



SPDIFRX internal peripheral



Contents

1. SPDIFRX internal peripheral	3
2. ALSA overview	9
3. Device tree	15
4. How to assign an internal peripheral to a runtime context	21
5. SPDIFRX Linux driver	27
6. SPDIFRX device tree configuration	33
7. STM32CubeMP1 architecture	39
8. STM32CubeMX	45
9. STM32MP15 resources	51
10. STM32MPU Embedded Software architecture overview	57
11. Soundcard configuration	63



A quality version of this page, approved on 11 February 2019, was based off this revision.

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	4
2 Peripheral overview	5
2.1 Features	5
2.2 Security support	5
3 Peripheral usage and associated software	6
3.1 Boot time	6
3.2 Runtime	6
3.2.1 Overview	6
3.2.2 Software frameworks	6
3.2.3 Peripheral configuration	6
3.2.3.1 Configuration in Cortex-A7 non-secure software	6
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	6
3.2.4 Peripheral assignment	6
4 How to go further	8
5 References	9



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

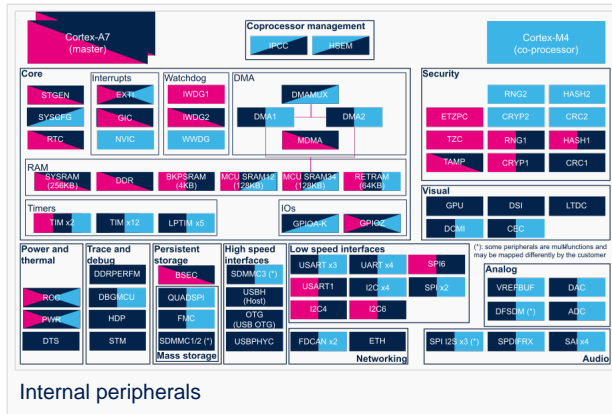
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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](#)
Stable: 05.01.2021 - 15:38 / Revision: 04.01.2021 - 17:18

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	10
2 Peripheral overview	11
2.1 Features	11
2.2 Security support	11
3 Peripheral usage and associated software	12
3.1 Boot time	12
3.2 Runtime	12
3.2.1 Overview	12
3.2.2 Software frameworks	12
3.2.3 Peripheral configuration	12
3.2.3.1 Configuration in Cortex-A7 non-secure software	12
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	12
3.2.4 Peripheral assignment	12
4 How to go further	14
5 References	15



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

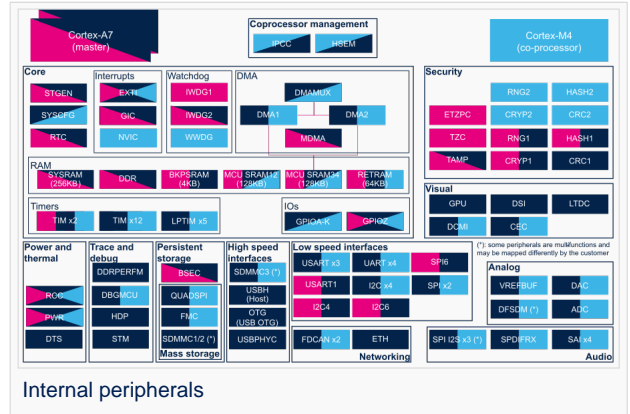
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Internal peripherals

Domain	Periphera	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
Stable: 05.11.2021 - 11:08 / Revision: 05.11.2021 - 11:05

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	16
2 Peripheral overview	17
2.1 Features	17
2.2 Security support	17
3 Peripheral usage and associated software	18
3.1 Boot time	18
3.2 Runtime	18
3.2.1 Overview	18
3.2.2 Software frameworks	18
3.2.3 Peripheral configuration	18
3.2.3.1 Configuration in Cortex-A7 non-secure software	18
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	18
3.2.4 Peripheral assignment	18
4 How to go further	20
5 References	21



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

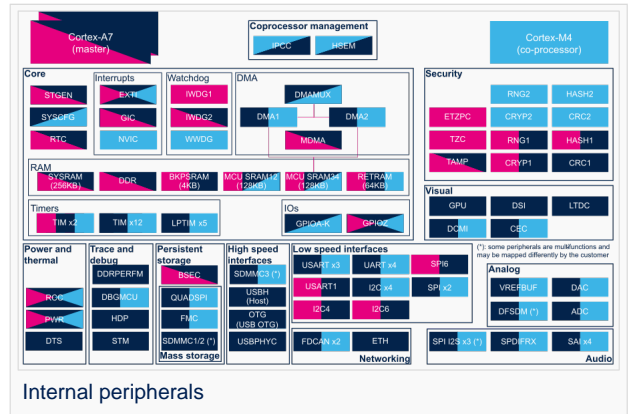
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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
Stable: 08.03.2021 - 16:13 / Revision: 16.02.2021 - 17:11

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	22
2 Peripheral overview	23
2.1 Features	23
2.2 Security support	23
3 Peripheral usage and associated software	24
3.1 Boot time	24
3.2 Runtime	24
3.2.1 Overview	24
3.2.2 Software frameworks	24
3.2.3 Peripheral configuration	24
3.2.3.1 Configuration in Cortex-A7 non-secure software	24
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	24
3.2.4 Peripheral assignment	24
4 How to go further	26
5 References	27



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

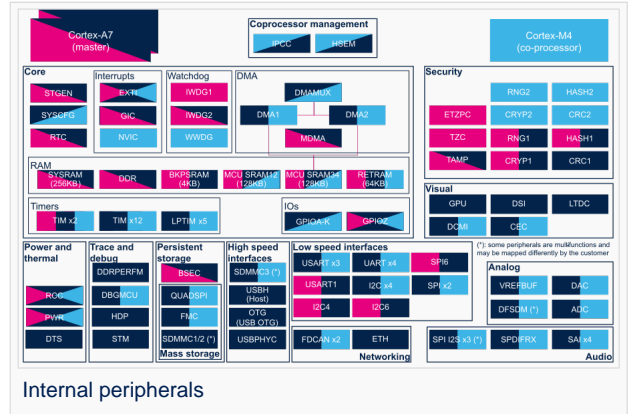
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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](#)
Stable: 22.01.2020 - 15:46 / Revision: 22.01.2020 - 10:02

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	28
2 Peripheral overview	29
2.1 Features	29
2.2 Security support	29
3 Peripheral usage and associated software	30
3.1 Boot time	30
3.2 Runtime	30
3.2.1 Overview	30
3.2.2 Software frameworks	30
3.2.3 Peripheral configuration	30
3.2.3.1 Configuration in Cortex-A7 non-secure software	30
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	30
3.2.4 Peripheral assignment	30
4 How to go further	32
5 References	33



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

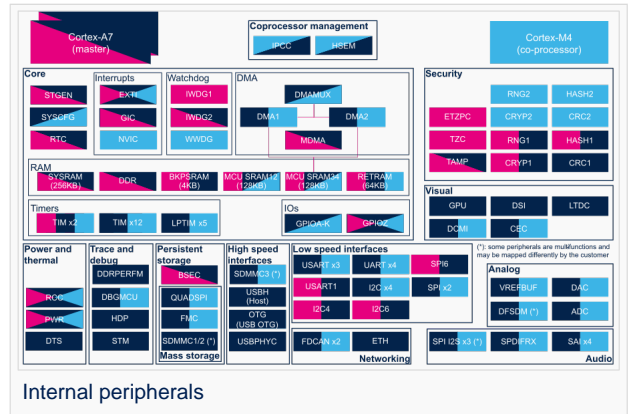
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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](#)
Stable: 03.03.2021 - 17:28 / Revision: 25.01.2021 - 08:45

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	34
2 Peripheral overview	35
2.1 Features	35
2.2 Security support	35
3 Peripheral usage and associated software	36
3.1 Boot time	36
3.2 Runtime	36
3.2.1 Overview	36
3.2.2 Software frameworks	36
3.2.3 Peripheral configuration	36
3.2.3.1 Configuration in Cortex-A7 non-secure software	36
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	36
3.2.4 Peripheral assignment	36
4 How to go further	38
5 References	39



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

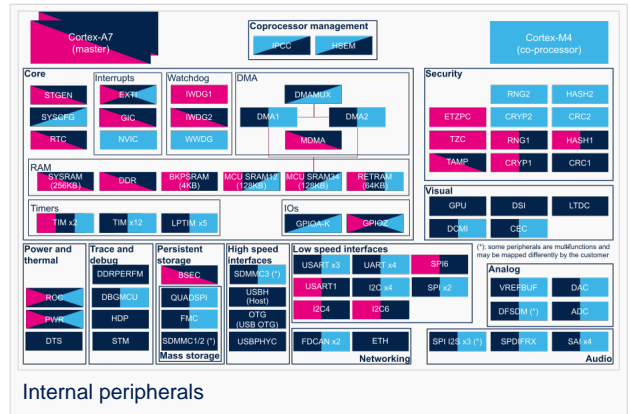
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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](#)
Stable: 31.03.2021 - 11:58 / Revision: 23.03.2021 - 14:07

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	40
2 Peripheral overview	41
2.1 Features	41
2.2 Security support	41
3 Peripheral usage and associated software	42
3.1 Boot time	42
3.2 Runtime	42
3.2.1 Overview	42
3.2.2 Software frameworks	42
3.2.3 Peripheral configuration	42
3.2.3.1 Configuration in Cortex-A7 non-secure software	42
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	42
3.2.4 Peripheral assignment	42
4 How to go further	44
5 References	45



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

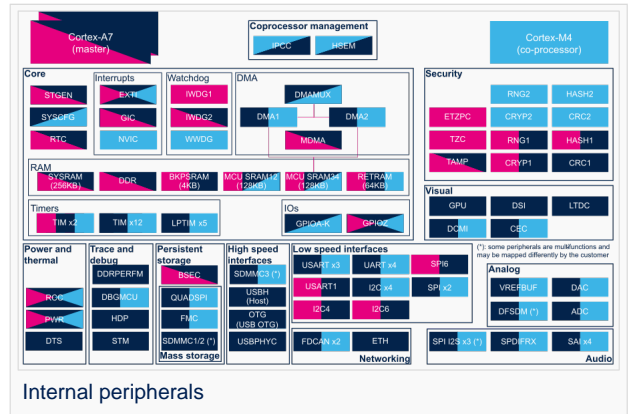
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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
Stable: 23.09.2020 - 13:22 / Revision: 12.06.2020 - 13:25

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	46
2 Peripheral overview	47
2.1 Features	47
2.2 Security support	47
3 Peripheral usage and associated software	48
3.1 Boot time	48
3.2 Runtime	48
3.2.1 Overview	48
3.2.2 Software frameworks	48
3.2.3 Peripheral configuration	48
3.2.3.1 Configuration in Cortex-A7 non-secure software	48
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	48
3.2.4 Peripheral assignment	48
4 How to go further	50
5 References	51



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

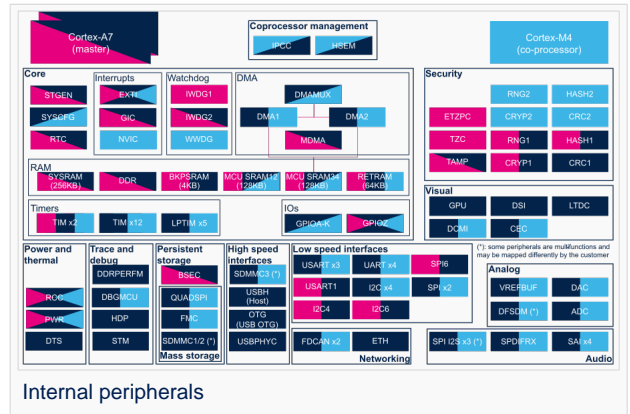
3.2.4 Peripheral assignment

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 possibilities might be described in STM32MP15 reference manuals.



Domain	Periphera	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
Stable: 17.11.2021 - 16:41 / Revision: 17.11.2021 - 10:47

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	52
2 Peripheral overview	53
2.1 Features	53
2.2 Security support	53
3 Peripheral usage and associated software	54
3.1 Boot time	54
3.2 Runtime	54
3.2.1 Overview	54
3.2.2 Software frameworks	54
3.2.3 Peripheral configuration	54
3.2.3.1 Configuration in Cortex-A7 non-secure software	54
3.2.3.2 Arm® Cortex®-M4 software configuration	54
3.2.4 Peripheral assignment	54
4 How to go further	56
5 References	57



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

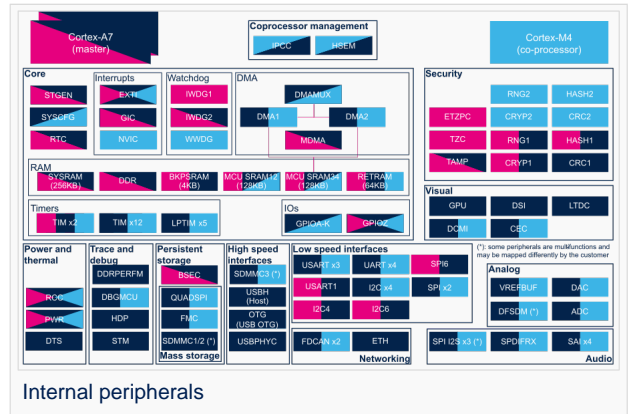
3.2.4 Peripheral assignment

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 possibilities might be described in STM32MP15 reference manuals.



Domain	Periphera	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
Stable: 26.03.2021 - 11:32 / Revision: 12.03.2021 - 11:07

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	58
2 Peripheral overview	59
2.1 Features	59
2.2 Security support	59
3 Peripheral usage and associated software	60
3.1 Boot time	60
3.2 Runtime	60
3.2.1 Overview	60
3.2.2 Software frameworks	60
3.2.3 Peripheral configuration	60
3.2.3.1 Configuration in Cortex-A7 non-secure software	60
3.2.3.2 Arm [®] Cortex [®] -M4 software configuration	60
3.2.4 Peripheral assignment	60
4 How to go further	62
5 References	63



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

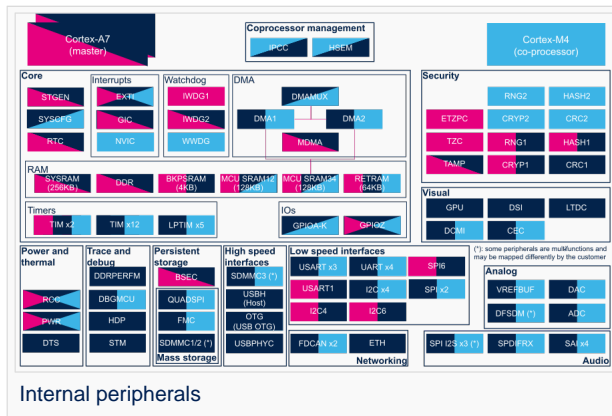
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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
- Stable: **Not stable** / Revision: 25.11.2021 - 09:40

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose	64
2 Peripheral overview	65
2.1 Features	65
2.2 Security support	65
3 Peripheral usage and associated software	66
3.1 Boot time	66
3.2 Runtime	66
3.2.1 Overview	66
3.2.2 Software frameworks	66
3.2.3 Peripheral configuration	66
3.2.3.1 Configuration in Cortex-A7 non-secure software	66
3.2.3.2 Arm® Cortex®-M4 software configuration	66
3.2.4 Peripheral assignment	66
4 How to go further	68
5 References	69



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 SPDIFRX 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 SPDIFRX is a **non secure** peripheral.



3 Peripheral usage and associated software

3.1 Boot time

The SPDIFRX 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 SPDIFRX driver

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

3.2.2 Software frameworks

Domain	Peripheral	Software components			Comment
OP-TEE	Linux	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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

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].

3.2.3.2 Arm[®] Cortex[®]-M4 software configuration

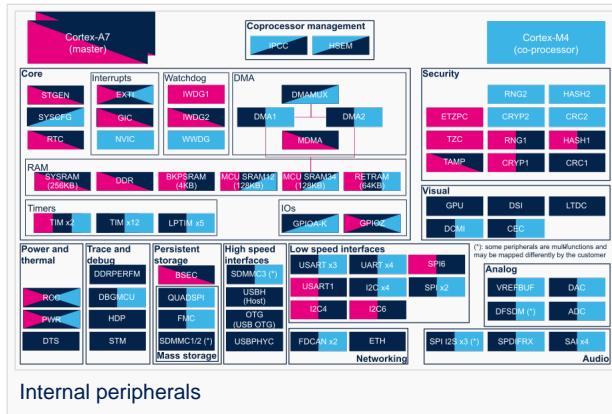
3.2.4 Peripheral assignment

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 possibilities might be described in *STM32MP15 reference manuals*.



Domain	Periphera	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