NVIC internal peripheral

Contents

NVIC internal peripheral

Stable: 11.02.2019 - 11:21 / Revision: 18.01.2019 - 16:04

A quality version of this page, accepted on *11 February 2019*, was based off this revision.

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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

<table>
<tr><th>Do</th><th>Peri</th><th colspan="3">Software frameworks</th><th>Comment</th></tr>
<tr><td>main Cortex-A7 secure (OP-TEE)</td><td>Cortex-A7 non-secure (Linux)</td><td colspan="3">Cortex-M4

(STM32Cube)</td><td></td></tr>
<tr><td>Core/Interrupts</td><td>NVIC</td><td></td><td></td><td>STM32Cube NVIC driver</td><td></td></tr>
</table>

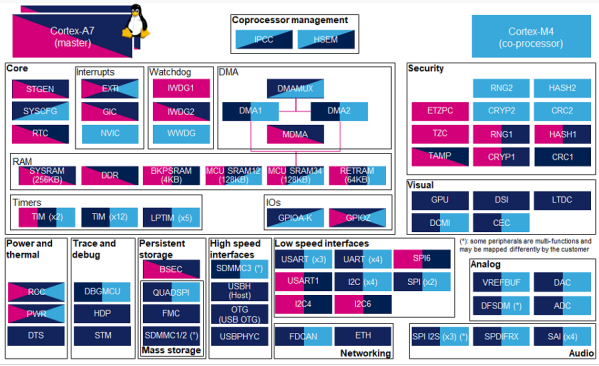
3.2.3 Peripheral configuration

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.



Internal peripherals

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.

Domain	Peripheral	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (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

Open Portable Trusted Execution Environment

NVIC internal peripheral

Stable: ***Not stable*** */ Revision: 11.05.2020 - 10:13*

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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	Peri	Software frameworks	Comment
Cortex-A7	Cortex-A7 no	Cortex-M4	

Do	Peri	Software frameworks			Comment
secure (OP-TEE)	non-secure (Linux)	(STM32Cube)			
Core/Interrupts	NVIC			STM32Cube NVIC driver	

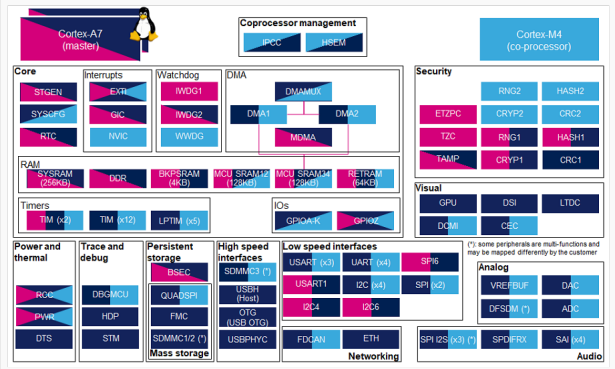
3.2.3 Peripheral configuration

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.



Internal peripherals

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.

Do	Per	Runtime allocation		Comment
main	ipheral	Cortex-A7	Cortex-M4	
Inst	Cortex-A7 secu			

Domain	Peripheral	Runtime allocation				Comment
	(OP-TEE)	non-secure (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

Open Portable Trusted Execution Environment

NVIC internal peripheral

Stable: 21.02.2020 - 08:39 / Revision: 04.02.2020 - 15:22

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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

Do	Peri	Software frameworks			Comment
main Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/Interrupts	NVIC			STM32Cube NVIC driver	

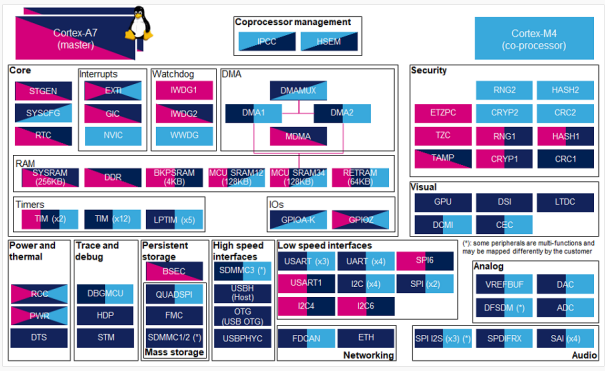
3.2.3 Peripheral configuration

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.



Internal peripherals

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.

Domain	Peripheral	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (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

Open Portable Trusted Execution Environment

NVIC internal peripheral

Stable: 04.02.2020 - 07:47 / Revision: 04.02.2020 - 07:38

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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 frameworks			Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/Interrupts	NVIC			STM32Cube NVIC driver	

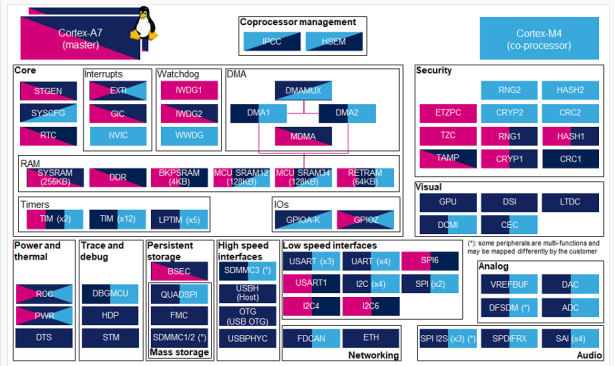
3.2.3 Peripheral configuration

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.



Internal peripherals

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.

Domain	Peripheral	Runtime allocation	Comment

Domain instance	Peripheral	Runtime allocation				Comment
	Text-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (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

Open Portable Trusted Execution Environment

NVIC internal peripheral

Stable: 31.01.2020 - 13:04 / Revision: 31.01.2020 - 13:02

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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

Do	Peri	Software frameworks			Comment
mai n Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (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.

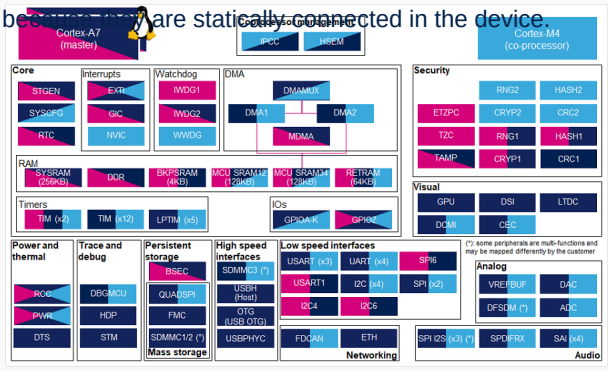
3.2.4 Peripheral assignment

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 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)			
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

Open Portable Trusted Execution Environment

NVIC internal peripheral

Stable: 15.10.2019 - 11:55 / Revision: 15.10.2019 - 11:55

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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	Peri	Software frameworks	Comment
Cortex-A7	Cortex-A7 no	Cortex-M4	

Do	Peri	Software frameworks			Comment
sec ure (O P-TE E)	n-sec ure (Li nux)	(STM32Cube)			
Co re /In ter ru pts	N VI C			STM32Cube NVIC driver	

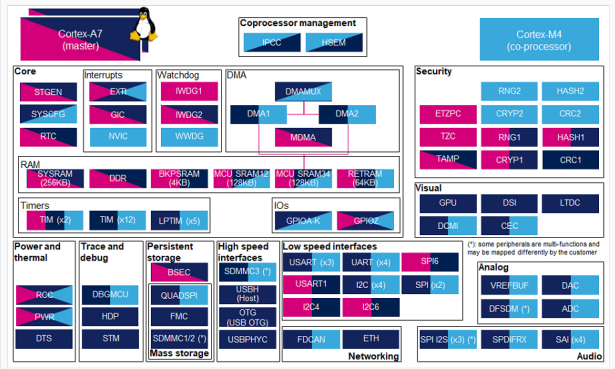
3.2.3 Peripheral configuration

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.



Internal peripherals

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.

Do ma in	Per iph era l	Runtime allocation		Comme nt
In st	Cor te x-A 7 se cu	Cortex-A7	Cortex-M4	

Domain	Peripheral	Runtime allocation				Comment
	(OP-TEE)	non-secure (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

Open Portable Trusted Execution Environment

NVIC internal peripheral

*Stable: **Not stable** / Revision: 30.01.2020 - 08:45*

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

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

Do	Peri	Software frameworks			Comment
main Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/Interrupts	NVIC			STM32Cube NVIC driver	

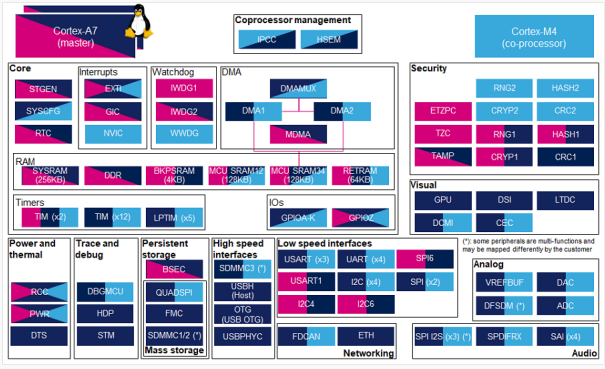
3.2.3 Peripheral configuration

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.



Internal peripherals

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.

Domain	Peripheral	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (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

Open Portable Trusted Execution Environment