QUADSPI device tree configuration

Contents

Stable: 01.02.2021 - 16:34 / Revision: 01.02.2021 - 16:34

Stable: 01.02.2021 - 10.04 | Revision: 01.02.2021 - 10.04

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the QUADSPI peripheral, used by the STM32 QUADSPI Linux driver and by the MTD framework.

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 QUADSPI device tree bindings are composed by:

- generic SPI-NOR / SPI-NAND Flash memory bindings [1].
- QUADSPI driver bindings [2].

3 DT configuration

This hardware description is a combination of the **STM32 microprocessor** device tree files (*.dtsi* extension) and **board** device tree files (*.dts* extension). 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 QUADSPI peripheral node is located in *stm32mp151.dtsi*[3] file.

```
   qspi: spi@58003000 {                                          Comments
       compatible = "st,stm32f469-qspi";
       reg = <0x58003000 0x1000>,                                --> Register location
             <0x70000000 0x10000000>;                            --> Memory mapping address
       reg-names = "qspi", "qspi_mm";
       interrupts = <GIC_SPI 92 IRQ_TYPE_LEVEL_HIGH>;            --> The interrupt number used
       dmas = <&mdma1 22 0x10 0x100002 0x0 0x0 0x0>,             --> DMA specifiers [4]
              <&mdma1 22 0x10 0x100008 0x0 0x0 0x0>;
       dma-names = "tx", "rx";
       clocks = <&rcc QSPI_K>;
       resets = <&rcc QSPI_R>;
       status = "disabled";
   };
```

This device tree part related to the STM32 should be kept as is, the customer should not modify it.

3.2 DT configuration (board level)

The QUADSPI peripheral may connect a maximum of 2 SPI-NOR Flash memories.

SPI-NOR Flash memory nodes [1] must be children of the QUADSPI peripheral node.

```
   &qspi {                                                       Comments
       pinctrl-names = "default", "sleep";                       --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&qspi_clk_pins_a &qspi_bk1_pins_a &qspi_bk2_pins_a>;
       pinctrl-1 = <&qspi_clk_sleep_pins_a &qspi_bk1_sleep_pins_a &qspi_bk2_sleep_pins_a>;
       reg = <0x58003000 0x1000>,
             <0x70000000 0x4000000>;                             --> Overwrite the memory map
to the Flash device size, avoid the waste of virtual memory that will not be used
       #address-cells = <1>;
       #size-cells = <0>;
       status = "okay";                                          --> Enable the node

       flash0: mx66l51235l@0 {
           compatible = "jdec,spi-nor";
           reg = <0>;                                            --> Chip select number
           spi-rx-bus-width = <4>;                               --> The bus width (number of
data wires used)
           spi-max-frequency = <108000000>;                      --> Maximum SPI clocking
```

```
speed of device in Hz
            #address-cells = <1>;
            #size-cells = <1>;
        };
    };
```

3.3 DT configuration example

The below example shows how to configure the QUADSPI peripheral when 1 SPI-NAND Flash and 1 SPI-NOR Flash memories are connected.

```
    &qspi {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&qspi_clk_pins_a &qspi_bk1_pins_a &qspi_bk2_pins_a>;
        pinctrl-1 = <&qspi_clk_sleep_pins_a &qspi_bk1_sleep_pins_a &qspi_bk2_sleep_pins_a>;
        reg = <0x58003000 0x1000>,
              <0x70000000 0x4000000>;
        #address-cells = <1>;
        #size-cells = <0>;
        status = "okay";

        flash0: mx66l51235l@0 {
            compatible = "jdec,spi-nor";
            reg = <0>;
            spi-rx-bus-width = <4>;
            spi-max-frequency = <108000000>;
            #address-cells = <1>;
            #size-cells = <1>;
        };

        flash1: mt29f2g01abagd@1 {
            compatible = "spi-nand";
            reg = <1>;
            spi-rx-bus-width = <4>;
            spi-tx-bus-width = <4>;
            spi-max-frequency = <133000000>;
            #address-cells = <1>;
            #size-cells = <1>;
        };
    };
```

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 full description:

- 1.01.1 Documentation/devicetree/bindings/spi/spi-controller.yaml
- Documentation/devicetree/bindings/spi/st,stm32-qspi.yaml
- arch/arm/boot/dts/stm32mp151.dtsi
- Documentation/devicetree/bindings/dma/st,stm32-mdma.yaml

Linux® is a registered trademark of Linus Torvalds.

Operating System

Memory Technology Device

Device Tree

Serial Peripheral Interface

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Generic Interrupt Controller

Direct Memory Access

Stable: 19.03.2021 - 08:52 / Revision: 19.03.2021 - 08:49

Invalid target: no reviewed revision corresponds to the given ID.

Return to Device tree.

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

Invalid target: no reviewed revision corresponds to the given ID.

Return to How to assign an internal peripheral to a runtime context.

Stable: 05.11.2020 - 08:23 / Revision: 21.10.2020 - 11:37

Invalid target: no reviewed revision corresponds to the given ID.

Return to MTD overview.

Stable: 11.06.2020 - 09:00 / Revision: 10.06.2020 - 15:35

Invalid target: no reviewed revision corresponds to the given ID.

Return to Pinctrl device tree configuration.

Stable: 09.10.2019 - 12:44 / Revision: 16.09.2019 - 12:51

Invalid target: no reviewed revision corresponds to the given ID.

Return to QUADSPI internal peripheral.

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

Invalid target: no reviewed revision corresponds to the given ID.

Return to STM32CubeMX.