PWR internal peripheral

PWR internal peripheral

Stable: 23.03.2020 - 09:16 / Revision: 23.03.2020 - 09:12

Contents

1 Article purpose

The purpose of this article is to:

- briefly introduce the PWR 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 PWR peripheral.

2 Peripheral overview

The **PWR** peripheral is used to control the device power supply configuration.

It has 6 input pins (named wakeup pins) which can be programmed to wake the system up from low power. The wakeup pins are listed with **WKUP** prefix in the STM32MP15 Datasheet.

These pins can be used by the Cortex®-A7 non secure (via Cortex®-A7 secure services) or the Cortex®-M4.

The PWR peripheral provides 2 output hardware lines named PWR_ON and PWR_LP:

- In STPMIC1 configuration, PWR_ON allows to select the register bank (active or low power). PWR_LP is not used.
- In the power discrete solution they drive VDDcore which feeds the Cortex®-A7, the Cortex®-M4 and the peripherals. They also control the DDR power supplies (VDD_DDR, VREF_DDR, VTT_DDR).

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 PWR is **secure aware** with the security control managed via RCC **TZEN** bit.

3 Peripheral usage and associated software

3.1 Boot time

The PWR is closely configured together with RCC by all the boot components: the ROM code, the FSBL, the SSBL and up to Linux® kernel. Its configuration is carried by the device tree.

3.2 Runtime

3.2.1 Overview

The PWR peripheral is shared at runtime:

- the Cortex®-A7 secure controls all secure registers (cf. TZEN description above) with PWR OP-TEE driver.

and

- the Cortex®-A7 non-secure mainly controls it via the regulator framework and the interrupt framework in Linux

and

- the Cortex®-M4 controls it in STM32Cube with PWR HAL driver

A concurrent control from each context is possible because the described management is realized via independent registers.

3.2.2 Software frameworks

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

Domain	Peri	Software frameworks			Comment
secure (OP-TEE)	non-secure (Linux)	(STM32Cube)			
Power & Thermal	PWR	OP-TEE PWR driver	Linux regulator framework	STM32Cube PWR driver	

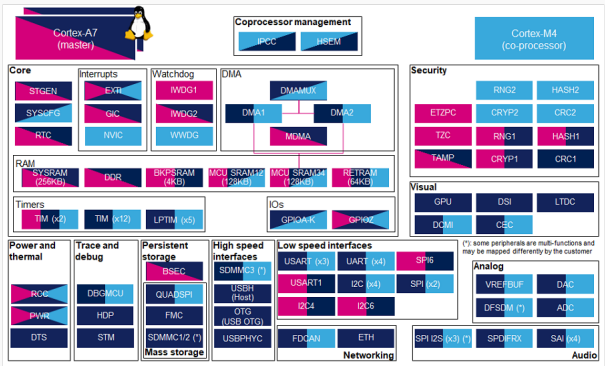
3.2.3 Peripheral configuration

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	Perih	Runtime allocation		Comment
	Cortex-A7			

Domain instance	Peripheral secure (OP-TEE)	Runtime allocation				Comment
		Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Power&Thermal	PWR	PWR				

4 How to go further

The PWR is interfaced with the hardware debug port (HDP) of the STM32MP15. This link offers the flexibility to observe the main PWR state signals on debug pins.

Please refer to STM32MP15 reference manuals for the exact list of signals that can be monitored.

5 References

Low Power (MIPI® Alliance DSI standard)

Doubledata rate (memory domain)

Read Only Memory

First Stage Boot Loader

Second Stage Boot Loader

Open Portable Trusted Execution Environment