HASH internal peripheral

HASH internal peripheral

Stable: 12.02.2020 - 16:46 / Revision: 12.02.2020 - 16:44

Contents

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

Domain	Peri	Software frameworks			Comment
secure (OP-TEE)	non-secure (Linux)	(STM32Cube)			
Security	HASH	OP-TEE HASH driver	Linux Crypto framework	STM32Cube HASH driver	

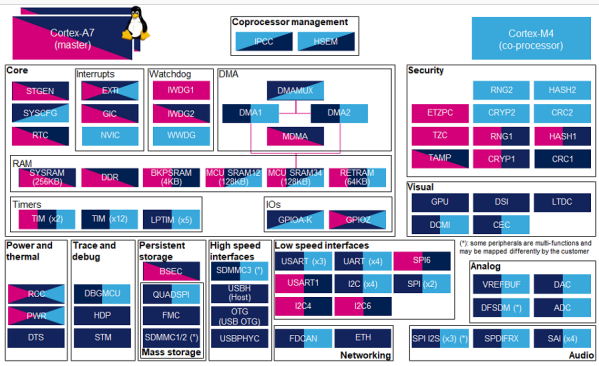
3.2.3 Peripheral configuration

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.



Internal peripherals

Refer to How to assign an internal peripheral to a runtime context for more information on how to assign peripherals manually or via STM32CubeMX.
The present chapter describes STMicroelectronics recommendations or choice of implementation. Additional possiblities might be described in STM32MP15 reference manuals.

Domain	Peripheral	Runtime allocation		Comment
Instance	Cortex-A7 secure (	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)	

Domain	Peripheral	Runtime allocation				Comment
	OP-TEE)					
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

Message Digest 5

Secure Hash Algorithm

Hash-based Message Authentication Code

Open Portable Trusted Execution Environment

Microprocessor Unit