NVIC internal peripheral

Contents

Stable: 25.03.2021 - 13:50 / Revision: 18.03.2021 - 17:29

Stable: 20.03.2021 - 10:09 / Revision: 18.03.2021 - 17:20

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

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

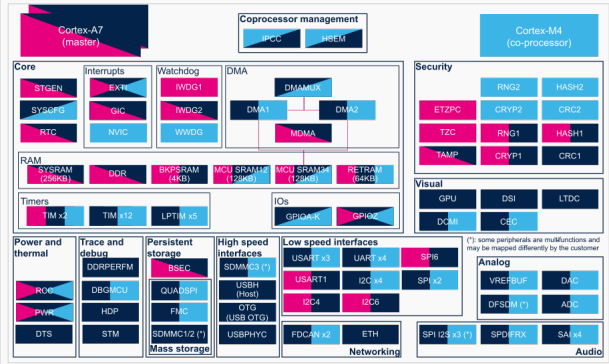
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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

Stable: 08.03.2021 - 16:13 / Revision: 16.02.2021 - 17:11

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

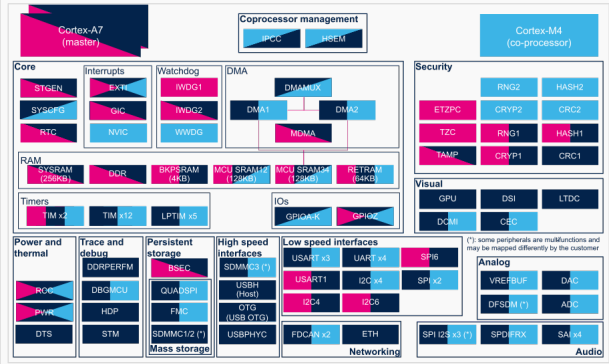
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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

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

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

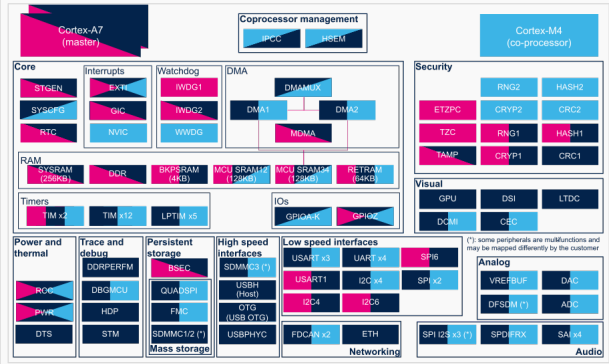
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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

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

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

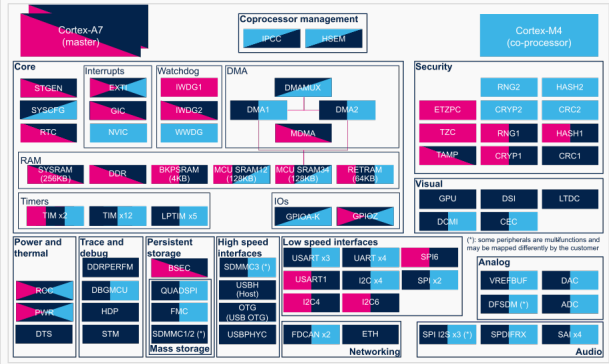
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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

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

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

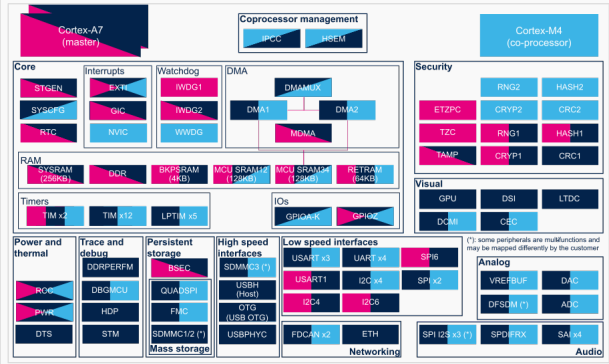
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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

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

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

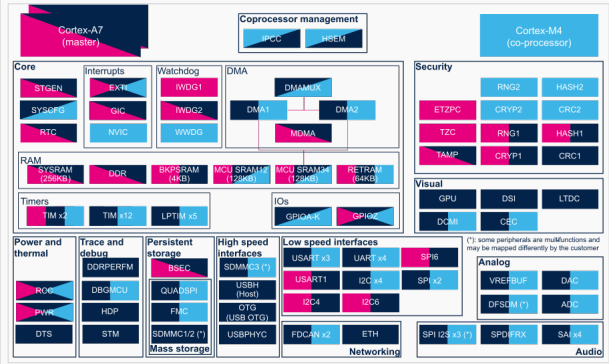
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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

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

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 components		Comment
OP-TEE	Linux	STM32Cube		
Core /Interrupts	NVIC		STM32Cube NVIC driver	

3.2.3 Peripheral configuration

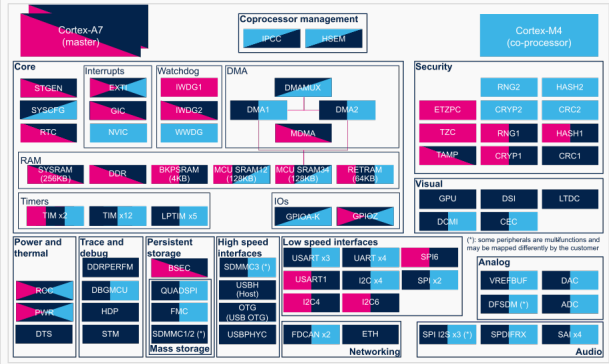
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in 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 possiblities might be described in STM32MP15 reference manuals
.



Internal peripherals

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