MCU SRAM internal memory

Stable: 04.02.2020 - 10:00 | Revision: 04.02.2020 - 10:45

A quality version of this page, accepted on *4 February 2020*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the MCU RAM memory 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 how to configure the MCU RAM memory.

2 Peripheral overview

The **MCU SRAM** internal memory is 384-Kbyte wide and physically near to the Cortex®-M4 for optimized performances from this core. It is split into four separate banks:

- MCU SRAM1 (128 Kbytes)
- MCU SRAM2 (128 Kbytes)
- MCU SRAM3 (64 Kbytes)
- MCU SRAM4 (64 Kbytes)

Those banks have individual security control (cf. security support below) and automatic clock gating (for power management optimization), but they are not supplied when the system goes to Standby low power mode, so their content is lost in that case.

2.1 Features

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

2.2 Security support

Each MCU SRAM1/SRAM2/SRAM3/SRAM4 bank is **secure aware** (under ETZPC control).

3 Peripheral usage and associated software

3.1 Boot time

The ROM code uses the MCU SRAM1 to store the USB context during a boot on USB for Flash programming (with STM32CubeProgrammer).

Linux remoteproc framework (running on the Cortex®-A7) loads the Cortex®-M4 firmware code into the MCU SRAM, except the exception table that must be loaded in the RETRAM since the Cortex®-M4 is looking for its reset entry point at address 0x00000000. The overall memory mapping is shown in the platform memory mapping section.

3.2 Runtime

3.2.1 Overview

Each MCU SRAM bank can be allocated to:

- the Arm® Cortex®-A7 secure for using in OP-TEE

or

- the Arm® Cortex®-A7 non-secure for using in Linux® with reserved memory, that is used by the dmaengine (for DMA buffers management) or RPMsg for interprocess communication with the coprocessor

and/or

- the Arm® Cortex®-M4 for using in STM32Cube

Notice the **and/or** allocation between Cortex®-A7 non-secure and Cortex®-M4, meaning that it is possible to share banks between those cores, typically to realize inter process communication between RPMsg on Linux side and OpenAMP on STM32Cube side.

The default assignement set in STMicroelectronics distribution is in line with the platform memory mapping, that can be adapted by the platform user.

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks			Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Core/RAM	MCU SRAM	OP-TEE overview	Linux reserved memory	STM32Cube	

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 done 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.

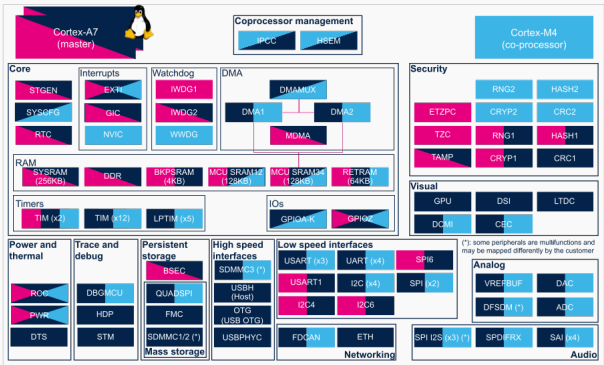
The several SRAM banks are accessible via different address ranges in order to benefit from the Cortex-M4 multiple ports.

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/RAM	MCU SRAM	SRAM1				Assignment (between A7 S and A7 NS / M4) Shareable (between A7 NS and M4)
		SRAM2				Assignment (between A7 S and A7 NS / M4) Shareable (between A7 NS and M4)
		SRAM3				Assignment (between A7 S and A7 NS / M4) Shareable (between A7 NS and M4)
		SRAM4				Assignment (between A7 S and A7 NS / M4) Shareable (between

Domain	Peripheral	Runtime allocation				Comment
						A7 NS and M4)

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

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.)

Open Portable Trusted Execution Environment