UM2285
User manual

Development guidelines for STM32Cube Expansion Packages

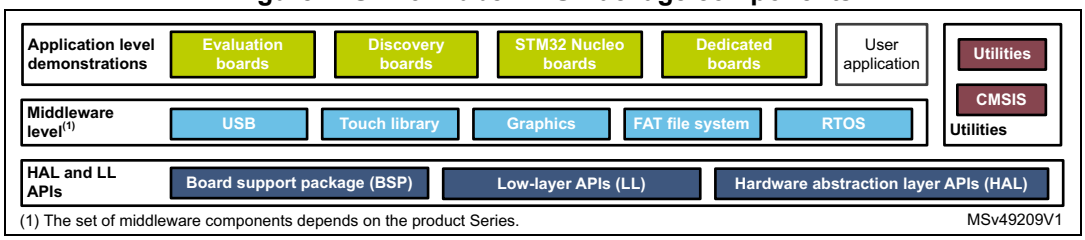
Introduction

STMCube™ is an STMicroelectronics original initiative to make developers' lives easier by reducing development effort, time and cost. STM32Cube covers the whole STM32 portfolio.

STM32Cube includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive STM32Cube MCU Package, delivered per STM32 microcontroller series (such as STM32CubeF4 for STM32F4 series), with:
  - The STM32Cube HAL, an STM32 abstraction-layer embedded software ensuring maximized portability across STM32 portfolio. The HAL is available for all peripherals
  - The low-layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. The LL APIs are available only for a set of peripherals.
  - A consistent set of middleware components such as RTOS, USB, TCP/IP and graphics
  - All embedded software utilities, delivered with a full set of examples

Figure 1. STM32Cube MCU Package components



Additionally, STM32Cube Expansion Packages contain embedded software components that complement the functionalities of STM32Cube MCU Packages, or enable the usage of a multitude of ST devices in various applicative domains together with the most appropriate STM32 microcontrollers, or both.

This document describes the compatibility requirements for STM32Cube Expansion Package development that ensure a proper match with STM32Cube MCU Package and tools, and overall consistency within STM32Cube ecosystem, enabling rapid application development based on proven and validated software elements.

Readers of this document must be familiar with STM32Cube architecture, HAL and LL APIs, and programming model. A complete documentation set is available in the STM32Cube MCU Packages page on *www.st.com*.

Contents

List of tables

List of figures

1 General information

The STM32Cube MCU Package and STM32Cube Expansion Package run on STM32 32-bit microcontrollers based on the Arm® Cortex®-M processor.

arm

2 References and acronyms

The following document available on *www.st.com* is concurrently used for the development of STM32Cube Expansion Packages:

1. *Development checklist for STM32Cube Expansion Packages* (UM2312)

Table 1 presents the definition of acronyms that are relevant for a better understanding of this document.

Table 1. List of acronyms

Term	Definition
API	Application programming interface
BSP	Board support package
CMSIS	Cortex® microcontroller system interface standard
DHCP	Dynamic host configuration protocol
FTP	File transfer protocol
HAL	Hardware abstraction layer
HTTP	Hypertext transfer protocol
LL	Low-layer
TCP/IP	Transmission control protocol / Internet protocol
TLS/SSL	Transport layer security / secure sockets layer

3 STM32Cube MCU Package and STM32Cube Expansion Package

The STM32Cube solution is composed of the STM32CubeMX which is the tools part, and the STM32Cube MCU Package providing the software bricks needed to benefit from STM32 microcontroller features.

Additionally to the STM32CubeMX and STM32Cube MCU Package, the STM32Cube Expansion Packages enrich the overall STM32Cube ecosystem with complementary add-ons.

3.1 STM32Cube MCU Package

The STM32Cube MCU Package (such as STM32CubeF4 for the microcontrollers in the STM32F4 Series), provides all the necessary software bricks needed to use STM32 microcontroller hardware features.

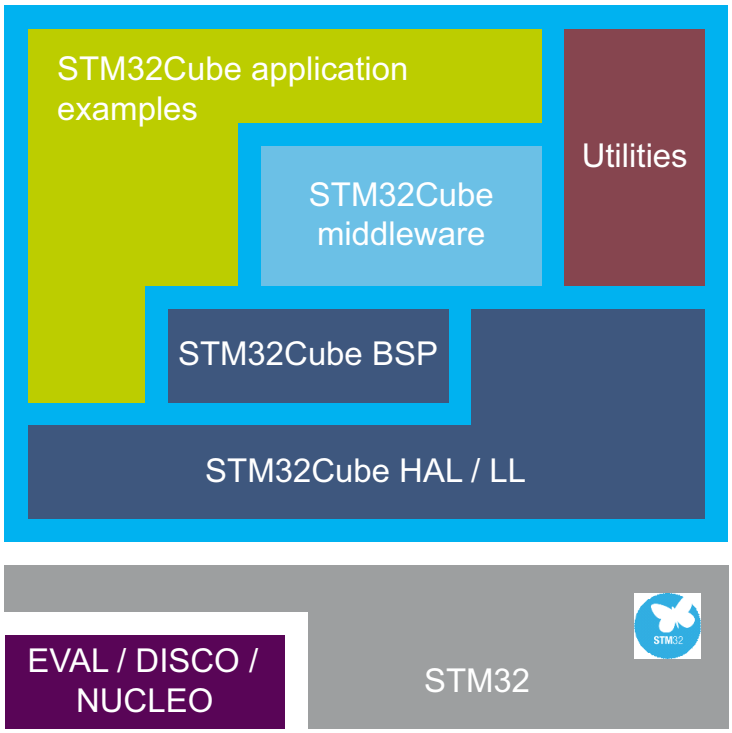
The STM32Cube MCU Package contains mainly:

- STM32 peripheral drivers
  - HAL (hardware abstraction layer)
  - LL (low-level API)
- Middleware covering STM32 peripheral set
  - RTOS, TCP/IP, TLS/SSL, USB, Graphics, File System, JPEG and others
- External components drivers (BSP) for STM32 boards
  - Evaluation boards
  - Discovery kits
  - Nucleo boards
- Rich set of examples aiming at demonstrating STM32 hardware and associated embedded software features and usage

The STM32Cube embedded software is distributed under the *Mix Liberty + OSS + 3rd party V1* mixed-licensed model as described on *www.st.com*.

Figure 2 presents the top-level structure of the STM32Cube MCU Package as distributed, maintained and supported by STMicroelectronics.

Figure 2. STM32Cube MCU Package content



3.2 STM32Cube Expansion Package

STM32Cube Expansion Package contains add-on software components that complement the functionalities of the STM32Cube MCU Package for:

- New middleware stack
- New hardware and board support (BSP)
- New examples
- Several of the items above

There are two categories of Expansion Packages: some STM32Cube Expansion Packages are developed, maintained and supported by STMicroelectronics, while some others are developed, maintained and distributed by third parties not affiliated to STMicroelectronics.

4 Packaging requirements

The STM32Cube MCU Package is the backbone of any STM32Cube Expansion Package. As a result, the folders and file structures shall always be organized without modifying the original folder structures as shown in *Figure 3 on page 9*.

The detailed requirements of the STM32Cube Expansion Package content are available within the *STM32Cube Expansion Package development checklist* document [*1*].

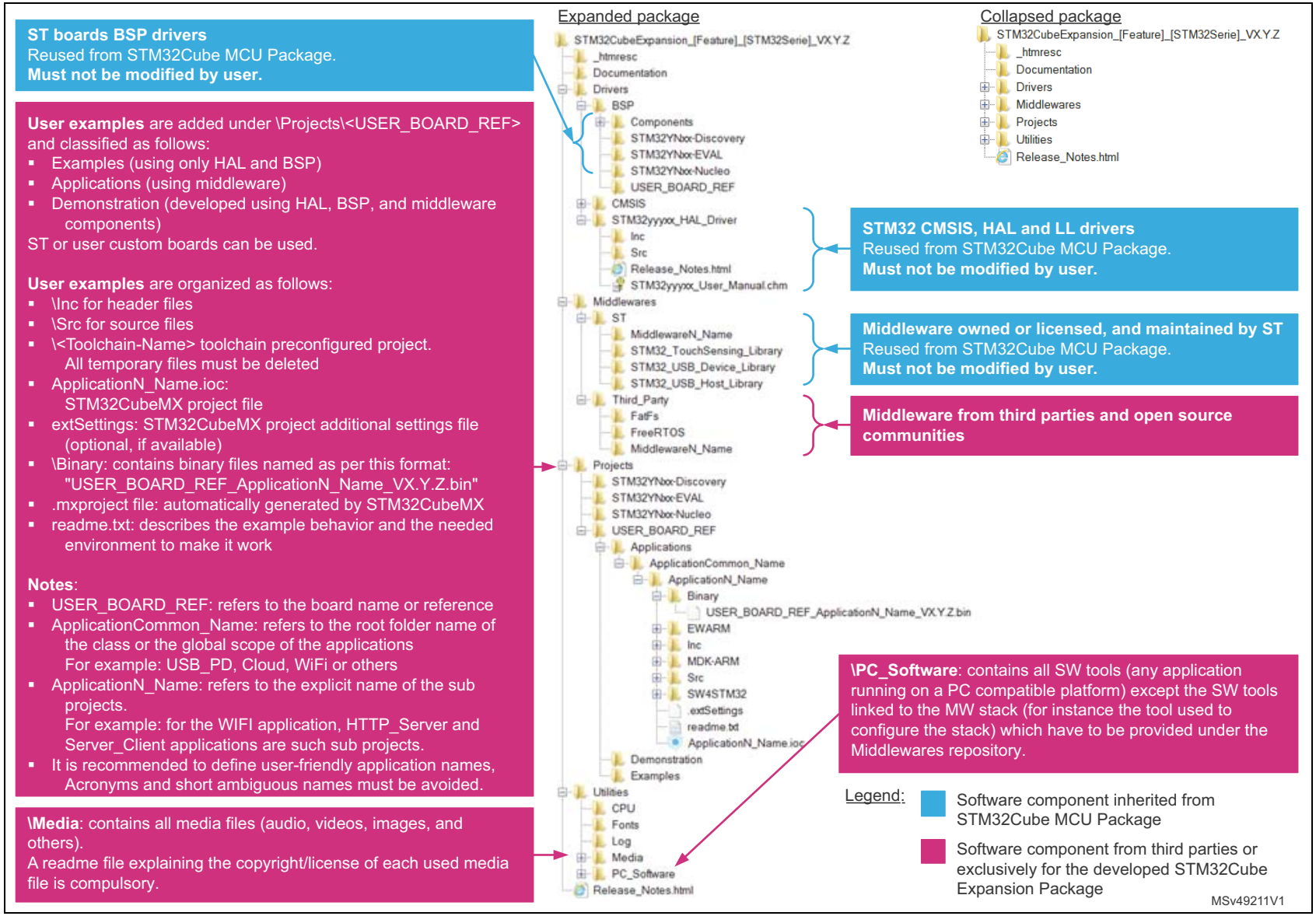
4.1 Example development with STM32CubeMX

In an STM32Cube Expansion Package, examples shall be developed using the STM32CubeMX tool. This requirement implies that the development satisfies to the following rules:

- The STM32CubeMX tool shall be used to configure the STM32 device and board, and generate the corresponding initialization code
- In the generated *.h and *.c source files, the user shall add the applicative code within the sections limited by the `/* USER CODE BEGIN */` and `/* USER CODE END */` markers
- The associated *.ioc file shall be available at the example root
- If additional STM32CubeMX project settings are used, the .extSettings file shall be kept at the same level as the *.ioc file

Location and naming convention of the *.ioc are described in *Figure 3*.

Figure 3. STM32Cube Expansion Package architecture and content



4.2 Extension of STM32Cube MCU Package drivers and middleware

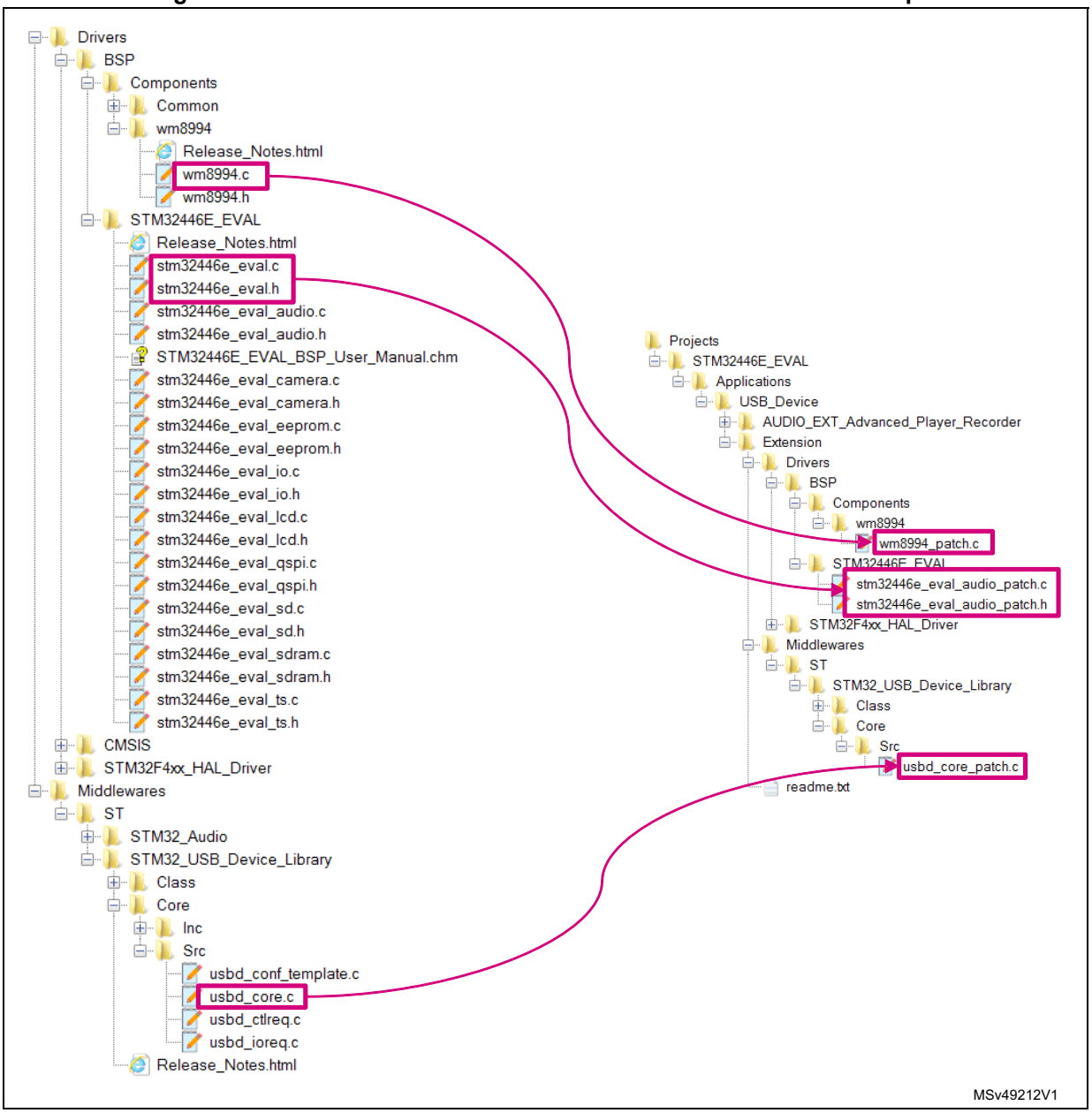
In some particular cases, it may happen that native drivers (HAL/LL, CMSIS, and BSP) or middleware provided within the STM32Cube MCU Package, or both, need to be updated by the developer of the STM32Cube Expansion Package (for instance for extending the implemented features, early fixing of a bug prior to the official ST release, or others).

In such a situation, the user shall proceed as follows:

1. Under the \ApplicationN_Name repository, create a folder \Extension into which the files to be patched shall be copied.
   This copy operation must respect the folder hierarchy. The number of sub-folders can anyhow be reduced if keeping the default folder hierarchy leads to a long path issue.
2. Rename the files to be patched by appending the *_patch* postfix.
3. Use patched files in the project instead of the original ones.

An example is provided in *Figure 4*, where files outlined in red are customized to implement specific features needed for an USB Device audio streaming application that are not natively be supported by the STM32Cube MCU Package.

Figure 4. STM32Cube MCU drivers and middleware extension example



5 New middleware integration

5.1 Requirements

Middleware components are firmware layers that lie between the STM32 hardware and the user application. Basically, any new middleware component must comply to the following requirements:

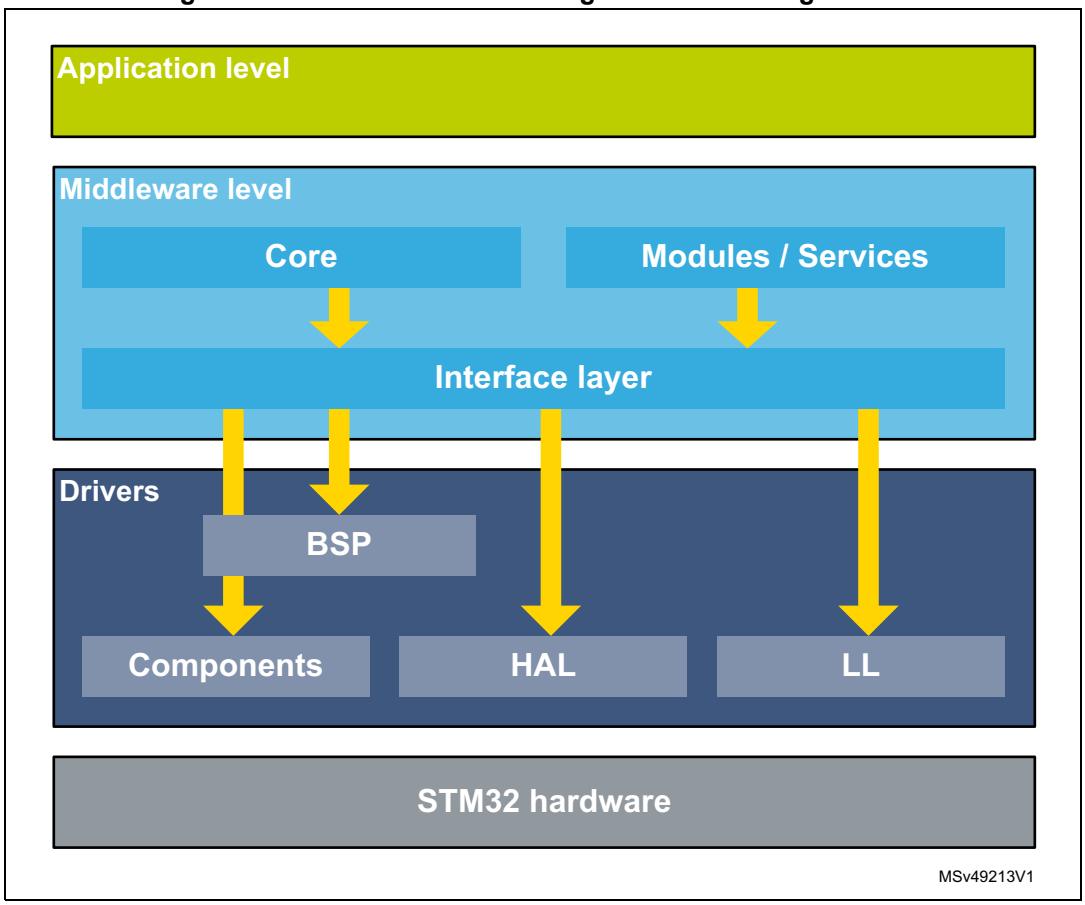
- Must implement a modular architecture, breaking down complexity into simpler chunks or files.
- Must be hardware (device and board) independent.
- The connection with hardware drivers and other middleware stacks shall be done through an interface file.
- The interface file shall be provided as template within the middleware folder to be customized by the user.

5.2 Organization

A middleware component is mostly composed of:

- **Modules**: a module is a common-functionality sub-layer which can be added or removed individually. For example the TCP/IP stack is composed of the TCP/IP core and of the DHCP, HTTP, and FTP components. Each component is considered as a module. It can be added or removed by a specific define/macro in the configuration file which enables or disables it.
- **Core**: this is the kernel of a component; it manages the main library state machine and manages the data flow between the various modules.
- **Interface layer**: the interface layer is generally used to link the middleware core component with lower layers like the HAL, LL, and BSP drivers.

Figure 5. STM32Cube MCU Package middleware organization



Depending on the middleware component, the interface layer can belong to one of these categories:

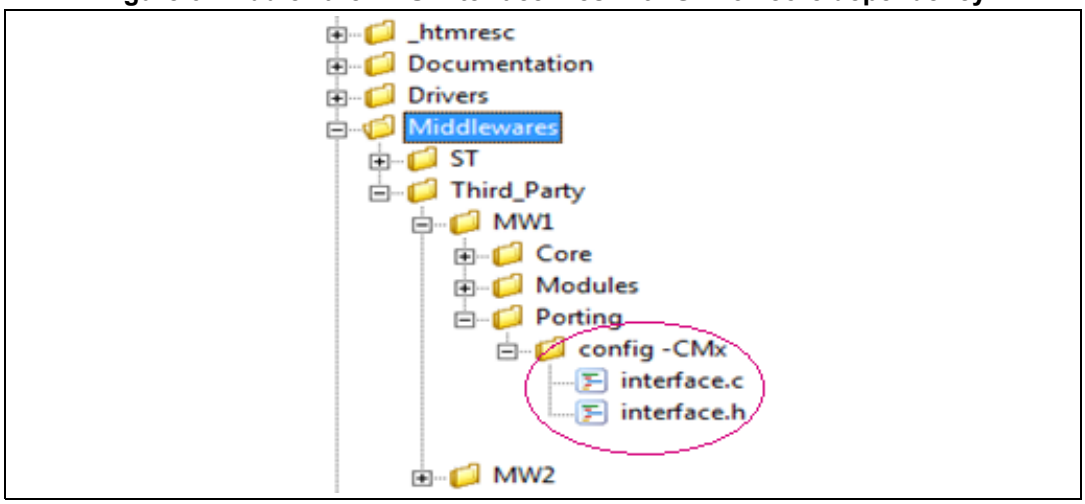
- Interface with the CPU depending only on the STM32 core used
- Interface with the HAL, LL, and BSP drivers

Interface with CPU depending only on the STM32 core used: in this case, the interface files shall be located in the Middlewares directory and used by all the applications working with the same core.

FreeRTOS™ can be considered as example as it provides porting files for each Arm® Core (such as Cortex®-M3, Cortex®-M4, or others) and for each compiler, since some assembler pieces of code are needed to handle context switching and registers save and restore mechanism.

The location of interface files in the middleware structure is illustrated in *Figure 6*.

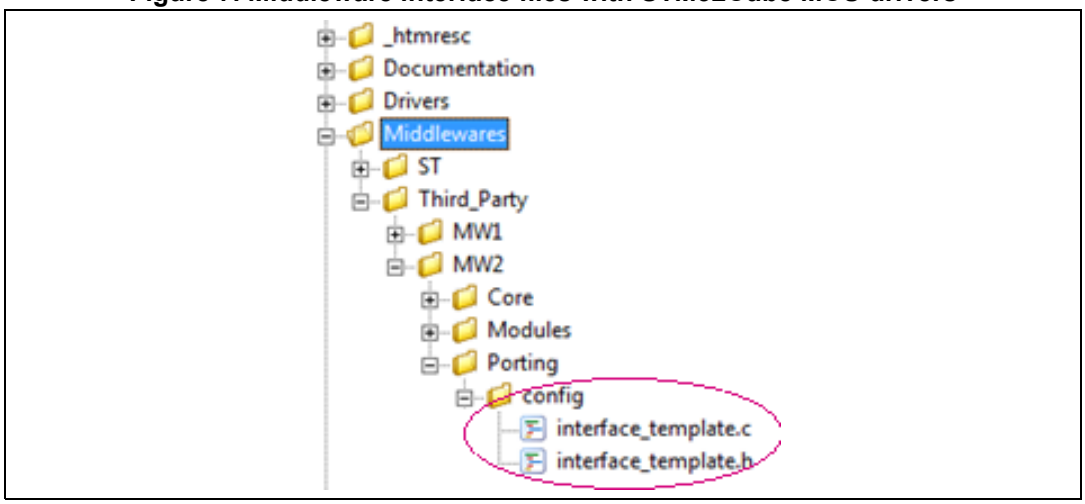
Figure 6. Middleware CPU interface files with STM32 core dependency



Interface with HAL, LL, and BSP drivers: in some middleware components, the interface is a set of empty or almost empty functions that need to be filled by the user to link it with the HAL and LL layer. Generally, the interface files are given as template files together with the middleware components. They must be copied and customized in the application layer.

The location of interface files in the middleware structure is illustrated in *Figure 7*.

Figure 7. Middleware interface files with STM32Cube MCU drivers



Each sub folder must contain a \Inc folder and a \Src folder where the header file (*.h) and the source file (*.c) are respectively located.

If the middleware component is quite simple; sub folders \Core, \Modules, and \Porting can be dropped; only the \Inc and \Src sub-folders shall then be provided at the middleware component root level.

5.3 Delivery in object format

When the middleware library is delivered in binary or object format, it must comply to a minimum set of requirements:

- A header file shall be provided to export the library interface API to end applications
- A release note shall be made available
- The library shall be provided in object format for all supported compilers. In case the library object is compiler dependent, the supported compilers have to be clearly indicated in the object file name.

  As an example, ***LibraryNameV_CMx_C_O.a*** is a library object file name where:
  - ***V:*** module version (for instance V=01 for a V0.1 release)
  - ***x:*** the CMx core class (CM0, CM3, CM4, CM7, CM23, CM33)
  - ***C:*** compiler (IAR™, Keil®, GCC)
  - ***O:*** specify the compiler optimization
  - ***<empty>***: high size optimization
  - ***Ot***: high-speed optimization
  - ***Otnsc***: high-speed optimization with No Size constraints
  - ***Ob***: high-balanced optimization

6 Software quality requirements

BSP drivers, middleware and projects developed within the STM32Cube Expansion Package (add-ons with respect to STM32Cube MCU Package) have to meet the minimum set of requirements below:

- Ensure compilation with all supported compilers (EWARM, MDK-ARM and SW4STM32) on Windows® and Linux® platforms, without errors nor warnings (warnings are accepted only if present on SW components not owned by the developer of the Expansion Package).
- Functional tests are performed with no known bugs left (minor bugs are accepted provided they are documented in the component release note), with evidence reports.

BSP drivers and middleware must comply to these additional requirements:

- MISRA C® compliance and static code analysis, with evidence reports
- MISRA C® non-compliance rules deviation shall be justified.

7 Revision history

Table 2. Document revision history

Date	Revision	Changes
14-Nov-2017	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

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

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

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

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

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

© 2017 STMicroelectronics – All rights reserved