Category:Mass storage peripherals

Contents

Stable: 17.06.2020 - 15:26 / Revision: 16.01.2020 - 09:15

Stable: 17:06.2020 - 10:20 | Revision: 16:01.2020 - 00:13

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to the **mass storage** internal peripherals (hardware blocks) embedded in the STM32 MPUs microprocessor devices.

Pages in category "Mass storage peripherals"

The following 3 pages are in this category, out of 3 total.

- FMC internal peripheral
- QUADSPI internal peripheral
- SDMMC internal peripheral

Stable: 19.11.2020 - 10:48 / Revision: 12.11.2020 - 09:10

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

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the FMC peripheral 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 FMC peripheral.

2 Peripheral overview

The **FMC** peripheral includes two memory controllers:

- The NOR/PSRAM memory controller
- The NAND memory controller

2.1 NOR/PSRAM memory controller (or external bus interface controller)

The **FMC** NOR/PSRAM memory controller is used to interface static memory devices, but it is also used to interface Ethernet devices, LCD devices, ...

The **FMC** NOR/PSRAM controller generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR Flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The **FMC** NOR/PSRAM controller supports a wide range of devices through programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock output.

The **FMC** NOR/PSRAM controller supports up to four external devices.

2.2 NAND Flash controller

The **FMC** NAND Flash controller is used to interface STM32 MPU with SLC 8-bit or 16-bit NAND Flash memory devices.

The **FMC** NAND Flash controller supports:

- Programmable error correction capability (ECC) using BCH8 code, BCH4 code or Hamming code
- Programmable page size of 2048, 4096 and 8192 bytes
- Programmable memory timings
- Multiple dice per package.

2.3 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.

2.4 Security support

The FMC is a **non-secure** peripheral.

3 Peripheral usage and associated software

3.1 Boot time

The FMC NAND Flash controller is the boot device that supports serial boot for Flash programming with STM32CubeProgrammer.

3.2 Runtime

3.2.1 Overview

The FMC instance can be allocated to the Arm®Cortex®-A7 non-secure core. The FMC NAND Flash controller can be controlled in Linux® by the MTD framework.

Chapter #Peripheral assignment describes which instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks		Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)		
Mass storage	FMC		Linux MTD Framework	

3.2.3 Peripheral configuration

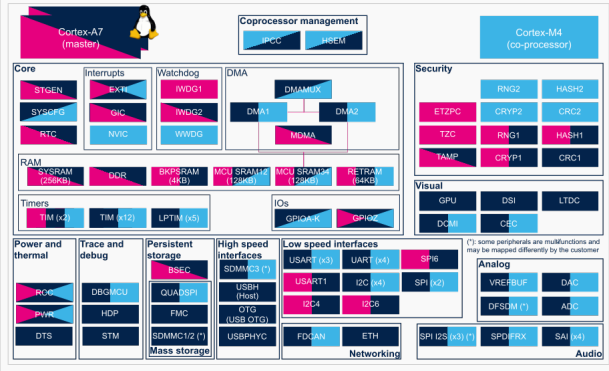
The configuration is applied by the firmware running in the context to which the peripheral is assigned. The configuration can be done alone via the STM32CubeMX tool for all internal peripherals, and then manually completed (particularly for external peripherals), according to the information given in the corresponding software framework article.

For Linux kernel configuration, please refer to FMC 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.



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	Periphera l	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Mass storage	FMC	FMC				

4 How to go further

5 References

Read Only Memory

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Microprocessor Unit

Single-Level Cell is a kind of NAND flash

Elliptic curve cryptography

Error Correction Capability

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

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment

Stable: 09.10.2019 - 12:44 / Revision: 16.09.2019 - 12:51

A quality version of this page, approved on *9 October 2019*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the QUADSPI peripheral 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 QUADSPI peripheral.

2 Peripheral overview

The Quad-SPI interface (**QUADSPI** peripheral) is used to interface the processor with serial NOR Flash and serial NAND Flash memories.
It supports:

- Single, dual or quad SPI Flash memories
- A dual-flash mode, allowing to agregate two Flash memories into a virtual single one
- Dual data rate and memory-mapped modes.

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

QUADSPI is **non secure** peripheral.

3 Using the peripheral - associated software

3.1 Boot time

QUADSPI instance is boot device that support serial boot for Flash programming with STM32CubeProgrammer.

3.2 Runtime

3.2.1 Overview

The QUADSPI instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the MTD framework

or

- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by QUADSPI HAL driver

Chapter #Peripheral assignment describes which peripheral instances can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks			Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Mass storage	QUADSPI		Linux MTD framework	STM32Cube QUADSPI 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.

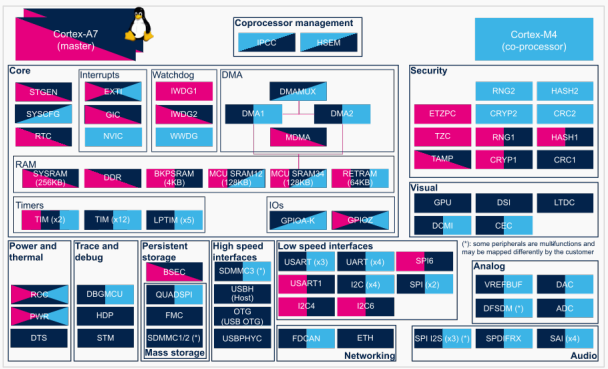
For Linux kernel configuration, please refer to QUADSPI device tree configuration.

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	Periphera l	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Mass storage	QUADSPI	QUADSPI				Assignment (single choice)

4 How to go further

5 References

Serial Peripheral Interface

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

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Microprocessor Unit

Open Portable Trusted Execution Environment

Stable: 14.05.2020 - 07:13 / Revision: 14.05.2020 - 07:12

A quality version of this page, approved on *14 May 2020*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to

- briefly introduce the SDMMC peripheral 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 necessary, how to configure the SDMMC peripheral.

2 Peripheral overview

The **SDMMC** peripheral is used to interconnect STM32 MPU to SD memory cards, SDIO and MMC devices.

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

SDMMC1/2/3 instances are either **non-secure** or **secure** peripherals (under ETZPC control).

Warning

- When an SDMMC instance is secure internal, the DMA cannot be used to perform data transfers.
- STMicroelectronics does not provide secure MMC driver (see below chapter)

3 Peripheral usage and associated software

3.1 Boot time

SDMMC1/2 instances can be used to support memory boot on SD or MMC Flash devices.

The SDMMC3 is not used at boot time.

Information

The SDMMC instances are ordered by address in the device tree arch/arm/boot/dts/stm32mp151.dtsi file:

```
sdmmc3: sdmmc@48004000 {
...
sdmmc1: sdmmc@58005000 {
...
sdmmc2: sdmmc@58007000 {
```

By default, in OpenSTLinux distribution, **sdmmc3 is disabled** so the sdmmc1 (SD card on Evaluation boards and Discovery kits) and sdmmc2 (eMMC on Evaluation boards and Wifi on Discovery kits) are respectively aliased to mmc0 and mmc1.
If you enable sdmmc3, it will take the mmc0 alias and the aliases above will shift, so don't forget to update the Linux kernel boot command accordingly!
For instance, 'root=/dev/mmcblk0p6' will become 'root=/dev/mmcblk1p6' to mount the rootfs from the sdmmc1 (SD card) when sdmmc3 is enabled.

3.2 Runtime

3.2.1 Overview

SDMMC1/2/3 instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by the MMC framework

or

- the Arm®Cortex®-M4 to be controlled in STM32Cube MPU Package by STM32Cube SDMMC driver

Chapter #Peripheral assignment describes which peripheral instance can be assigned to which context.

3.2.2 Software frameworks

Domain	Peripheral	Software frameworks			Comment
Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Mass storage	SDMMC		Linux MMC framework	STM32Cube SDMMC 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.

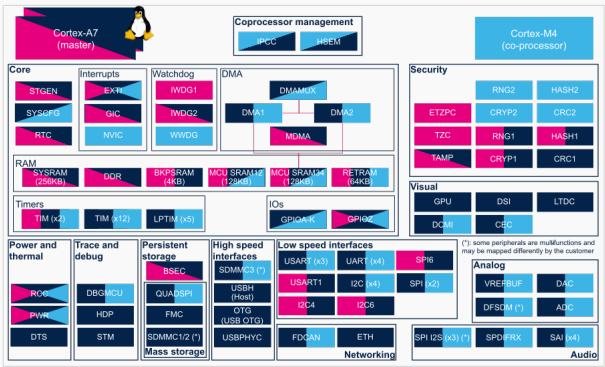
For Linux® kernel configuration, please refer to SDMMC device tree configuration.

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	Periphera l	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Mass storage	SDMMC	SDMMC1				
		SDMMC2				
		SDMMC3				Assignment (single choice)

4 How to go further

5 References

Microprocessor Unit

MultimediaCard

Direct Memory Access

SD memory card (https://www.sdcard.org)

former spelling for e•MMC ('e' in italic)

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