SPDIFRX device tree configuration

Contents

Stable: 03.03.2021 - 17:28 / Revision: 25.01.2021 - 08:45

Stable: 08.06.2021 - 17:29 / Revision: 28.01.2021 - 08:48

A quality version of this page, approved on *2 June 2020*, was based off this revision.

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- 2.02.1 arch/arm/boot/dts/stm32mp151.dtsi

Stable: 05.01.2021 - 15:38 / Revision: 04.01.2021 - 17:18

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- [2.02.1] arch/arm/boot/dts/stm32mp151.dtsi

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

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- 2.02.1 arch/arm/boot/dts/stm32mp151.dtsi

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

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- [2.02.1] arch/arm/boot/dts/stm32mp151.dtsi

Stable: 22.01.2020 - 15:46 / Revision: 22.01.2020 - 10:02

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- [2.02.1] arch/arm/boot/dts/stm32mp151.dtsi

Stable: 23.03.2021 - 16:53 / Revision: 23.03.2021 - 10:29

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- 2.02.1 arch/arm/boot/dts/stm32mp151.dtsi

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

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- 2.02.1 arch/arm/boot/dts/stm32mp151.dtsi

Stable: Not stable / Revision: 25.11.2021 - 09:40

Contents

1 Article purpose

This article explains how to configure the SPDIFRX internal peripheral when it is assigned to the **Linux® OS**. In that case, it is controlled by the ALSA framework.

The configuration is performed using the device tree mechanism that provides a hardware description of the SPDIFRX peripheral, used by the SPDFIRX linux driver.

If the peripheral is assigned to another execution context, refer to How to assign an internal peripheral to a runtime context article for guidelines on peripheral assignment and configuration.

2 DT bindings documentation

STM32 SPDIFRX device tree bindings [1] document describes all the required and optional configuration properties.

3 DT configuration

This hardware description is a combination of STM32 microprocessor [2] and board device tree files. See the Device tree for an explanation of the device tree file split.

STM32CubeMX can be used to generate the board device tree. Refer to How to configure the DT using STM32CubeMX for more details.

3.1 DT configuration (STM32 level)

The SPDIFRX node is declared in stm32mp151.dtsi[2]. It describes the hardware parameters such as register addresses, interrupt, clock and DMA. This set of properties may not vary for a given STM32MPU.

Warning

This device tree part is related to STM32 microprocessors. It must be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SPDIFRX is an audio peripheral, which can be used as a component of a soundcard through Linux® kernel ALSA framework. This part of the device tree allows the configuration of the SPDIFRX to implement a soundcard. Refer to soundcard configuration for examples of SPDIFRX configuration on various boards.

3.3 DT configuration examples

```
&spdifrx {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&spdifrx_pins_a>;
        pinctrl-1 = <&spdifrx_sleep_pins_a>
        /* Use spdifrx_pins_a instead of spdifrx_sleep_pins_a configuration,
        to allow IEC958 status bits capture, when an audio stream record is
        not active. Note that sleep state has to be defined, to allow pin state
        recovery from low power modes. */

        spdifrx_port: port {
                spdifrx_endpoint: endpoint {
                        remote-endpoint = <&spdif_in_endpoint>;
                };
        };
};
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

- Documentation/devicetree/bindings/sound/st,stm32-spdifrx.txt
- 2.02.1 arch/arm/boot/dts/stm32mp151.dtsi