RTC internal peripheral

A quality version of this page, approved on *19 February 2020*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the RTC 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 needed, how to configure the RTC peripheral.

2 Peripheral overview

The **RTC** peripheral is used to provide the date and clock to the application. It supports programmable alarms and wake up capabilities.

2.1 Features

Refer to STM32MP15 reference manuals for the complete list of features, and to the software components, introduced below, to know which features are really implemented.

2.2 Security support

The **RTC** peripheral is a **secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

By default after a backup domain power-on reset (performed at boot time), all RTC registers can be read or written in both secure and non-secure modes.
In OpenSTLinux distribution, the first stage bootloader (FSBL, running in secure mode) keeps this default configuration, leaving full control to Linux® at runtime.

The **RTC** peripheral is able to generate two interrupts:

- A secure interrupt, connected to the Arm®Cortex®-A7 GIC, not used in OpenSTLinux distribution.
- A non-secure interrupt, connected both to Arm®Cortex®-A7 GIC and Cortex-M4 NVIC: this interrupt is used on Linux® and by default in OpenSTLinux distribution.

The **RTC** peripheral is part of the backup domain which reset and clock are controlled via the RCC by the first stage bootloader (FSBL, running in secure mode) at boot time.
The RTC reset occurs when the backup domain is reset. To avoid clearing the TAMP register contents, this is only done on cold boot, not on wake up.

3.2 Runtime

3.2.1 Overview

The **RTC** peripheral can be allocated to the Arm®Cortex®-A7 non-secure core to be used under Linux® with RTC framework.

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	STM32Cube			
Core	RTC		Linux RTC framework		

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 STM32CubeMX tool for all internal peripherals, and then manually completed (especially for external peripherals) according to the information given in the corresponding software framework article.

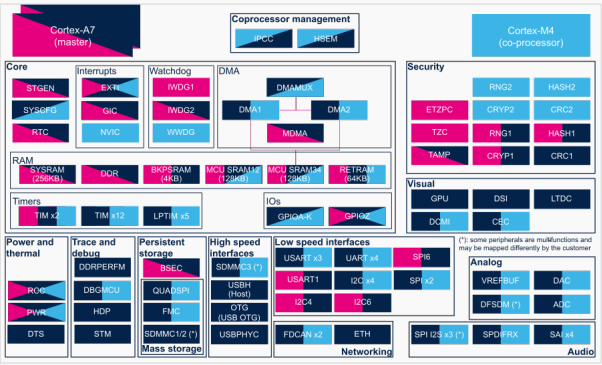
For Linux kernel configuration, please refer to RTC device tree configuration.

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 possiblities might be described in STM32MP15 reference manuals .



Internal peripherals

Domain	Peripheral	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core	RTC	RTC				RTC is mandatory to resynchronize STGEN after exiting low-power modes.

Real Time Clock

First Stage Boot Loader

Linux® is a registered trademark of Linus Torvalds.

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

Open Portable Trusted Execution Environment