# Category:Audio peripherals

## Contents



Stable: 17.06.2020 - 15:26 / Revision: 16.01.2020 - 08:09

Stable: 17 Jun 2020 - 10:20 | Revision: 16 Jul 2020 - 08:00

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 **audio** internal peripherals (hardware blocks) embedded in the STM32 MPUs microprocessor devices.

# Pages in category "Audio peripherals"

The following 4 pages are in this category, out of 4 total.

- DFSDM internal peripheral
- SAI internal peripheral
- SPDIFRX internal peripheral
- SPI internal peripheral

Stable: 12.02.2020 - 16:42 / Revision: 12.02.2020 - 16:41

A quality version of this page, approved on *12 February 2020*, was based off this revision.

Contents

# 1 Article purpose

The purpose of this article is to

- briefly introduce the DFSDM 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 DFSDM peripheral.

# 2 Peripheral overview

The **DFSDM** peripheral (Digital Filter for Sigma-Delta Modulator) is used as a generic ADC. It benefits from external analog frontend interfaces and internal digital filters.

It can be used in various applications[1] such as: **audio record** with MEMS microphones, **energy measurement** with STPMS2[2] for electricity meters or motor control...

## 2.1 Features

The **DFSDM** peripheral provides several features, among which:

- Up to 8 external analog frontend serial interfaces (SPI, manchester coded single wire interface, clock output), for various sigma-delta modulators
- Up to 8 internal digital parallel interfaces (from internal ADC[3] or memory data stream via DMA[4] or CPU)
- Up to 6 digital filters, that offers up to 24-bit final ADC resolution
- Conversions that can be launched continuously, or using various triggers: by software, TIM[5], LPTIM[6], EXTI[7] or synchronously with DFSDM filter 0
- Event detectors: analog watchdog high/low thresholds, short-circuit detector, extremes detector
- Break generation to TIM[5] on analog watchdog or short-circuit detector events

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

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

# 3 Peripheral usage and associated software

## 3.1 Boot time

The DFSDM is not used at boot time.

## 3.2 Runtime

### 3.2.1 Overview

The DFSDM can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be used under Linux® with the IIO or ALSA framework

or

- the Arm®Cortex®-M4 for using in STM32Cube with STM32Cube DFSDM driver.

The peripheral assignment chapter 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) | | | |
| Analog | DFSDM | | Linux IIO framework Linux ALSA framework | STM32Cube DFSDM 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 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.

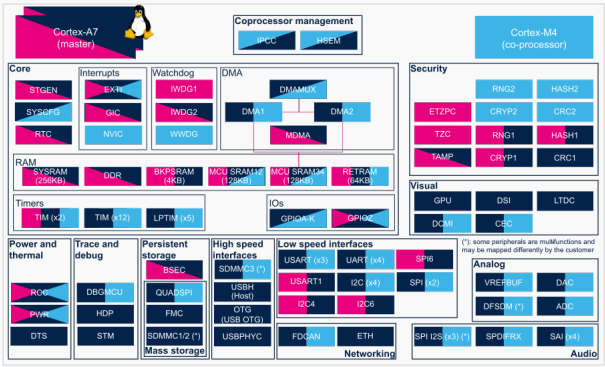
For the Linux kernel configuration, please refer to DFSDM device tree configuration and DFSDM Linux driver articles.

### 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) | | | |
| Analog | DFSDM | DFSDM | | | | Assignment (single choice) |

# 4 How to go further

See:

- *STM32L4 System Digital Filter for SD Modulators interface*[1], online DFSDM training with application examples from STMicroelectronics
- *Getting started with sigma-delta digital interface*[8], application note from STMicroelectronics

# 5 References

- 1.01.1 STM32L4 System Digital Filter for SD Modulators interface, online DFSDM training from STMicroelectronics
- STPMS2 "Smart sensor" device
- ADC internal peripheral
- DMA internal peripheral
- 5.05.1 TIM internal peripheral
- LPTIM internal peripheral
- EXTI internal peripheral
- Getting started with sigma-delta digital interface, application note from STMicroelectronics

Digital Filter for Sigma-Delta Modulator

Analog-to-digital converter. The process of converting a sampled analog signal to a digital code that represents the amplitude of the original signal sample.

Serial Peripheral Interface

Direct Memory Access

Central processing unit

low-power timer (STM32 specific)

External Interrupt



Open Portable Trusted Execution Environment

Stable: 21.01.2020 - 13:23 / Revision: 21.01.2020 - 13:21

A quality version of this page, approved on *21 January 2020*, was based off this revision.

Contents

# 1 Article purpose

The purpose of this article is to:

- briefly introduce the SAI 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 how to configure the SAI peripheral.

# 2 Peripheral overview

The SAI (Serial Audio Interface) offers a wide set of audio protocols, such as: I2S standards (LSB or MSB-justified), PCM/DSP, TDM and S/PDIF. The SAI contains two independent audio sub-blocks. Each sub-block has its own clock generator and I/O line controller, and can be configured either as transmitter or receiver.

## 2.1 Features

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

## 2.2 Security support

All the SAI instances are **non secure** peripherals.

# 3 Peripheral usage and associated software

## 3.1 Boot time

The SAI is not used at boot time.

## 3.2 Runtime

### 3.2.1 Overview

SAI instances can be allocated to:

- the Cortex-A7 non-secure for use in Linux with ALSA framework
- the Cortex-M4 for use in STM32Cube with STM32Cube SAI driver

Chapter #Peripheral assignment exposes 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) | | | |
| Audio | SAI | | ALSA framework | STM32Cube SAI 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, then manually completed (particularly for external peripherals), according to the information given in the corresponding software framework article.

#### *3.2.3.1 Configuration in Cortex-A7 non-secure software*

When the Arm®Cortex®-A7 core operates in non-secure access mode, the SAI is controlled by the Linux kernel framework. Refer to SAI Linux driver to drive the SAI through Linux kernel ALSA framework. Refer to Soundcard configuration and SAI device tree configuration to configure the SAI through the Linux kernel device tree[1].

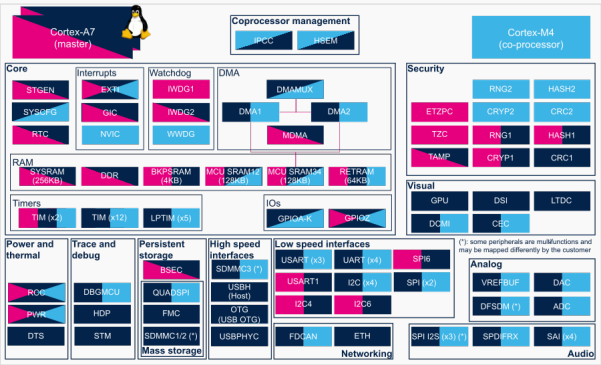
#### *3.2.3.2 Arm®Cortex®-M4 software configuration*

### 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) | | | |
| Audio | SAI | SAI1 | | | | Assignment (single choice) |
| | | SAI2 | | | | Assignment (single choice) |
| | | SAI3 | | | | Assignment (single choice) |
| | | SAI4 | | | | Assignment (single choice) |

# 4 How to go further

STM32H7 SAI training [2] introduces the SAI features and applications. The SAI versions in STM32H7 and STM32MP15 are very close. In consequence this training is also relevant for STM32MP15. The user should refer to the STM32MP15 reference manuals for a complete description.

# 5 References

- Device tree
- STM32H7 SAI training

Serial Audio Interface (Mechanism used to transfer non-buffered audio data between processors and/or audio converters.)

Integrated Interchip Sound

Sony/Philips Digital Interface Format (Protocol (IEC-60958))

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Open Portable Trusted Execution Environment



Stable: 08.03.2021 - 16:35 / Revision: 08.03.2021 - 14:55

A quality version of this page, approved on *8 March 2021*, was based off this revision.

Contents

# 1 Article purpose

The purpose of this article is to:

- briefly introduce the SPDIFRX 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 how to configure the SPDFIRX peripheral.

# 2 Peripheral overview

The **SPDIFRX** peripheral, is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. The SPDIFRX receiver provides two separated paths to retrieve the audio data and the user and channel information.

## 2.1 Features

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

## 2.2 Security support

The SPDFIRX is a **non secure** peripheral.

# 3 Peripheral usage and associated software

## 3.1 Boot time

The SPDFIRX is not used at boot time.

## 3.2 Runtime

### 3.2.1 Overview

The SPDIFRX instance can be allocated to:

- the Arm®Cortex®-A7 non-secure for use in Linux with ALSA framework
- the Cortex-M4 for use in STM32Cube with STM32Cube SPDFIRX driver

Chapter #Peripheral assignment exposes 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) | | | |
| Audio | SPDIFRX | | ALSA framework | STM32Cube SPDIFRX 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.

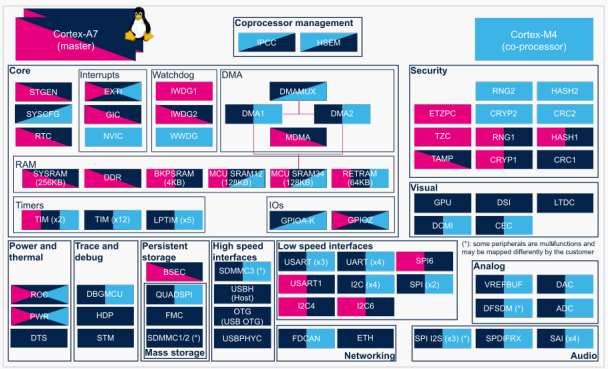
When the Arm®Cortex®-A7 core operates in non-secure access mode, the SPDIFRX is controlled by the Linux kernel framework. Refer to the SPDIFRX Linux driver to drive the SPDIFRX through Linux kernel ALSA framework. Refer to Soundcard configuration and SPDIFRX device tree configuration to configure the SPDIFRX through Linux kernel device tree[1].

### 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) | | | |
| Audio | SPDIFRX | SPDIFRX | | | | Assignment (single choice) |

# 4 How to go further

The STM32H7 SPDIFRX training [2], introduces the STM32 S/PDIF Receiver interface on the STM32H7. This training also applies to the STM32 MPU SPDIFRX internal peripheral.

## 5 References

- Device tree
- STM32H7 SPDIFRX training

Sony/Philips Digital Interface Format (Protocol (IEC-60958))



Open Portable Trusted Execution Environment

Microprocessor Unit

Stable: 04.01.2021 - 10:24 / Revision: 21.12.2020 - 10:48

A quality version of this page, approved on *4 January 2021*, was based off this revision.

Contents

# 1 Article purpose

The purpose of this article is to:

- briefly introduce the SPI 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 SPI peripheral and for some of them the I2S features.

# 2 Peripheral overview

The SPI peripheral can be used to communicate with an external devices using the SPI (Serial Peripheral Interface).
A subset of the SPI instances supports the I2S audio protocol. These **SPI/I2S** peripherals can alternatively be used in **audio** applications, when they are configured as an I2S interface. Refer to peripheral assignment chapter to check I2S feature support for each SPI instance.

## 2.1 Features

### 2.1.1 SPI main features

- Full-duplex, half-duplex and simplex synchronous modes.
- Slave and master modes.

### 2.1.2 I2S main features

Only available for SPI supporting I2S mode.

- Full-duplex or simplex modes.
- Slave and master modes.
- Four audio protocols supported.

### 2.1.3 Specific features

Some of the SPI peripheral characteristics depend on I2S support, as summarized in following table:

| SPI modes/features | I2S supported | I2S not supported |
|---|---|---|
| Rx & TxFIFO size (N) [x 8-bit] | 16 | 8 |
| Maximum configurable data size [bits] | 32 | 16 |

Refer to 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

SPI6 is a **secure** peripheral (under ETZPC control).
The other SPI instances are **non-secure** peripherals.

# 3 Peripheral usage and associated software

## 3.1 Boot time

The SPI is not used at boot time.

## 3.2 Runtime

### 3.2.1 Overview

The SPI6 can be allocated to:

- the Arm®Cortex®-A7 secure core to be controlled in OP-TEE by the SPI OP-TEE driver

All the SPI instances can be allocated to:

- the Arm®Cortex®-A7 non-secure core to be controlled in Linux® by:
  - the SPI framework for SPI configured in SPI mode
  - the ALSA framework for SPI configured in I2S mode

or

- the Cortex-M4 to be controlled in STM32Cube MPU Package by STM32Cube SPI 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) | | | |
| Low speed interface | SPI | OP-TEE SPI driver | Linux SPI framework | STM32Cube SPI driver | SPI configured in SPI mode **The OP-TEE SPI driver is not yet available** |
| Audio | SPI | | Linux ALSA framework | STM32Cube SPI driver | SPI configured in I2S mode **Only for SPI supporting I 2S feature** |

### 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 STM32CubeMX tool for all internal peripheral, then it can manually be completed (especially for external peripherals) according to the information given in the corresponding software framework article.

When the Arm®Cortex®-A7 core operates in non-secure access mode, the SPI is controlled by the Linux kernel framework.

- SPI mode:

Refer to SPI framework to check how to drive SPI through Linux kernel.

- I2S mode:

Refer to I2S Linux driver to drive the SPI through Linux kernel ALSA framework. Refer to Soundcard configuration to configure it through the Linux kernel device tree[1].

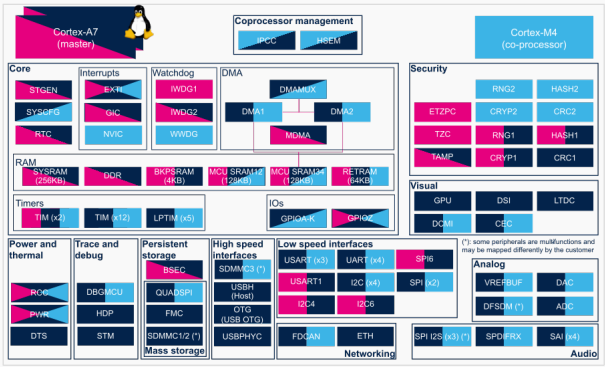
### 3.2.4 Peripheral assignment

**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) | | | |
| | | SPI2S1 | | | | Assignment (single choice) |
| | | SPI2S2 | | | | Assignment (single choice) |
| | | SPI2S3 | | | | Assignment (single choice) |
| | | SPI4 | | | | Assignment (single choice) |
| | | SPI5 | | | | Assignment (single choice) |

| | | | | | | |
|---|---|---|---|---|---|---|
| interface *or* audio | I<br>SPI | SPI6 | | | | Assignment (single choice) |

# 4 References

- Device tree

Serial Peripheral Interface

Integrated Interchip Sound

Extended TrustZone Protection Controller

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

Cortex®

Open Portable Trusted Execution Environment

Linux® is a registered trademark of Linus Torvalds.

Advanced Linux sound architecture

Microprocessor Unit