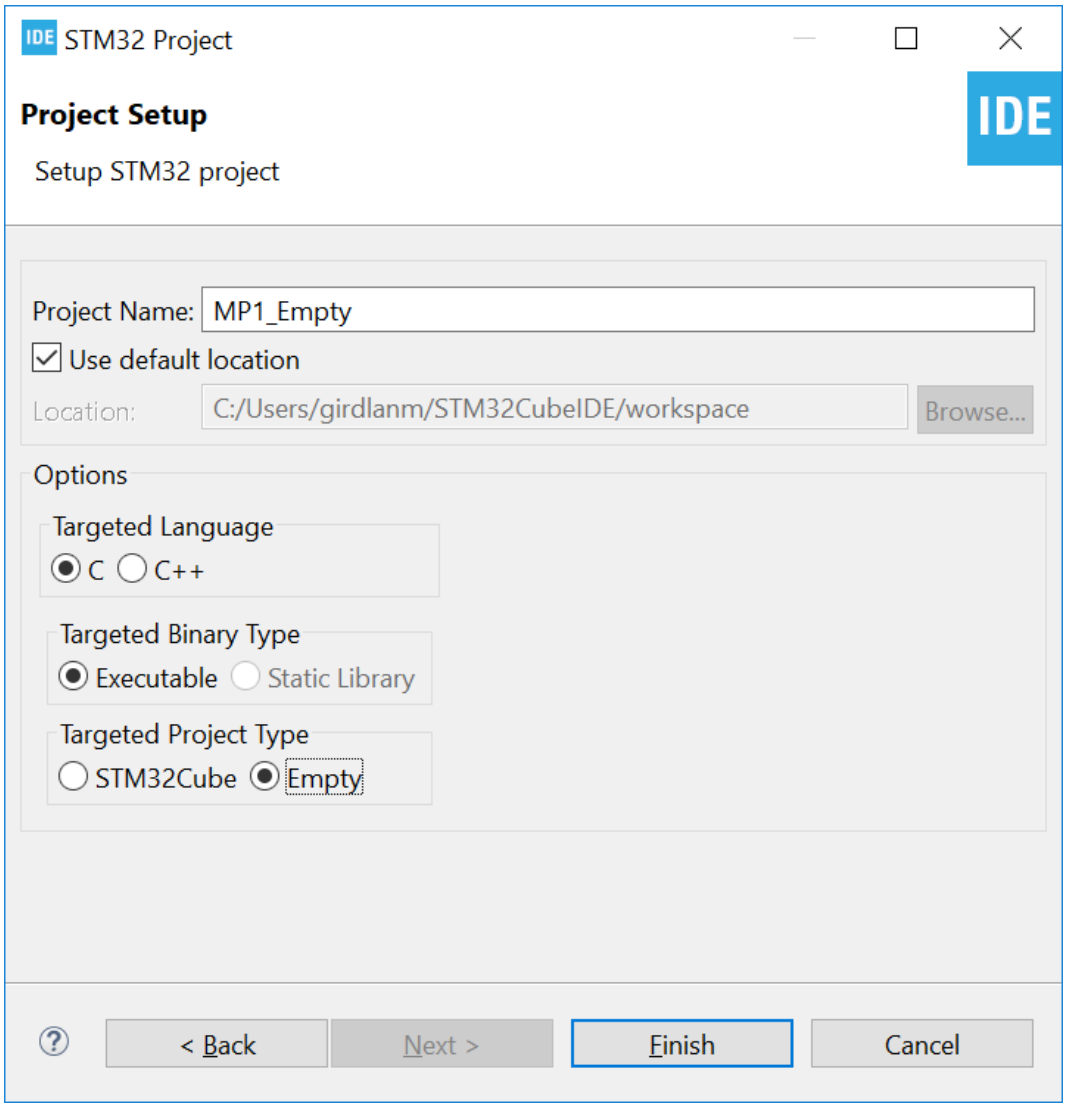
Figure 7. New STM32 project



Select the device for the project being imported and click on [**Next >**].

When setting up the project as shown in Figure 8, make sure the *Targeted Project Type* is set to *Empty* and click on [**Finish**].

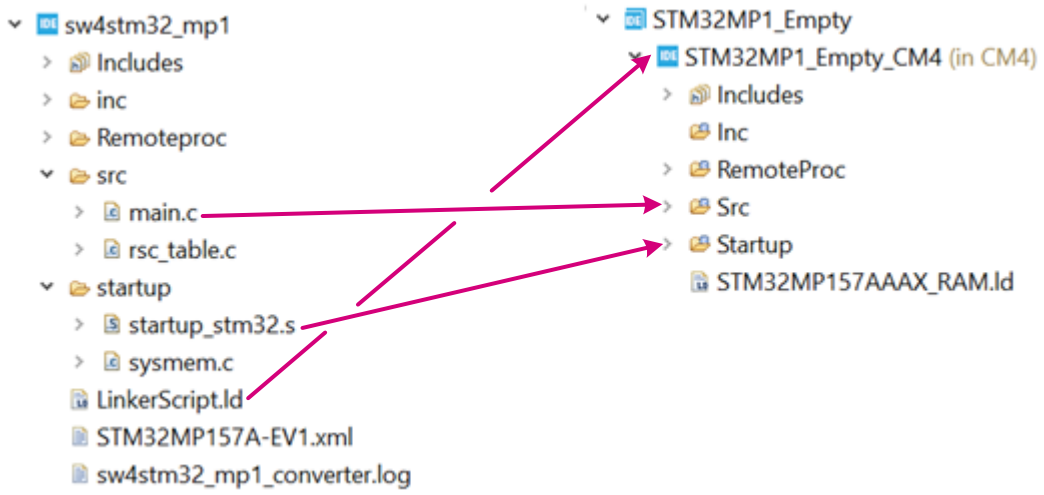
Figure 8. Projet setup



After the empty hierarchical project is generated:

1. Go to [**File**]>[**Import…**]
2. Import the SW4STM32 project as *Import ac6 System Workbench for STM32 Project*
3. Copy and paste the project content into the sub-project of the empty project by means of STM32CubeIDE project explorer as shown in Figure 9

Figure 9. Copy project content to empty sub- project



Note: It is not recommended to import the `.cproject`, `.project` *or* `.settings` *files.*

It is important to remember to also configure the same build settings that was used previously while the project was in the SW4STM32 environment. If the project contains linked resources those needs to be updates to point to the correct resource in the file system.

This process is necessary because when importing a project from SW4STM32 without any special treatment and that does not have an ioc-file then it will be imported into STM32CubeIDE with a flat project structure.

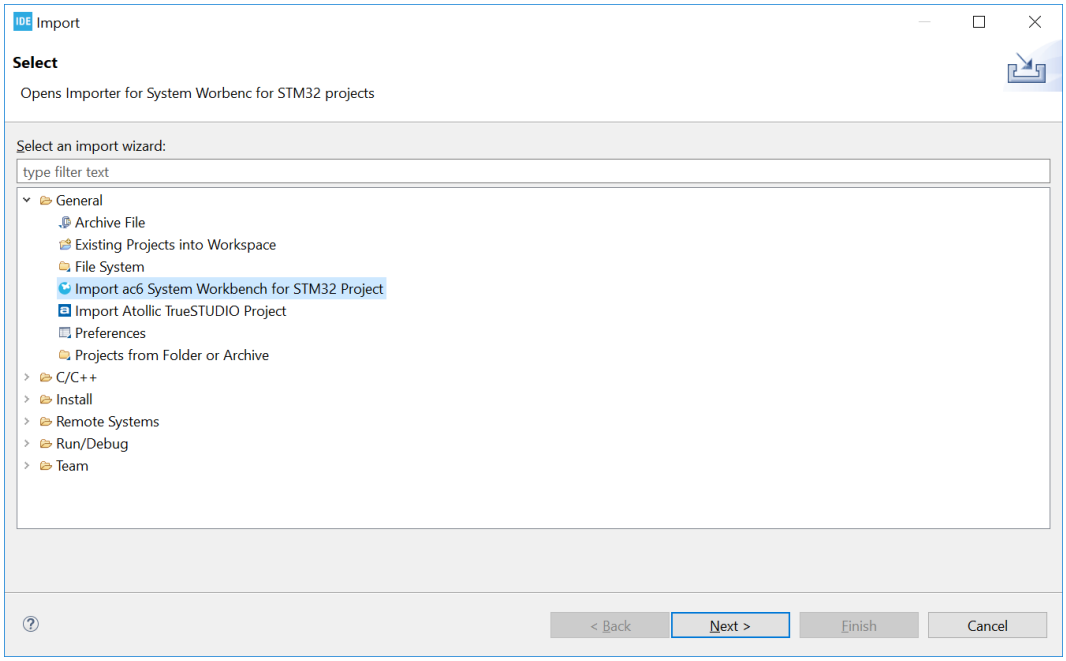
2.4 Create an empty project based on the template in the STM32CubeMP1 MCU Package

Follow the same steps as in Section 2.3 but use STM32Cube_FW_MP firmware in the STM32CubeMP1 MCU Package as input.

2.5 Import a project from the STM32CubeMP1 MCU Package

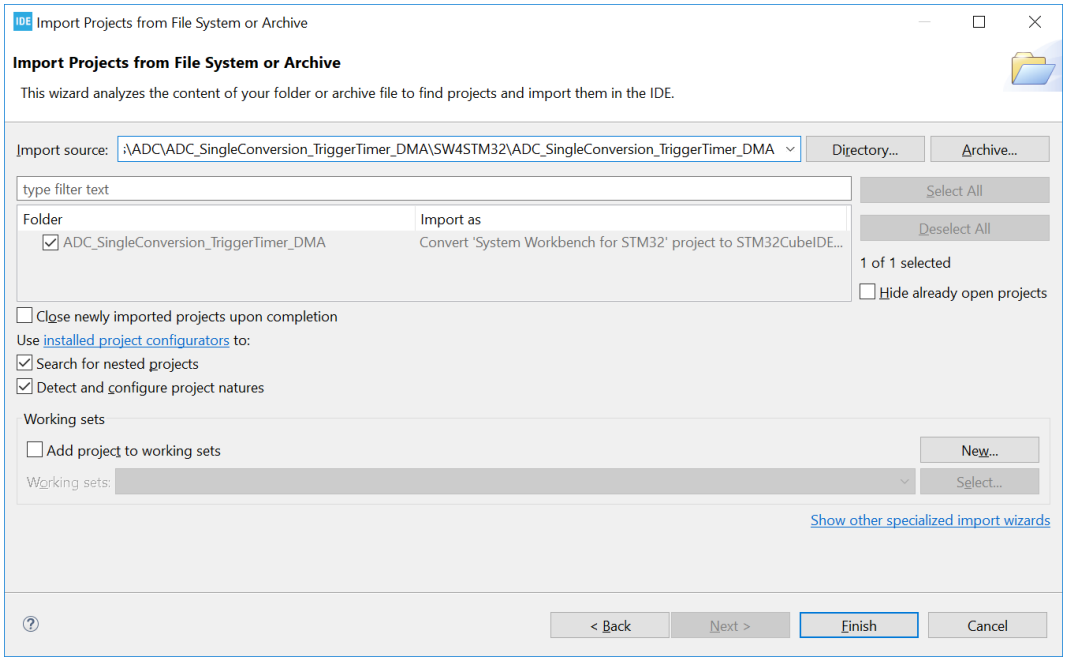
In order to import the STM32Cube firmware project into STM32CubeIDE, go to [**File**]>[**Import**] and select *Import ac6 System Workbench for STM32 Project* as shown in Figure 10 and click on [**Next >**].

Figure 10. Import of firmware project info STM32CubeIDE



Then select the correct project. A project example is by default located at `$HOME\STM32Cube\Repository\STM32Cube_FW_MP1_VX.X.X\Projects\STM32MP157C-EV1\Examples\ADC\ADC_SingleConversion_TriggerTimer_DMA\SW4STM32\ADC_SingleConversion_TriggerTimer_DMA`.

Figure 11. Firmware project selection



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

3 Debugging

This chapter highlights some of the points to bear in mind while debugging a device in the STM32MP1 Series.

3.1 Debug modes

There are two modes for debugging a device in the STM32MP1 Series, the production mode and the engineering mode.

Production mode

The production mode makes full use of the MPU potential by including the Cortex®-A7. It makes it possible to use the Cortex®-A7 to drive the application while having the Cortex®-M4 available for run-time critical tasks, taking the following points into consideration:

- To enable the production mode, the switches on the board must be set correctly. Consult STMicroelectronics MPU wiki at wiki.st.com/stm32mpu in the *Boot related switches* section of the board being used. For the STM32MP157C-EV1 Evaluation board, related information is provided in the *Boot options* section of the user manual (UM2535).
- Firmware is downloaded to the embedded Linux® file system and then uploaded to the Cortex®-M4 through the `remoteproc` framework. Due to the fact that the Cortex®-M4 core is started by Linux®, there is no way to monitor the early startup phase of the debug session as it attaches to a running target. If the monitoring of the application startup phase is required, one possibility is to modify the startup code of the Cortex®-M4 application to have a busy-wait loop based on a register value and then manually set the release value to the register through the debug session to release the Cortex®-M4.
- The target needs to be connected to a network and Linux® must be running. Make sure that the status light is green and an IP address is presented to know that the connection is up and running (refer to Section 3.2).
- In this mode, the Cortex®-A7 Linux® core gives commands to the Cortex®-M4.

Engineering mode

- To enable the engineering mode, the switches on the board must be set correctly. Consult STMicroelectronics MPU wiki at wiki.st.com/stm32mpu in the *Boot related switches* section of the board being used. For the STM32MP157C-EV1 Evaluation board, related information is provided in the *Boot options* section of the user manual (UM2535).
- The Cortex®-A7 goes into a loop and the Cortex®-M4 is debugged as a regular STM32 device, where the application is loaded using the debugger connection.

3.2 Target status

In the production mode, a status light in the bottom right of the STM32CubeIDE window provides information regarding the current status of the connection between the computer and the embedded Linux® system.

Note: The serial console is a shared resource and the target widget status has the lowest priority. If there is an active console view for the serial port, this prevents further target status updates until the serial port is disconnected from the view.

The various values of the target status light are presented in Table 1.

Table 1. Target status light

Status light	Icon	Description
Black		The light completely off means that the widget is disabled.
Red		STM32CubeIDE cannot establish contact and cannot detect any target.
Yellow		Indicates a dysfunction such as: 1. No network connection between the computer and the MPU. 2. The consoled is opened.
Green		The connection is up and running.

3.3 Serial console

To open the serial console, click on this icon:  .

At any given moment, there can be only one serial connection active to a single target. Using the *Remote System Explorer* perspective, a second connection to the target can be made over SSH. When the serial console is closed while there is an application still running within the serial console, the application continues to run. It is still running if a new serial console is later opened. In this case, the application that is still running causes interference with the *Target Widget Status* when it needs to refresh the IP address of the target.

Revision history

Table 2. **Document revision history**

Date	Version	Changes
29-Oct-2019	1	Initial release.

Contents

List of tables

List of figures