SPDIFRX internal peripheral

SPDIFRX internal peripheral

Stable: 04.02.2020 - 16:00 / Revision: 04.02.2020 - 15:52

Contents

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

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

Do	Peri	Software frameworks			Comment
mai n 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.

3.2.3.1 Configuration in Cortex-A7 non-secure software

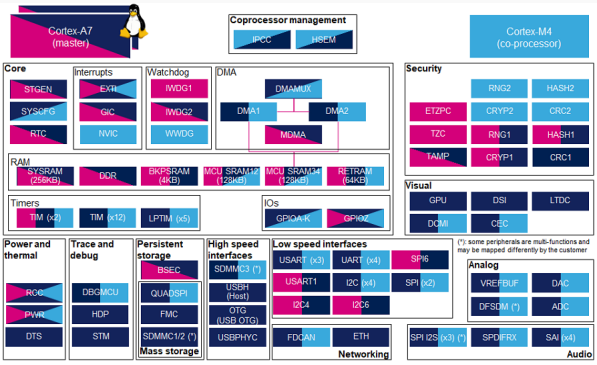
When the Arm® Cortex®-A7 core operates in non-secure access mode, the SPDIFRX is controlled by the Linux kernel framework. Refer to the SPDIFRX Linux driver to drive the SPDIFRX through Linux kernel ALSA framework. Refer to Soundcard configuration and SPDIFRX device tree configuration to configure the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



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	Peripheral	Runtime allocation				Comment
Instance	Cortex-A7 secure (OP-TEE)	Cortex-A7 non-secure (Linux)	Cortex-M4 (STM32Cube)			
Au	SPDIF					Assignment (single

Domain	Peripheral	Runtime allocation				Comment
	SPDIFRX					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