# DCMI internal peripheral

Stable: 22.02.2020 - 10.42 / Revision: 22.02.2020 - 10.42

## Contents

# 1 Article purpose

The purpose of this article is to

- briefly introduce **DCMI** 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 needed, how to configure DCMI peripheral.

# 2 Peripheral overview

The **DCMI** (digital camera memory interface) is an STM32 internal peripheral allowing to receive some video data from an external parallel camera sensor device or any other digital video equipment supporting parallel interface.

The DCMI hardware block can receive raw data frames in RGB565 and YUV422 formats as well as JPEG compressed data.

## 2.1 Features

Refer to STM32MP15 reference manuals for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

Refer to STM32 DCMI presentation [1] for an overview of DCMI hardware block and its capabilities.

## 2.2 Security support

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

# 3 Peripheral usage and associated software

## 3.1 Boot time

The DCMI is **not used at boot time**.

## 3.2 Runtime

### 3.2.1 Overview

The DCMI internal peripheral can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with the V4L2 framework
- or to the Arm®Cortex® -M4 core to be used with STM32Cube MPU Package with DCMI HAL driver

Chapter #Peripheral assignment describes which peripheral instance can be assigned to which context.

### 3.2.2 Software frameworks

| Domain | Peripheral | Software components | | Comment |
|---|---|---|---|---|
| OP-TEE | Linux | STM32Cube | | |
| High-speed interface | DCMI | V4L2 framework | DCMI HAL driver | |

### 3.2.3 Peripheral configuration

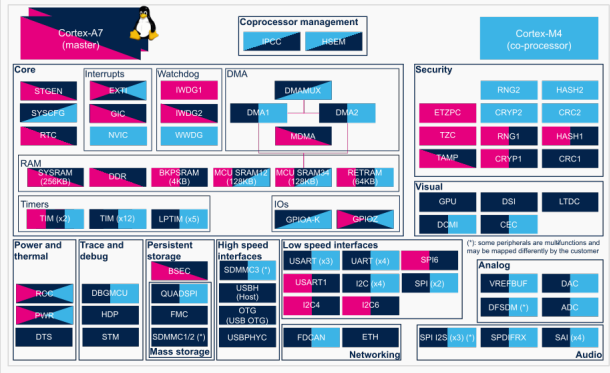
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in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

### 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

| Domain | Peripheral | Runtime allocation | | | | Comment |
|---|---|---|---|---|---|---|
| Instance | Cortex-A7 secure (OP-TEE) | Cortex-A7 non-secure (Linux) | Cortex-M4 (STM32Cube) | | | |
| Visual | DCMI | DCMI | | | | Assignment (single choice) |

# 4 How to go further

Refer to STM32 DCMI Application Note (AN5020)[2] for a detailed description of the DCMI peripheral and applicable use-cases.

This application note is related to STM32 microcontrollers but it is also applicable to STM32 MPUs. This document can help to better understand stm32-dcmi V4L2 kernel driver and debug camera sensor and DCMI interactions.

# 5 References

- STM32 DCMI presentation
- STM32 DCMI Application Note (AN5020)

Digital Camera Memory Interface



Microprocessor Unit

Open Portable Trusted Execution Environment

Video 4 Linux® version 2