



HASH internal peripheral



Contents

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



HASH internal peripheral

Stable: 24.09.2019 - 13:54 / Revision: 24.09.2019 - 13:54

A quality version of this page, [accepted](#) on 24 September 2019, was based off this revision.

Contents

1 Article purpose	3
2 Peripheral overview	3
2.1 Features	4
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	5
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 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

Do	Peri	Software frameworks	Comment
Cor tex -A7	Cor tex -A7 no	Cortex-M4	

Do	Peri	Software frameworks			Comment
main (OP-TEE)	non-secure (Linux)	(STM32Cube)			
		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.

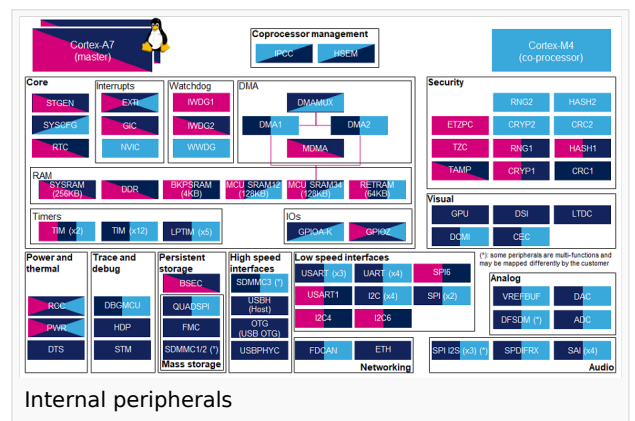
3.2.4 Peripheral assignment

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.



Do	Peri	Runtime allocation			Comment
main (Insecure)	Cortex-A7 (secure)	Cortex-A7 non-secure (Linux)			
		Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		



Do ma in	Per i n t e r f a c e	Runtime allocation				Comme nt
S e c u r i t y	H A S H	HASH1				Assig nment (singl e choic e)
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit

HASH internal peripheral

Stable: 21.02.2020 - 08:59 / Revision: 14.02.2020 - 10:13

Contents

1 Article purpose	7
-------------------------	---



2 Peripheral overview	7
2.1 Features	7
2.2 Security support	7
3 Peripheral usage and associated software	8
3.1 Boot time	8
3.2 Runtime	8
3.2.1 Overview	8
3.2.2 Software frameworks	8
3.2.3 Peripheral configuration	8
3.2.4 Peripheral assignment	9
4 How to go further	10
5 References	10

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

Do	Peri	Software frameworks			Comment
mai Cor tex -A7 sec ure (O P- T E E)	Cor tex -A7 no n- sec ure (Li nux)	Cortex-M4 (STM32Cube)			
Se cu rity	H A S H	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.

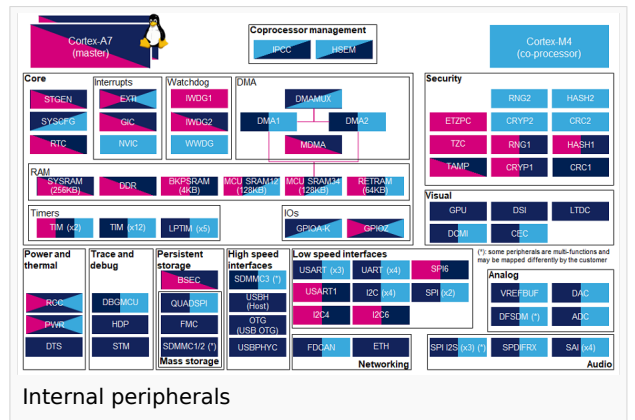
3.2.4 Peripheral assignment

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](#).



Do Per	ma in	Runtime allocation				Comme nt
S e c u r i t y	H A S H	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			Assignment (single choice)
		HASH1				
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit

HASH internal peripheral

Stable: **Not stable** / Revision: 12.02.2020 - 16:43

Contents

1 Article purpose	11
2 Peripheral overview	11
2.1 Features	11
2.2 Security support	11
3 Peripheral usage and associated software	11
3.1 Boot time	11
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	13
5 References	14

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

Do	Peri	Software frameworks			Comment
mai Cor tex -A7 sec ure (O P- T E E)	Cor tex -A7 no n- sec ure (Li nux)	Cortex-M4 (STM32Cube)			
Se cu rity	H A S H	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.



5 References

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

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit

HASH internal peripheral

Stable: 12.03.2020 - 12:15 / Revision: 14.10.2019 - 14:35

Contents

1 Article purpose	14
2 Peripheral overview	15
2.1 Features	15
2.2 Security support	15
3 Peripheral usage and associated software	15
3.1 Boot time	15
3.2 Runtime	15
3.2.1 Overview	15
3.2.2 Software frameworks	16
3.2.3 Peripheral configuration	16
3.2.4 Peripheral assignment	16
4 How to go further	17
5 References	17

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

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

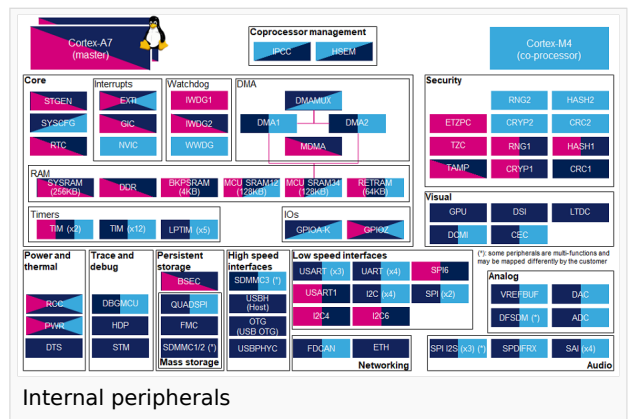
3.2.4 Peripheral assignment

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.



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



HASH internal peripheral

Do main st a nc e	Per iph era l (O P T E E)	Runtime allocation				Comme nt
		Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
S e c u r i t y	H A S H	HASH1				Assig nment (singl e choic e)
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit



HASH internal peripheral

Stable: **Not stable** / Revision: 31.03.2020 - 14:35

Contents

1 Article purpose	18
2 Peripheral overview	18
2.1 Features	19
2.2 Security support	19
3 Peripheral usage and associated software	19
3.1 Boot time	19
3.2 Runtime	19
3.2.1 Overview	19
3.2.2 Software frameworks	19
3.2.3 Peripheral configuration	20
3.2.4 Peripheral assignment	20
4 How to go further	21
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

Do	Peri	Software frameworks	Comment
Cor tex -A7	Cor tex -A7 no	Cortex-M4	

Do	Peri	Software frameworks			Comment
main (OP-TEE)	non-secure (Linux)	(STM32Cube)			
		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.

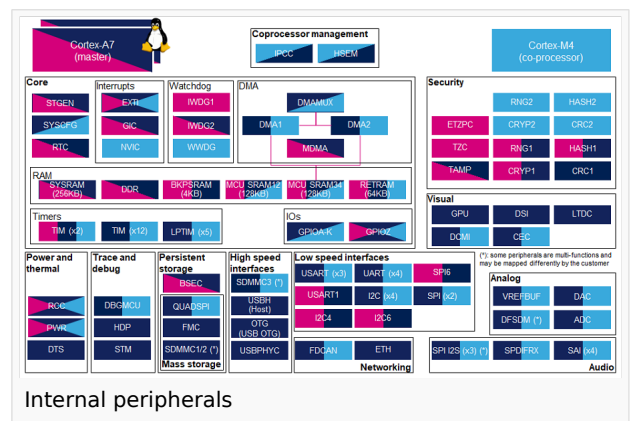
3.2.4 Peripheral assignment

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.



Internal peripherals

Do	Peri	Runtime allocation		Comment
main (OP-TEE)	non-secure (Linux)	Cortex-A7	Cortex-M4	
		non-secure (Linux)	(STM32Cube)	



Do ma in	Per i n t e r f a c e	Runtime allocation				Comme nt
S e c u r i t y	H A S H	HASH1				Assig nment (singl e choic e)
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit

HASH internal peripheral

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

Contents

1 Article purpose	22
-------------------------	----



2 Peripheral overview	22
2.1 Features	22
2.2 Security support	22
3 Peripheral usage and associated software	23
3.1 Boot time	23
3.2 Runtime	23
3.2.1 Overview	23
3.2.2 Software frameworks	23
3.2.3 Peripheral configuration	23
3.2.4 Peripheral assignment	24
4 How to go further	25
5 References	25

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

Do	Peri	Software frameworks			Comment
mai Cor tex -A7 sec ure (O ure P- TE E)	Cor tex -A7 no n- sec ure (Li nux)	Cortex-M4 (STM32Cube)			
Se cu rity	H AS H	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.

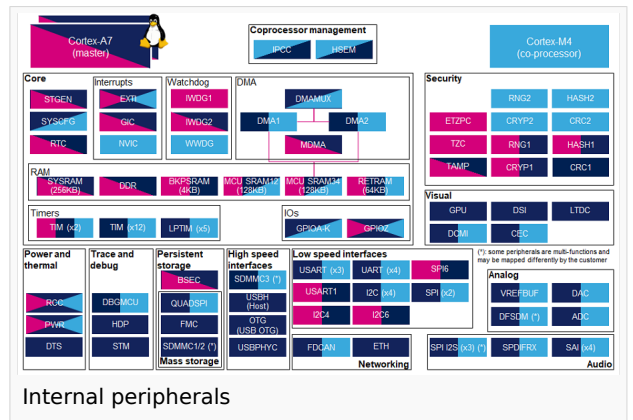
3.2.4 Peripheral assignment

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](#).



Do Per ma in In st a nc e	Per i n t e r n a l C o r t e x - A 7 s e c u r e (O P - T E E)	Runtime allocation				Comme nt
		Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
S e c u r i t y	H A S H	HASH1				Assig nment (singl e choic e)
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit

HASH internal peripheral

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

Contents

1 Article purpose	26
2 Peripheral overview	26
2.1 Features	26
2.2 Security support	26
3 Peripheral usage and associated software	26
3.1 Boot time	26
3.2 Runtime	27
3.2.1 Overview	27
3.2.2 Software frameworks	27
3.2.3 Peripheral configuration	27
3.2.4 Peripheral assignment	27
4 How to go further	28
5 References	29

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

Do	Peri	Software frameworks			Comment
mai Cor tex -A7 sec ure (O P- T E E)	Cor tex -A7 no n- sec ure (Li nux)	Cortex-M4 (STM32Cube)			
Se cu rity	H A S H	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.



5 References

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

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit

HASH internal peripheral

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

Contents

1 Article purpose	29
2 Peripheral overview	30
2.1 Features	30
2.2 Security support	30
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	31
3.2.3 Peripheral configuration	31
3.2.4 Peripheral assignment	31
4 How to go further	32
5 References	32

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

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

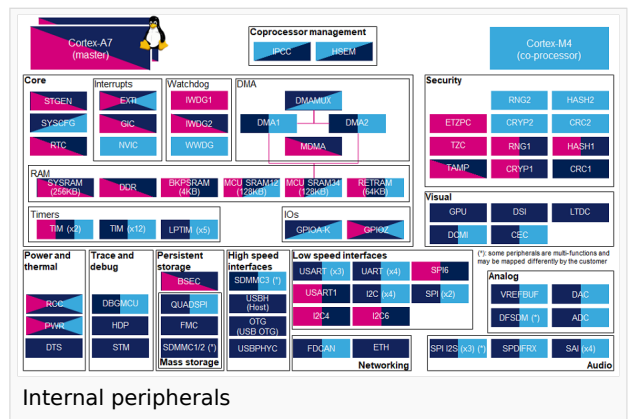
3.2.4 Peripheral assignment

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.



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



HASH internal peripheral

Do	Per	Runtime allocation				Comme nt
		Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
S e c u r i t y	H A S H	HASH1				Assig nment (singl e choic e)
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit



HASH internal peripheral

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

Contents

1 Article purpose	33
2 Peripheral overview	33
2.1 Features	34
2.2 Security support	34
3 Peripheral usage and associated software	34
3.1 Boot time	34
3.2 Runtime	34
3.2.1 Overview	34
3.2.2 Software frameworks	34
3.2.3 Peripheral configuration	35
3.2.4 Peripheral assignment	35
4 How to go further	36
5 References	36

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

Do	Peri	Software frameworks	Comment
Cor tex -A7	Cor tex -A7 no	Cortex-M4	

Do	Peri	Software frameworks			Comment
main (OP-TEE)	non-secure (Linux)	(STM32Cube)			
		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.

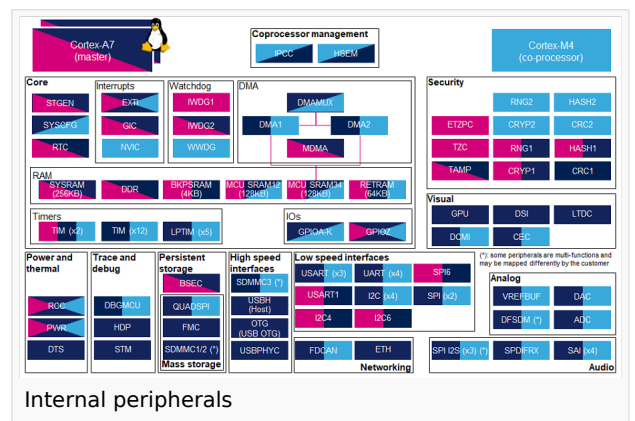
3.2.4 Peripheral assignment

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.



Do	Peri	Runtime allocation			Comment
main (Insecure)	Cortex-A7 (non-secure)	Cortex-M4 (STM32Cube)			
		Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		



Do ma in	Per i o p a r t e n t e r n e t e	Runtime allocation				Comme nt
S e c u r i t y	H A S H	HASH1				Assig nment (singl e choic e)
		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>

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit