

# DBGMCU internal peripheral

Stable: 26.09.2019 - 15:46 / Revision: 26.09.2019 - 15:46

## Contents

1 Article purpose .....	1
2 Peripheral overview .....	1
2.1 Features .....	1
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 .....	2
3.2.4 Peripheral assignment .....	2

## 1 Article purpose

The purpose of this article is to:

- briefly introduce the DBGMCU 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 DBGMCU peripheral.

## 2 Peripheral overview

The **DBGMCU** peripheral is used to configure internal peripherals behavior when one of the cores (Cortex<sup>®</sup>-A7 or Cortex<sup>®</sup>-M4) enters in debug mode.

For instance, it allows to freeze a watchdog (**IWDG**) to avoid getting a watchdog reset when the Cortex<sup>®</sup>-A7 enters in debug mode (via a breakpoint or JTAG break).

### 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 DBGMCU is a **non secure** peripheral.

## 3 Peripheral usage and associated software

### 3.1 Boot time

DBGMCU is partly initialized by the [FSBL](#) at boot time in order to freeze [IWDG2](#) when the Cortex<sup>®</sup>-A7 enters in debug mode.

### 3.2 Runtime

#### 3.2.1 Overview

There is no real runtime support for DBGMCU. Instead, users who need to freeze some peripherals for a debug session should complete the minimal initialization done in the [FSBL](#) with their needs.

#### 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)	
Trac e & Deb ug	<a href="#">DBG MCU</a>				

#### 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](#):

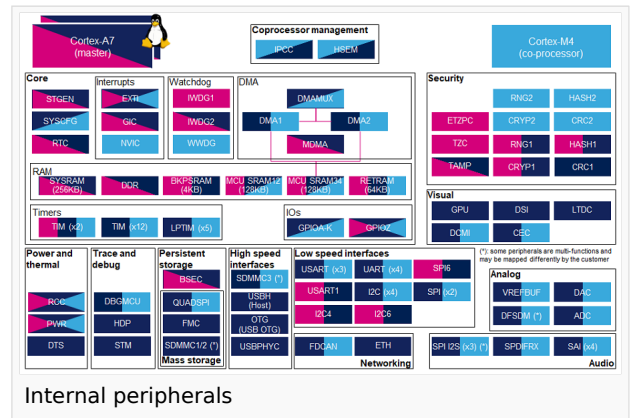
- means that the peripheral can be assigned () to the given runtime context.

- ✓ 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](#).



Domain	Peripheral	Runtime allocation			Comment
		Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	
Trace & Debug	DBGMCU				No assignment

debug and test protocol, named from the Joint Test Action Group that developed it

Open Portable Trusted Execution Environment