



NVIC internal peripheral



A quality version of this page, approved on *25 March 2021*, was based off this revision.

Contents

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



1 Article purpose

The purpose of this article is to:

- briefly introduce the NVIC and its main features
- indicate the level of security supported by this hardware block
- explain how the NVIC can be configured.



2 Peripheral overview

The **NVIC** is the Arm[®] Cortex[®]-M4 interrupt controller. As a result, it cannot be accessed by the Arm Cortex-A7 core.

2.1 Features

Refer to [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 NVIC is a **non-secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The NVIC can be configured through the STM32Cube.

3.2 Runtime

3.2.1 Overview

The NVIC can be allocated only to the Arm Cortex-M4 core to be controlled in the STM32Cube by the NVIC HAL driver.

3.2.2 Software frameworks

Domain	Peripheral	Software components	Comment
OP-TEE	Linux	STM32Cube	
Core /Interrupts	NVIC		STM32Cube NVIC 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.

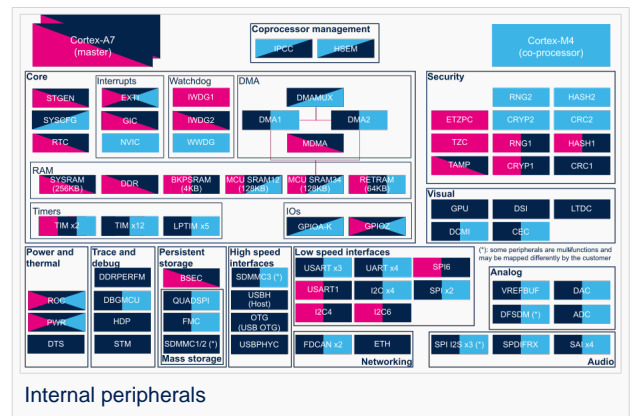
3.2.4 Peripheral assignment

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



Domain	Periphera	Runtime allocation	Comment
	Cortex-A7	Cortex-A7	
		Cortex-M4	
		(STM32Cub	



Domain	Periphera	Runtime allocation			Comment
Instance	secure (OP-TEE)	non-secure (Linux)	e)		
Core /Interrupts	NVIC	NVIC			