RETRAM internal memory

Contents

RETRAM internal memory

Stable: 04.02.2020 - 15:59 / Revision: 04.02.2020 - 15:50

Contents

1 Peripheral overview

The **RETRAM** internal memory is 64 Kbytes wide and is physically near to the Arm® Cortex®-M4 for optimized performance from the core. It is located in the VSW power domain, allowing it to be supplied during Standby low power mode, and to retain retention firmware that can be executed very quickly by the Cortex-M4 on wake up from Standby mode.

1.1 Features

Refer to STM32MP15 reference manuals for the complete feature list, and to the software components introduced below to see which features are actually implemented.

1.2 Security support

The RETRAM is a **secure** peripheral (under ETZPC control).

2 Peripheral usage and associated software

2.1 Boot time

Linux® remoteproc framework (running on the Cortex-A7) loads the Cortex-M4 firmware to the RETRAM, starting at address 0x00000000. At least, it must load the part of the firmware containing the vector table, since the Cortex-M4 reset entry point is address 0x00000004. The rest of the firmware code is loaded into the MCU SRAM. The overall memory mapping is shown in the platform memory mapping section.

2.2 Runtime

2.2.1 Overview

The Cortex-M4 vector table is mapped from address 0x00000000 (so to the RETRAM) at reset, but it can be remapped by software to any other location by means of the vector table offset register (VTOR). Beyond the reset entry point (0x00000004), the exception table also contains the software entries table used by the NVIC to branch the software execution to the right interrupt service routine.

While going to Standby low power mode, the RETRAM can remain supplied, so it can preserve a (small) Cortex-M4 piece of retention firmware that is executed on wake up when the ROM code (running on Cortex-A7) restarts the Cortex-M4.
All these constraints make the RETRAM the minimum (and default) choice for Cortex-M4 firmware.

RETRAM can be allocated to:

- the Cortex-A7 secure to be used under OP-TEE.

or

- the Cortex-A7 non-secure to be used under Linux as reserved memory.

or

- the Cortex-M4 for use with the STM32Cube MPU Package, either for **runtime firmware** that ca be mapped in both RETRAM and MCU SRAM, or for **retention firmware** that only fits into the RETRAM, but could have some data in MCU SRAM (keeping in mind that these data are lost while entering Standby low power mode).

2.2.2 Software frameworks

Do	Peri	Software frameworks			Comment
main Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/RAM	RETRAM	OP-TEE overview	Linux reserved memory	STM32Cube	

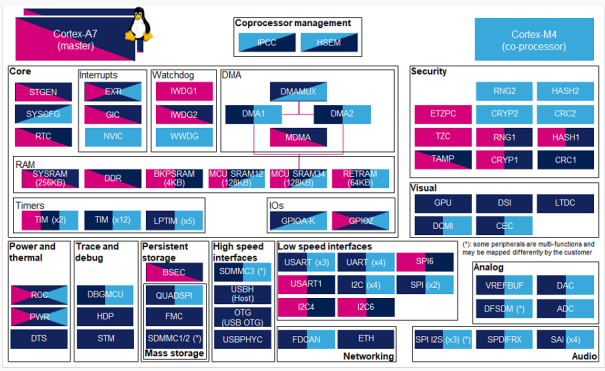
2.2.3 Peripheral configuration

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 (especially for external peripherals), according to the information given in the corresponding software framework article.

2.2.4 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	Periph	Runtime allocation				Comment
Instance	General	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/RAM	RETRAM	RETRAM				Assignment (single choice)

Microprocessor Unit

Open Portable Trusted Execution Environment

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today forvolatile random-accesssemiconductor memories.)

Power overview

Stable: 21.02.2019 - 14:20 / Revision: 20.02.2019 - 15:21

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

Return to Power overview.

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.

ETZPC internal peripheral

Stable: 20.02.2019 - 10:27 / Revision: 20.02.2019 - 10:27

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

Return to ETZPC internal peripheral.

Linux remoteproc framework overview

Stable: 03.06.2020 - 09:44 / Revision: 03.06.2020 - 09:44

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

Return to Linux remoteproc framework overview.

MCU SRAM internal memory

Stable: 04.02.2020 - 15:59 / Revision: 04.02.2020 - 15:48

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

Return to MCU SRAM internal memory.

STM32MP15 RAM mapping

Stable: 07.04.2020 - 12:20 / Revision: 25.03.2020 - 10:06

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

Return to STM32MP15 RAM mapping.

NVIC internal peripheral

Stable: 04.02.2020 - 15:41 / Revision: 04.02.2020 - 15:38

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

Return to NVIC internal peripheral.

STM32MP15 ROM code overview

Stable: 29.01.2020 - 16:12 / Revision: 29.01.2020 - 16:12

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

Return to STM32MP15 ROM code overview.

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.

Reserved memory

Stable: 31.01.2020 - 13:55 / Revision: 31.01.2020 - 13:47

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

Return to Reserved memory.

STM32CubeMP1 architecture

Stable: 21.02.2020 - 08:39 / Revision: 04.02.2020 - 15:22

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

Return to STM32CubeMP1 architecture.

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.