



NVIC internal peripheral



Contents

1. NVIC internal peripheral	3
2. How to assign an internal peripheral to a runtime context	9
3. STM32CubeMP1 architecture	15
4. STM32CubeMX	21
5. STM32MP15 interrupts	27
6. STM32MP15 resources	33
7. STM32MPU Embedded Software architecture overview	39



A quality version of this page, approved on 4 February 2020, was based off this revision.

Contents

1 Article purpose	4
2 Peripheral overview	5
2.1 Features	5
2.2 Security support	5
3 Peripheral usage and associated software	6
3.1 Boot time	6
3.2 Runtime	6
3.2.1 Overview	6
3.2.2 Software frameworks	6
3.2.3 Peripheral configuration	6
3.2.4 Peripheral assignment	6
4 How to go further	8
5 References	9



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. 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

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.

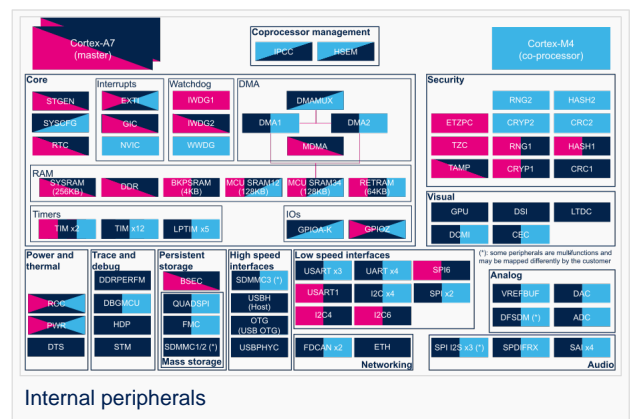
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further

Victor P. Nelson's training ^[1] 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 *ARM and STM32F4xx Operating Modes & Interrupt Handling*

Nested Vectored Interrupt Controller

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex®

Open Portable Trusted Execution Environment

Linux® is a registered trademark of Linus Torvalds.

Stable: 08.03.2021 - 16.13 / Revision: 16.02.2021 - 17.11

Contents

1 Article purpose	10
2 Peripheral overview	11
2.1 Features	11
2.2 Security support	11
3 Peripheral usage and associated software	12
3.1 Boot time	12
3.2 Runtime	12
3.2.1 Overview	12
3.2.2 Software frameworks	12
3.2.3 Peripheral configuration	12
3.2.4 Peripheral assignment	12
4 How to go further	14
5 References	15



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. 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

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.

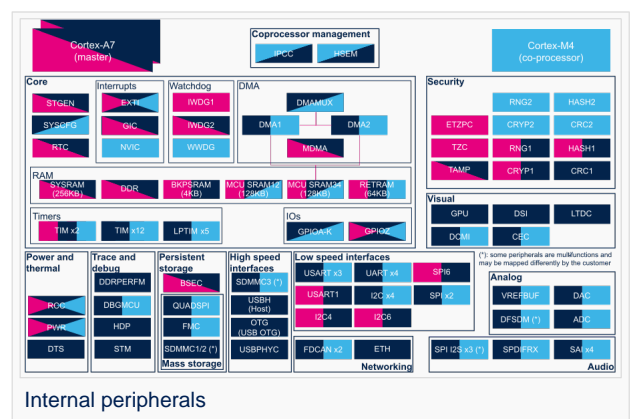
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further

Victor P. Nelson's training ^[1] 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 *ARM and STM32F4xx Operating Modes & Interrupt Handling*

Nested Vectored Interrupt Controller

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex®

Open Portable Trusted Execution Environment

Linux® is a registered trademark of Linus Torvalds.

Stable: 31.03.2021 - 11:58 / Revision: 23.03.2021 - 14:07

Contents

1 Article purpose	16
2 Peripheral overview	17
2.1 Features	17
2.2 Security support	17
3 Peripheral usage and associated software	18
3.1 Boot time	18
3.2 Runtime	18
3.2.1 Overview	18
3.2.2 Software frameworks	18
3.2.3 Peripheral configuration	18
3.2.4 Peripheral assignment	18
4 How to go further	20
5 References	21



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. 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

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.

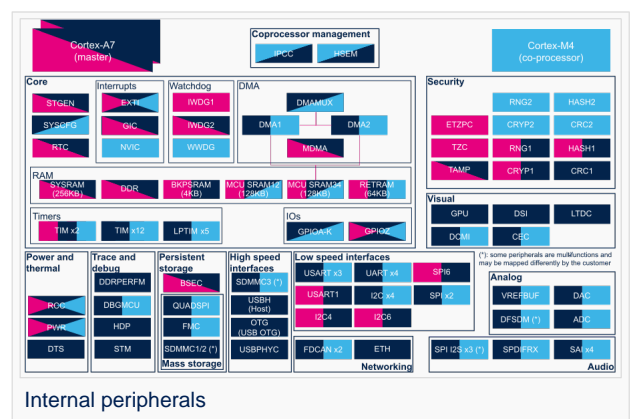
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further

Victor P. Nelson's training ^[1] 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 ARM and STM32F4xx Operating Modes & Interrupt Handling

Nested Vectored Interrupt Controller

Arm[®] is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex[®]

Open Portable Trusted Execution Environment

Linux[®] is a registered trademark of Linus Torvalds.

Stable: 23.09.2020 - 13:22 / Revision: 12.06.2020 - 13:25

Contents

1 Article purpose	22
2 Peripheral overview	23
2.1 Features	23
2.2 Security support	23
3 Peripheral usage and associated software	24
3.1 Boot time	24
3.2 Runtime	24
3.2.1 Overview	24
3.2.2 Software frameworks	24
3.2.3 Peripheral configuration	24
3.2.4 Peripheral assignment	24
4 How to go further	26
5 References	27



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. 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

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.

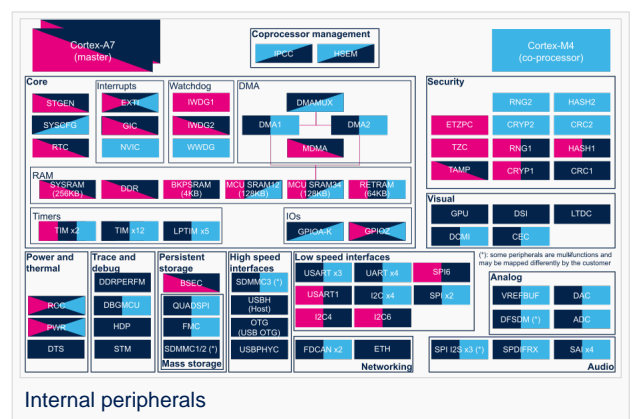
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further

Victor P. Nelson's training ^[1] 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 ARM and STM32F4xx Operating Modes & Interrupt Handling

Nested Vectored Interrupt Controller

Arm[®] is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex[®]

Open Portable Trusted Execution Environment

Linux[®] is a registered trademark of Linus Torvalds.

Stable: **Not stable** / Revision: 08.01.2021 - 14:34

Contents

1 Article purpose	28
2 Peripheral overview	29
2.1 Features	29
2.2 Security support	29
3 Peripheral usage and associated software	30
3.1 Boot time	30
3.2 Runtime	30
3.2.1 Overview	30
3.2.2 Software frameworks	30
3.2.3 Peripheral configuration	30
3.2.4 Peripheral assignment	30
4 How to go further	32
5 References	33



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. 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

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.

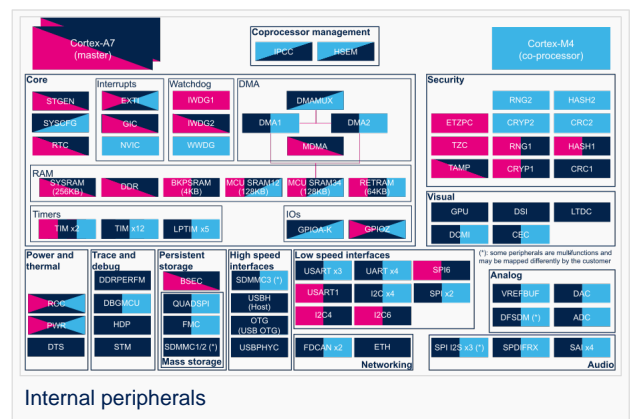
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further

Victor P. Nelson's training ^[1] 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 *ARM and STM32F4xx Operating Modes & Interrupt Handling*

Nested Vectored Interrupt Controller

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex®

Open Portable Trusted Execution Environment

Linux® is a registered trademark of Linus Torvalds.

Stable: 20.07.2021 - 09:02 / Revision: 11.03.2021 - 08:07

Contents

1 Article purpose	34
2 Peripheral overview	35
2.1 Features	35
2.2 Security support	35
3 Peripheral usage and associated software	36
3.1 Boot time	36
3.2 Runtime	36
3.2.1 Overview	36
3.2.2 Software frameworks	36
3.2.3 Peripheral configuration	36
3.2.4 Peripheral assignment	36
4 How to go further	38
5 References	39



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. 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

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.

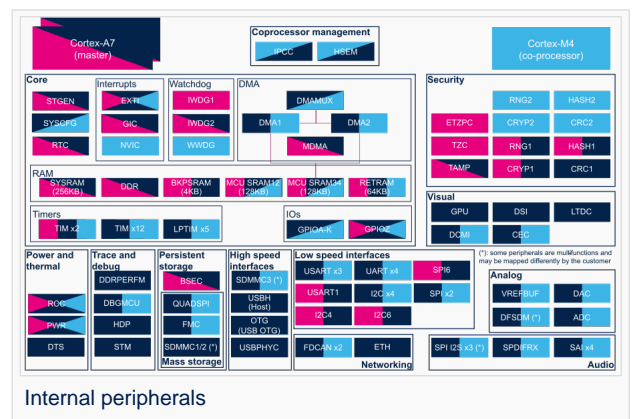
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further

Victor P. Nelson's training ^[1] 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 *ARM and STM32F4xx Operating Modes & Interrupt Handling*

Nested Vectored Interrupt Controller

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.



Cortex®

Open Portable Trusted Execution Environment

Linux® is a registered trademark of Linus Torvalds.

Stable: 26.03.2021 - 11:32 / Revision: 12.03.2021 - 11:07

Contents

1 Article purpose	40
2 Peripheral overview	41
2.1 Features	41
2.2 Security support	41
3 Peripheral usage and associated software	42
3.1 Boot time	42
3.2 Runtime	42
3.2.1 Overview	42
3.2.2 Software frameworks	42
3.2.3 Peripheral configuration	42
3.2.4 Peripheral assignment	42
4 How to go further	44
5 References	45



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. 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

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.

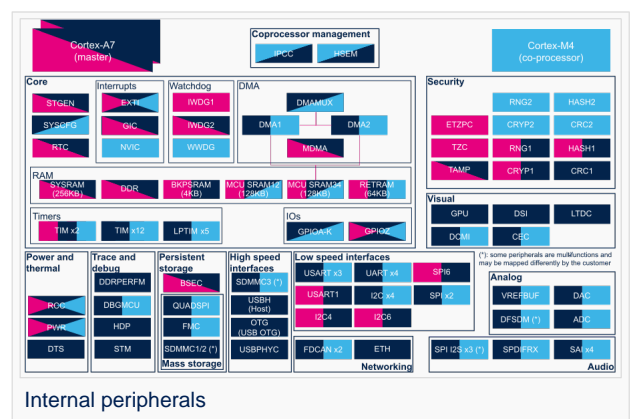
3.2.4 Peripheral assignment

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	Peripheral	Runtime allocation	Comment
	Cortex-A7 secure	Cortex-A7 non-secure	Cortex-M4



Domain	Peripheral	Runtime allocation			Comment
Instance	(OP-TEE)	(Linux)	(STM32Cube)		
Core /Interrupts	NVIC	NVIC			



4 How to go further


Victor P. Nelson's training ^[1] 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 *ARM and STM32F4xx Operating Modes & Interrupt Handling*

Nested Vectored Interrupt Controller

Arm[®] is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere. 

Cortex[®]

Open Portable Trusted Execution Environment

Linux[®] is a registered trademark of Linus Torvalds.