

# DMA internal peripheral

Stable: 11.02.2019 - 12:21 / Revision: 18.01.2019 - 17:05

## Contents

1 Article purpose .....	1
2 Peripheral overview .....	1
2.1 Features .....	2
2.2 Security support .....	2
3 Peripheral usage and associated software .....	2
3.1 Boot time .....	2
3.2 Runtime .....	2
3.2.1 Overview .....	2
3.2.2 Software frameworks .....	2
3.2.3 Peripheral configuration .....	3
3.2.4 Peripheral assignment .....	3
4 References .....	3

## 1 Article purpose

The purpose of this article is to:

- briefly introduce the DMA peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the DMA peripheral.

## 2 Peripheral overview

The **DMA** peripheral is used to perform direct accesses from/to a device or a memory. Each DMA instance supports 8 channels. The selection of the device connected to each DMA channel and controlling the DMA transfers is done via the [DMAMUX](#).

Note: Directly accessing [DDR](#) from the DMA is not recommended for high-bandwidth or latency-critical transfers. This means that DMA transfers configured by the Arm<sup>®</sup> Cortex<sup>®</sup>-A7 operating system, that usually target buffers in [MCU SRAM](#), require a hardware mechanism to chain the DMA and a [MDMA](#) channel in order to achieve the following flow:

DDR<-> MDMA <-> MCU SRAM <-> DMA <-> device

This feature was already present on STM32H7 microcontroller Series. It is documented in application note AN5001<sup>[1]</sup>.

## 2.1 Features

Refer to the [STM32MP15 reference manuals](#) for the complete list of features, and to the software components, introduced below, to see which features are implemented.

## 2.2 Security support

The DMA is a **non-secure** peripheral.

## 3 Peripheral usage and associated software

### 3.1 Boot time

The DMA is not used at boot time.

### 3.2 Runtime

#### 3.2.1 Overview

DMA instances can be allocated to:

- the Arm<sup>®</sup> Cortex<sup>®</sup>-A7 non-secure core to be controlled in Linux<sup>®</sup> by the [dmaengine](#) framework

or

- the Arm<sup>®</sup> Cortex<sup>®</sup>-M4 to be controlled in STM32Cube MPU Package by the [DMA HAL driver](#)

#### 3.2.2 Software frameworks

Do mai n	Peri phe ral	Software frameworks			Comment
		Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4  (STM32Cube)	
Core /DM A	DMA		Linux dmaengine framework	STM32Cube DMA driver	

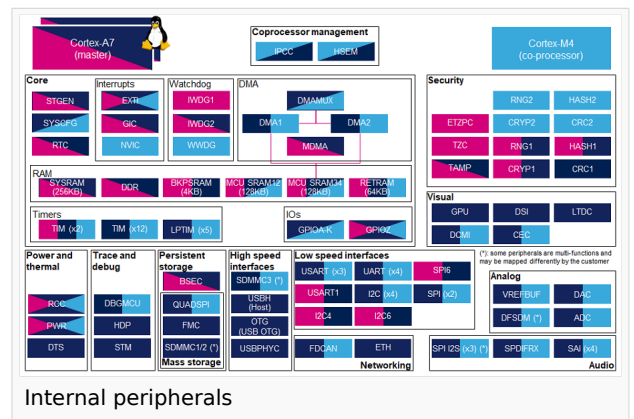
### 3.2.3 Peripheral configuration

The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the [STM32CubeMX](#) tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in the corresponding software framework article.

### 3.2.4 Peripheral assignment

**Check boxes** illustrate the possible peripheral allocations supported by [STM32 MPU Embedded Software](#):

- means that the peripheral can be assigned () to the given runtime context.
- is used for system peripherals that cannot be unchecked because they are statically connected in the device.



Refer to [How to assign an internal peripheral to a runtime context](#) for more information on how to assign peripherals manually or via [STM32CubeMX](#).

The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possibilities might be described in [STM32MP15 reference manuals](#).

Do ma in	Pe ri ph e ra l	Runtime allocation			Comme nt
		Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	
Cor e / D MA	DM A	DMA1		<input type="checkbox"/>	Assignm ent (single choice)
		DMA2		<input type="checkbox"/>	Assignm ent (single choice)

## 4 References

- [http://www.st.com/resource/en/application\\_note/dm00360392.pdf](http://www.st.com/resource/en/application_note/dm00360392.pdf)