

# NVIC internal peripheral

Stable: 11.02.2019 - 13:21 / Revision: 18.01.2019 - 18:04

## 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
4 How to go further .....	3
5 References .....	3

## 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<sup>®</sup> Cortex<sup>®</sup>-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](#). Refer to the [STM32MP15 interrupts](#) article for more information on the interrupt configuration strategy.

### 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

Do mai n	Peri phe ral	Software frameworks			Comment
		Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4  (STM32Cube)	
Core /Inte rrup ts	NVI C			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](#):

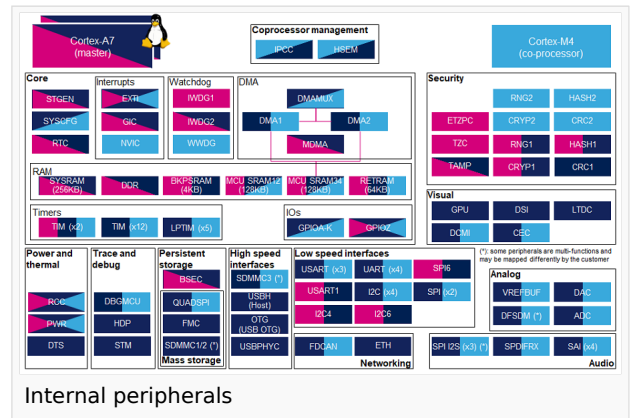
- means that the peripheral can be assigned () to the given runtime context.

- ✓ 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	Peripheral	Runtime allocation			Comment
		Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	
Core /Interrupts	NVIC	NVIC			✓

## 4 How to go further

Victor P. Nelson's training <sup>[1]</sup> provides detailed information on the NVIC behavior and implementation in STM32F4 microcontrollers, that can easily be transposed to the STM32MP15 Cortex-M4 based coprocessor.

## 5 References

- [↑ http://www.eng.auburn.edu/~nelson/courses/elec5260\\_6260/slides/ARM%20STM32F407%20Interrupts.pdf](http://www.eng.auburn.edu/~nelson/courses/elec5260_6260/slides/ARM%20STM32F407%20Interrupts.pdf) ARM and STM32F4xx Operating Modes & Interrupt Handling

Nested Vectored Interrupt Controller

Open Portable Trusted Execution Environment