DFSDM device tree configuration

Stable: 26.03.2021 - 10:29 | Revision: 30.03.2021 - 15:42

A quality version of this page, approved on *26 March 2021*, was based off this revision.

Contents

1 Article purpose

The purpose of this article is to explain how to configure the DFSDM internal peripheral when the peripheral is assigned to Linux®OS**, and in particular:**

- How to configure the DFSDM peripheral to enable filters and associated channels
- How to configure the board, e.g. serial interface input/output pins

The configuration is performed using the device tree mechanism.

It is used by the DFSDM Linux driver which registers the relevant information in IIO and ALSA frameworks.

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

The DFSDM maybe used as various functions: ADC and DMIC (for audio).

Each one is represented by a separate **compatible string**, documented in *STM32 DFSDM device tree bindings*[1].

The external analog frontend (e.g. sigma-delta modulator) is documented in *Device-Tree bindings for sigma delta modulator*[2]

3 DT configuration

This hardware description is a combination of STM32 microprocessor and board device tree files. See Device tree for more explanations on device tree file split.

The **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)

DFSDM nodes are declared in stm32mp151.dtsi[3].

- DT root node ('dfsdm') describes the ADC hardware block parameters such as registers area, clocks.
- DT child nodes ('dfsdm0', 'dfsdm1', ...) describe each filter independently: compatible string, interrupts, DMAs.

```
dfsdm: dfsdm@4400d000 {
        compatible = "st,stm32mp1-dfsdm";
        ...                                                                     /*
common resources in 'dfsdm' root node. */
        dfsdm0: filter@0 {
                compatible = "st,stm32-dfsdm-adc";                              /*
can either be st,stm32-dfsdm-(adc or dmic) */
                ...                                                             /*
private resources in 'dfsdm0' child node. */
        }
        dfsdm1: filter@1 {
                ...
        }
```

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

3.2 DT configuration (board level)

Follow the sequences described in the below chapters to configure and enable the DFSDM on your board.

3.2.1 Common resources for all DFSDM filters

Configure the 'dfsdm' **DT root node**:

- Enable the DT root node for the DFSDM, by setting **status = "okay"**.
- Configure the pins in use via pinctrl, by setting **pinctrl-0**, **pinctrl-1** and **pinctrl-names**.
- Configure the SPI clock output frequency, by setting **spi-max-frequency** (optional: only for SPI master mode).
- Configure the audio clock to be used, by setting **clocks** and **clock-names** (optional: to use more accurate clock for audio).

3.2.2 Private resources for each DFSDM filter

Configure the filter(s) **DT child node(s)**:

- Enable the DT child node(s) for the DFSDM filter(s) in use, by setting **status = "okay"**.
- Override the **compatible** string by setting **"st,stm32-dfsdm-dmic"** (optional: only for audio digital microphone).

- Enable the channel(s), by setting **st,adc-channels = <0 1 2...>**.
- Configure the channel(s) by setting **st,adc-channel-names**, **st,adc-channel-types** (e.g. SPI or manchester) and **st,adc-channel-clk-src** (e.g. external or internal).
- Configure the filter order, by setting **st,filter-order**.

3.2.3 Additional configuration for DFSDM ADC

The DFSDM ADC device has an external analog front-end, the *sigma delta modulator*.

Configure the external sigma delta modulator for each channel (optional, not needed for audio digital microphone):

- Add **your_sd_modulator** DT node in the board dts file (see the generic sd-modulator[2] example here after).
- Add **io-channels = <&your_sd_modulator>** to the DFSDM filter child node in order to assign it to the filter channel(s).

3.2.4 Additional configuration for DFSDM audio

Additional child nodes must be added for audio soundcard configuration.

3.3 DT configuration examples

The example below shows how to configure the DFSDM ADC channel 1, assigned to DFSDM filter 0:

- Declare pins used in pinctrl DT node (see Pinctrl device tree configuration):
  - Configure PB13 as DFSDM CLKOUT alternate function (AF3 by default, ANALOG for low-power mode).
  - Configure PC3 as DFSDM DATA1 alternate function (AF3 by default, ANALOG for low-power mode).

```
dfsdm_clkout_pins_a: dfsdm-clkout-pins-0 {
        pins {
                pinmux = <STM32_PINMUX('B', 13, AF3)>;     /* DFSDM_CKOUT */
                bias-disable;
                drive-push-pull;
                slew-rate = <1>;
        };
};

dfsdm_clkout_sleep_pins_a: dfsdm-clkout-sleep-pins-0 {
        pins {
                pinmux = <STM32_PINMUX('B', 13, ANALOG)>;  /* DFSDM_CKOUT */
        };
};
```

```
dfsdm_data1_pins_a: dfsdm-data1-pins-0 {
        pins {
                pinmux = <STM32_PINMUX('C', 3, AF3)>;      /* DFSDM_DATA1 */
        };
};

dfsdm_data1_sleep_pins_a: dfsdm-data1-sleep-pins-0 {
        pins {
                pinmux = <STM32_PINMUX('C', 3, ANALOG)>;   /* DFSDM_DATA1 */
        };
};
```

- Add sd-modulator[2] in the board dts file.

```
sd_adc1: adc-1 {
        compatible = "sd-modulator";
        #io-channel-cells = <0>;
};
```

- Configure and enable DFSDM, configure channel 1 to use SPI (rising edge), associate it to filter0.

```
&dfsdm {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&dfsdm_clkout_pins_a &dfsdm_data1_pins_a>;                 /*
default pins */
        pinctrl-1 = <&dfsdm_clkout_sleep_pins_a &dfsdm_data1_sleep_pins_a>;     /*
sleep pins for low-power mode */
        spi-max-frequency = <2048000>;                                          /*
desired maximum clock rate */
        status = "okay";
        dfsdm0: filter@0 {
                st,adc-channels = <1>;                                          /*
Assign channel 1 to this filter */
                st,adc-channel-names = "in1";                                   /*
Give it a name */
                st,adc-channel-types = "SPI_R";                                 /*
SPI data on rising edge */
                st,adc-channel-clk-src = "CLKOUT_F";                            /*
internal clock source used for conversion */
                io-channels = <&sd_adc1>;                                       /*
phandle to the external sd-modulator */
                st,filter-order = <1>;
                status = "okay";
        };
};
```

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

Please refer to the following links for additional information:

- Documentation/devicetree/bindings/iio/adc/st,stm32-dfsdm-adc.yaml , STM32 DFSDM device tree bindings
- 2.02.12.2 Documentation/devicetree/bindings/iio/adc/sigma-delta-modulator.yaml , Generic Device-Tree bindings for sigma delta modulator
- arch/arm/boot/dts/stm32mp151.dtsi , STM32MP151 device tree file

Operating System

Digital Filter for Sigma-Delta Modulator

Device Tree

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.

Digital microphone

Serial Peripheral Interface