



DMA internal peripheral



# DMA internal peripheral

Stable: 11.02.2019 - 11:21 / Revision: 18.01.2019 - 16:05

Template:ArticleMainWriter Template:ArticleApprovedVersion

## Contents

1 Article purpose .....	2
2 Peripheral overview .....	2
<b>2.1 Features</b> .....	<b>3</b>
<b>2.2 Security support</b> .....	<b>3</b>
3 Peripheral usage and associated software .....	3
<b>3.1 Boot time</b> .....	<b>3</b>
<b>3.2 Runtime</b> .....	<b>3</b>
3.2.1 Overview .....	3
3.2.2 Software frameworks .....	3
3.2.3 Peripheral configuration .....	4
3.2.4 Peripheral assignment .....	4
4 References .....	5

## 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	Peri	Software frameworks			Comment
mai Co r t e x - A 7 s e c u r e (O P- T E E)	Cor t e x - A 7 n o n- s e c u r e (L i n u x )	Cortex-M4  (STM32Cube)			
	Co				

Do	Peri	Software frameworks		Comment
MA	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.

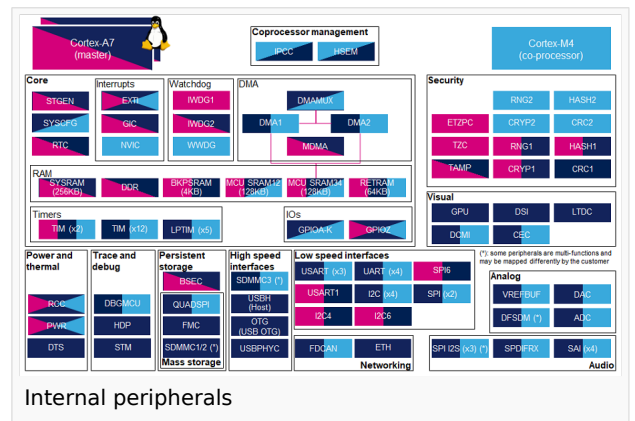
### 3.2.4 Peripheral assignment

**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	Peri	Runtime allocation		Comment
MA	DMA	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	Assign



Do ma in e/ D M A	Per iph era D M A	Runtime allocation				Comme nt
		DMA1				Assignment (single choice )
		DMA2				Assignment (single choice )

## 4 References

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

Direct Memory Access

Doubledata rate (memory domain)

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.)

Microprocessor Unit

Open Portable Trusted Execution Environment