TAMP internal peripheral

Contents

TAMP internal peripheral

Stable: 24.09.2019 - 13:57 / Revision: 19.09.2019 - 13:53

Contents

1 Article purpose

The purpose of this article is to:

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

2 Peripheral overview

The **TAMP** peripheral is used to prevent any attempt by an attacker to perform an unauthorized physical or electronic action against the device. It also includes the backup registers that remain powered-on when the platform is switched off.

It is important to notice that the backup registers are erased when a tamper detection occurs in TAMP internal peripheral

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

The TAMP is a **secure** peripheral. The access to some backup registers can be opened to the **non-secure** world via a direct configuration in TAMP, in **secure** mode.

3 Peripheral usage and associated software

3.1 Boot time

The TAMP is used at boot time to share data between the ROM code, FSBL and SSBL: see backup registers for further information.

3.2 Runtime

3.2.1 Overview

TAMP is seen as a system peripheral, that can be used by the Arm® Cortex®-A7 secure context with OP-TEE or the Arm® Cortex®-A7 non-secure context with Linux.

3.2.2 Software frameworks

Do	Peri	Software frameworks			Comment
main Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Security	TAMP	OP-TEE	Linux syscon framework[1]		

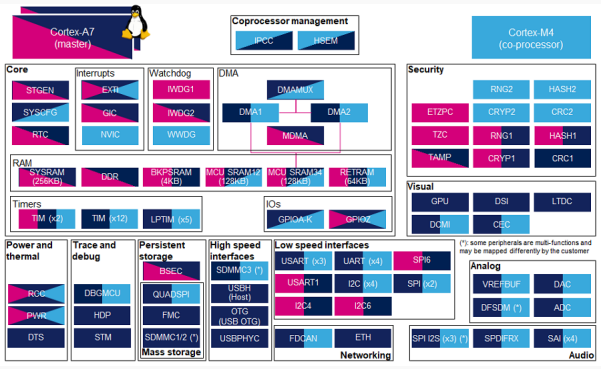
3.2.3 Peripheral configuration

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 (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Security	TAMP	TAMP				

4 How to go further

5 References

- Documentation/devicetree/bindings/mfd/syscon.txt

Tamper

Read Only Memory

First Stage Boot Loader

Second Stage Boot Loader

Open Portable Trusted Execution Environment

STM32MP15 backup registers

Stable: 12.02.2019 - 08:18 / Revision: 06.12.2018 - 11:20

Invalid target: no reviewed revision corresponds to the given ID.

Return to STM32MP15 backup registers.

STM32MP15 resources

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

Invalid target: no reviewed revision corresponds to the given ID.

Return to STM32MP15 resources.

OP-TEE overview

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

Invalid target: no reviewed revision corresponds to the given ID.

Return to OP-TEE overview.

STM32CubeMX

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

Invalid target: no reviewed revision corresponds to the given ID.

Return to STM32CubeMX.

STM32MPU Embedded Software architecture overview

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

Invalid target: no reviewed revision corresponds to the given ID.

Return to STM32MPU Embedded Software architecture overview.

How to assign an internal peripheral to a runtime context

Stable: 22.01.2020 - 16:08 / Revision: 22.01.2020 - 10:33

Invalid target: no reviewed revision corresponds to the given ID.

Return to How to assign an internal peripheral to a runtime context.