GIC internal peripheral

Contents

Stable: 26.03.2021 - 13:17 / Revision: 18.03.2021 - 14:47

Stable: 26.03.2021 - 10:17 / Revision: 18.03.2021 - 14:47

A quality version of this page, approved on *26 March 2021*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the **GIC** peripheral (generic interrupt controller) 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 needed, how to configure the GIC peripheral.

2 Peripheral overview

The **GIC** peripheral is the Arm® Cortex®-A7 interrupt controller. It is consequently not accessible from the Arm® Cortex®-M4 core.

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 GIC is a **secure** peripheral (under ETZPC control).

3 Peripheral usage and associated software

3.1 Boot time

The GIC is configured by the FSBL (see Boot chain overview), mainly to define the routing of each interrupt to the secure or non-secure context at runtime.

3.2 Runtime

3.2.1 Overview

The GIC can be allocated:

- to the Arm® Cortex®-A7 secure core to be used under OP-TEE with the GIC OP-TEE driver (or TF-A secure monitor if the OP-TEE is not present)
- or to the Arm® Cortex®-A7 non-secure core to be used under Linux® with the interrupts framework

3.2.2 Software frameworks

Domain	Peripheral	Software components		Comment
OP-TEE	Linux	STM32Cube		
Core/Interrupts	GIC	OP-TEE GIC driver	Linux interrupt framework	

3.2.3 Peripheral configuration

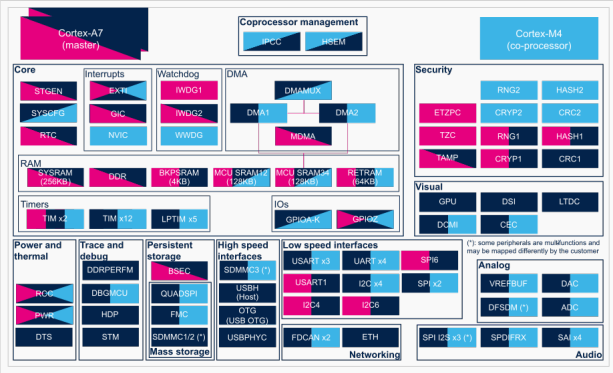
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration by itself can be performed via the STM32CubeMX tool for all internal peripherals. It can then be manually completed (especially for external peripherals) according to the information 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 possiblities might be described in STM32MP15 reference manuals .



Internal peripherals

Domain	Periphera l	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core /Interrupts	GIC	GIC				