



NVIC internal peripheral



Contents

1. NVIC internal peripheral	3
2. STM32MP15 resources	6
3. STM32CubeMP1 architecture	9
4. STM32MP15 interrupts	13
5. STM32CubeMX	16
6. STM32MPU Embedded Software architecture overview	20
7. How to assign an internal peripheral to a runtime context	23



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

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 non-secure (OTE)	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.

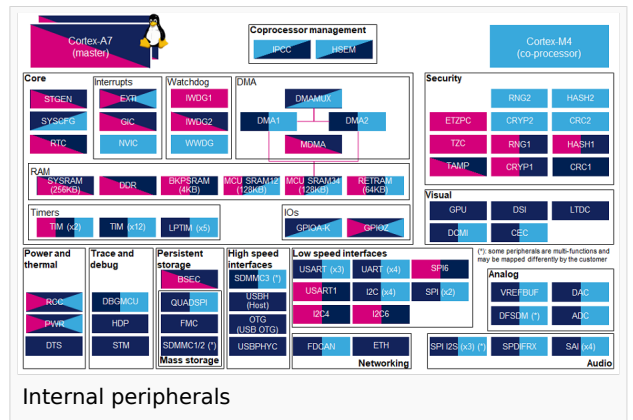
3.2.4 Peripheral assignment

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 possibilities might be described in STM32MP15 reference manuals.



Do	Per	Runtime allocation			Comme
ma	in	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
in	era				
	x-				
	A				
In	7				
st	se				
a	re				
nc	(
e	O				
	P				
	T				
	E				
	E)				
C					
ore					
/l	N				
nt	V	NVIC			
er	I				
ru	C				
pt					
s					



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	6
2 Peripheral overview	7
2.1 Features	7
2.2 Security support	7
3 Peripheral usage and associated software	7
3.1 Boot time	7
3.2 Runtime	7
3.2.1 Overview	7
3.2.2 Software frameworks	7
3.2.3 Peripheral configuration	8
3.2.4 Peripheral assignment	8
4 How to go further	9
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

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

Do	Peri	Software frameworks			Comment
main secure (O P- TE E)	n- sec ure (Li nux)	(STM32Cube)			
					Core /In ter ru pts

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.

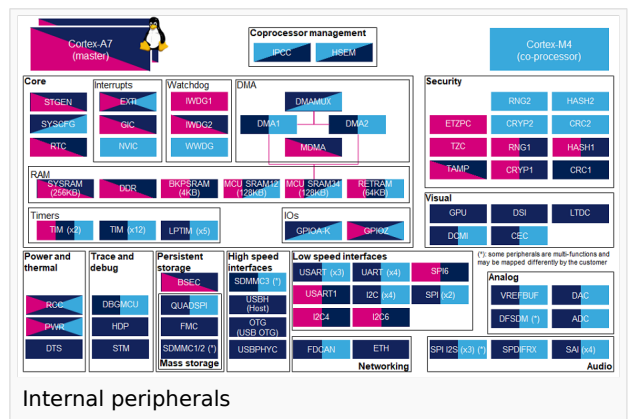
3.2.4 Peripheral assignment

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.



Do	Peri	Runtime allocation		Comment
main in ter ru pts	C o r t e x - A 7 in se cu re	Cortex-A7	Cortex-M4	



Do ma in nc e	Per iph era (O P- T E E)	Runtime allocation			Comme nt
		non-secure (Linux)	(STM32Cube)		
C or e / I nt er ru pt s	N V I C	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	10
2 Peripheral overview	10
2.1 Features	10
2.2 Security support	10
3 Peripheral usage and associated software	11
3.1 Boot time	11
3.2 Runtime	11
3.2.1 Overview	11
3.2.2 Software frameworks	11
3.2.3 Peripheral configuration	11
3.2.4 Peripheral assignment	11
4 How to go further	13
5 References	13

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.

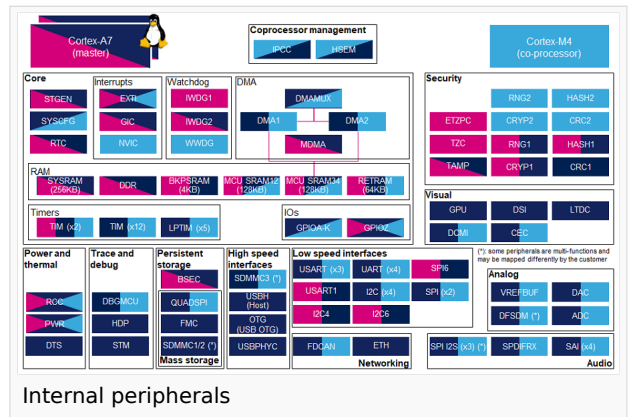
3.2.4 Peripheral assignment

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.

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.



Do	Per	Runtime allocation				Comme
ma	in	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
in	er					
	te					
	x-					
	A					
	7					
	se					
	cu					
	re					
	(
	O					
	P-					
	T					
	E					
	E)					
C						
o						
r						
e						
/						
i						
n						
t						
e						
r						
u						
r						
r						
u						
p						
t						
s						



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	14
2 Peripheral overview	14
2.1 Features	14
2.2 Security support	14
3 Peripheral usage and associated software	14
3.1 Boot time	14
3.2 Runtime	14
3.2.1 Overview	14
3.2.2 Software frameworks	15
3.2.3 Peripheral configuration	15
3.2.4 Peripheral assignment	15
4 How to go further	16
5 References	16



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 (OPE-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.

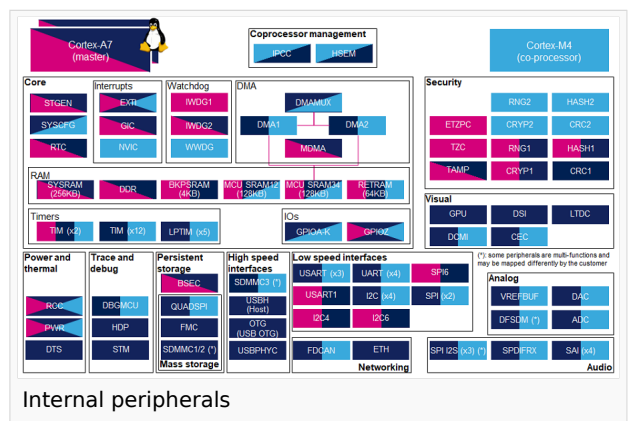
3.2.4 Peripheral assignment

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 possibilities might be described in [STM32MP15 reference manuals](#).



Do	Peri	Runtime allocation			Comment
main in Cortex-A7	Cortex-A7				



Do ma in st a nc e	Per in h era x A 7 se cu re (O P T E E)	Runtime allocation				Comme nt
		C or e / I nt er ru pt s	N V I C	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
		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



Contents

1 Article purpose	17
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	19
5 References	20

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 Cortex -A7 no sec ure (O P- TE E)	Cor tex -A7 no n- sec ure (Li nux)	Cortex-M4 (STM32Cube)			
Co re /In ter ru pts	N VIC		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

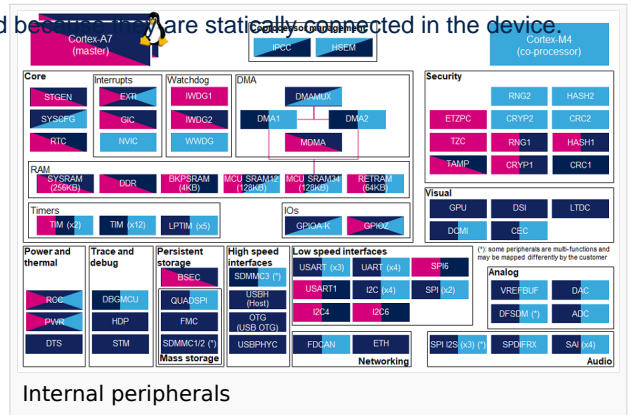
Check boxes illustrate the possible peripheral allocations supported by [STM32 MPU Embedded Software](#):

- 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.



		Runtime allocation				Comment
Do ma in in st a nc e	Peripherals Cortex-A7 non-secure (Linux) (O-P-T-E)	Cortex-A7 non-secure (Linux)				
		Cortex-M4 (STM32Cube)				
Core / Interrupts	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	20
2 Peripheral overview	21
2.1 Features	21
2.2 Security support	21
3 Peripheral usage and associated software	21
3.1 Boot time	21
3.2 Runtime	21
3.2.1 Overview	21
3.2.2 Software frameworks	21
3.2.3 Peripheral configuration	22
3.2.4 Peripheral assignment	22
4 How to go further	23
5 References	23

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-A7no	Cortex-M4	

Do	Peri	Software frameworks			Comment
main secure (O P- TE E)	n- sec ure (Li nux)	(STM32Cube)			
Co re / In ter ru pts	N V I 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.

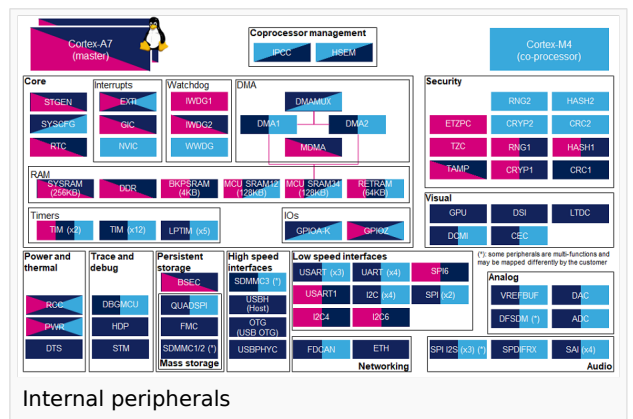
3.2.4 Peripheral assignment

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.



Do	Peri	Runtime allocation		Comment
main in ter ru pts	C o r e / A 7 s e c u r e	Cortex-A7		
		Cortex-M4		



Do ma in nc e	Per iph era (O P- T E E)	Runtime allocation				Comme nt
		non-secure (Linux)	(STM32Cube)			
C or e / I nt er ru pt s	N V I C	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	24
2 Peripheral overview	24
2.1 Features	24
2.2 Security support	24
3 Peripheral usage and associated software	25
3.1 Boot time	25
3.2 Runtime	25
3.2.1 Overview	25
3.2.2 Software frameworks	25
3.2.3 Peripheral configuration	25
3.2.4 Peripheral assignment	25
4 How to go further	27
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

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

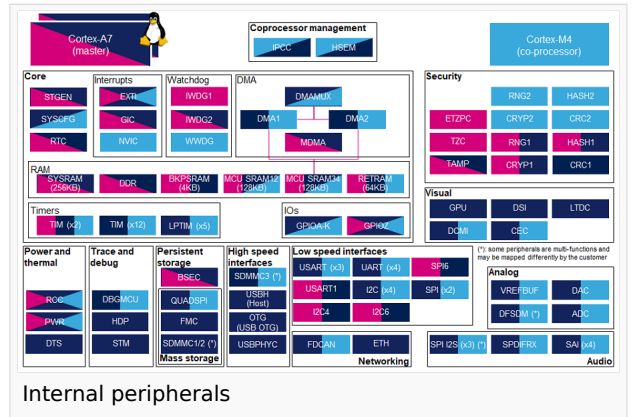
NVIC internal peripheral

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.



Do	Per	Runtime allocation				Comme
ma	in	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Inte	ra					
x-	A					
In	se					
sta	cu					
nc	re					
e	(
	O					
	P					
	T					
	E					
	E)					
C						
or						
e/						
Int	N	NVIC				
er	V					
ru	I					
pt	C					
s						



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