



HASH internal peripheral

HASH internal peripheral



Contents

1. HASH internal peripheral	3
2. Crypto API overview	9
3. ETZPC internal peripheral	15
4. How to assign an internal peripheral to a runtime context	21
5. OP-TEE overview	27
6. STM32CubeMP1 architecture	33
7. STM32CubeMX	39
8. STM32MP15 resources	45
9. STM32MPU Embedded Software architecture overview	51



A quality version of this page, approved on *12 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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

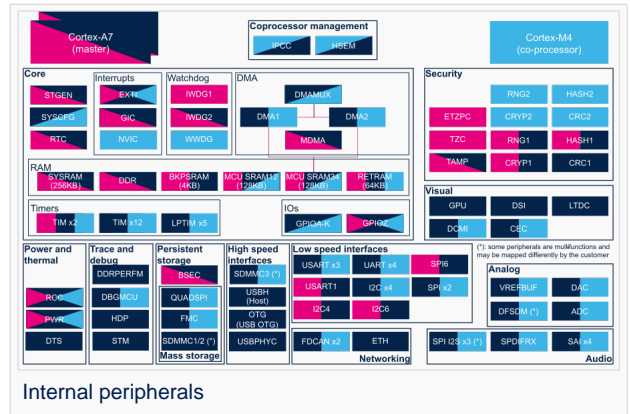
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

Stable: 19.10.2020 - 09:54 / Revision: 19.10.2020 - 09:52

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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

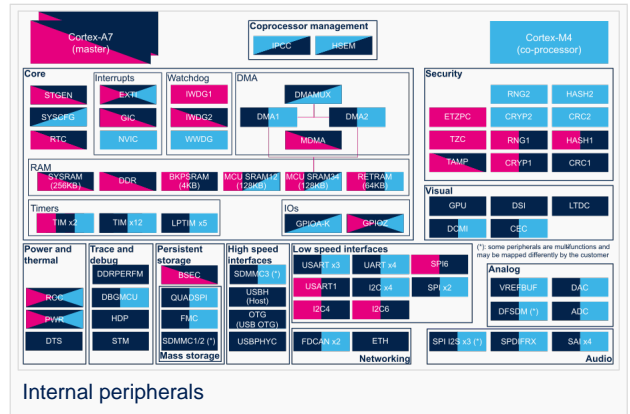
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

Stable: 31.07.2020 - 14:57 / Revision: 31.07.2020 - 14:56

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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
 HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm® Cortex®-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm® Cortex®-A7 non-secure core to be controlled in Linux® by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm® Cortex®-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

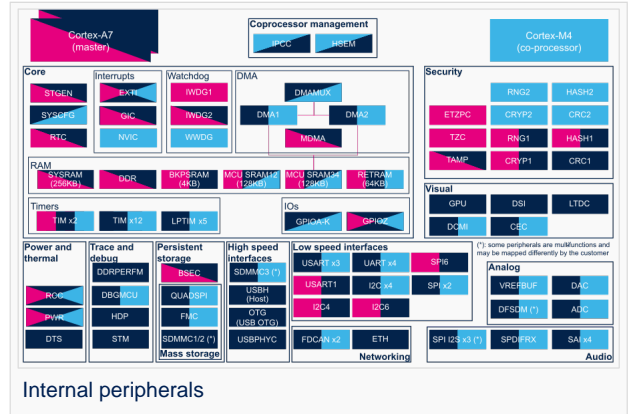
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

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

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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

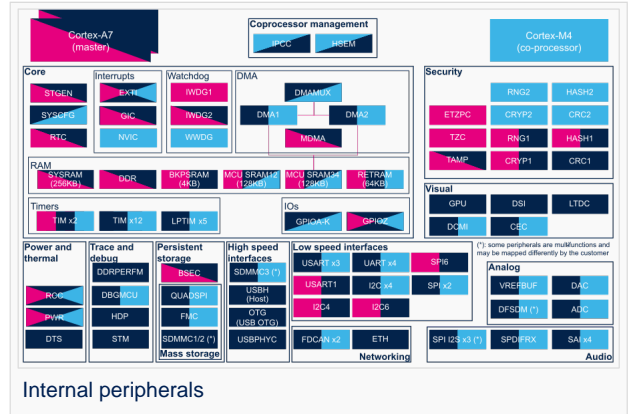
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

Stable: 13.05.2020 - 08:56 / Revision: 13.05.2020 - 08:54

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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

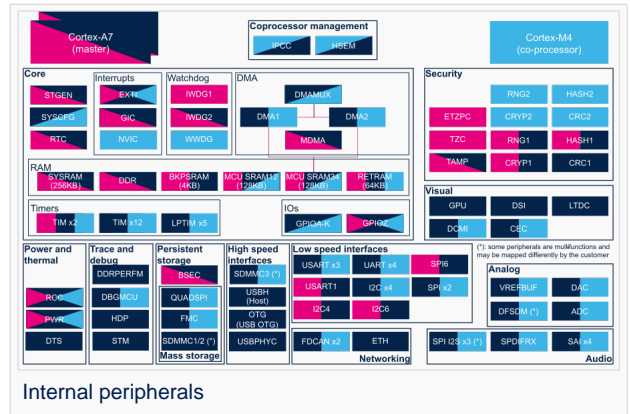
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

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

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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

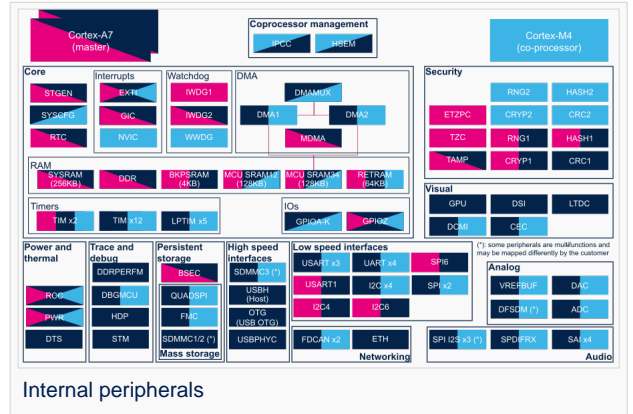
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

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

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 HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.

HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver

or

- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

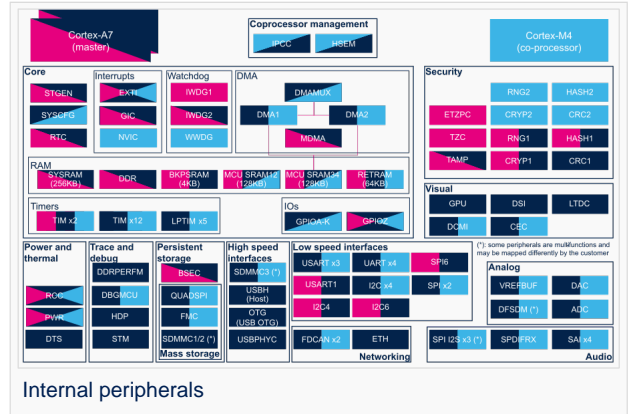
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

Stable: 17.11.2021 - 16:41 / Revision: 17.11.2021 - 10:47

Contents

1 Article purpose	46
2 Peripheral overview	47
2.1 Features	47
2.2 Security support	47
3 Peripheral usage and associated software	48
3.1 Boot time	48
3.2 Runtime	48
3.2.1 Overview	48
3.2.2 Software frameworks	48
3.2.3 Peripheral configuration	48
3.2.4 Peripheral assignment	48
4 How to go further	50
5 References	51



1 Article purpose

The purpose of this article is to:

- briefly introduce the HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.
HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver
- or
- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

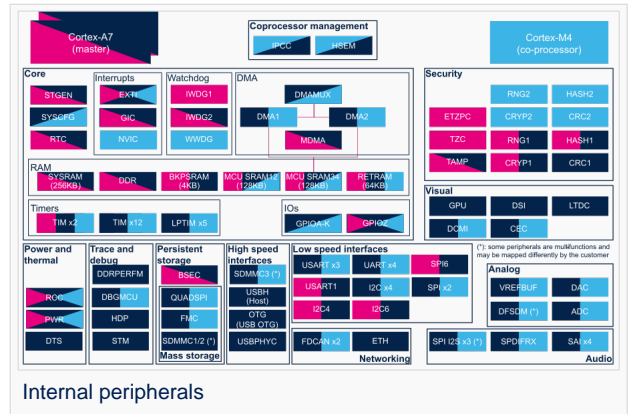
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>

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

Contents

1 Article purpose	52
2 Peripheral overview	53
2.1 Features	53
2.2 Security support	53
3 Peripheral usage and associated software	54
3.1 Boot time	54
3.2 Runtime	54
3.2.1 Overview	54
3.2.2 Software frameworks	54
3.2.3 Peripheral configuration	54
3.2.4 Peripheral assignment	54
4 How to go further	56
5 References	57



1 Article purpose

The purpose of this article is to:

- briefly introduce the HASH peripheral and its main features
- indicate the level of security supported by this hardware block
- explain how each instance can be allocated to the three runtime contexts and linked to the corresponding software components
- explain, when necessary, how to configure the HASH peripheral.



2 Peripheral overview

The **HASH** peripheral is used to compute a message digest.

Digest algorithms could be:

- MD5^[1]
- SHA^[2] (1, 224, 256)

The **HASH** peripheral is also able to give the HMAC^[3] used for authentication using the same algorithm support.

2.1 Features

Refer to the [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

HASH1 is a **secure** peripheral (under ETZPC control)

HASH2 is a **non secure** peripheral .



3 Peripheral usage and associated software

3.1 Boot time

HASH1 instance is used as boot device to support binary authentication.

HASH2 is not used at boot time.

3.2 Runtime

3.2.1 Overview

HASH1 instance can be allocated to:

- the Arm[®] Cortex[®]-A7 secure core to be controlled in OP-TEE by the OP-TEE HASH driver

or

- the Arm[®] Cortex[®]-A7 non-secure core to be controlled in Linux[®] by the Linux Crypto framework

HASH2 instance can be allocated to:

- the Arm[®] Cortex[®]-M4 to be controlled in STM32Cube MPU Package by STM32Cube HASH driver

Chapter [Peripheral assignment](#) describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH 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.

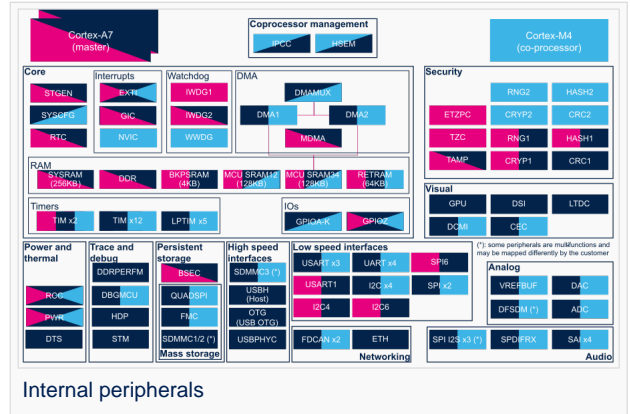
3.2.4 Peripheral assignment

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	Periphera	Runtime allocation		Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	
Security	HASH	HASH1		Assignment (single choice)
		HASH2		



4 How to go further

Not applicable.



5 References

- <https://en.wikipedia.org/wiki/MD5>
- https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
- <https://en.wikipedia.org/wiki/HMAC>