



DMA device tree configuration



Contents

1. DMA device tree configuration	3
2. DMA internal peripheral	5
3. Dmaengine overview	5
4. Device tree	5
5. STM32CubeMX	6



DMA device tree configuration

Stable: 06.02.2020 - 14:11 / Revision: 06.02.2020 - 13:58

Template:ArticleMainWriter

This article is based on model **Peripheral or framework device tree configuration model**. It must follow it as much as possible.

Template:ArticleApprovedVersion

Contents

1 Article purpose	3
2 DT bindings documentation	3
3 DT configuration	4
3.1 DT configuration (STM32 level)	4
3.2 DT configuration (board level)	4
4 How to configure the DT using STM32CubeMX	5
5 References	5

1 Article purpose

This article explains how to configure the DMA internal peripheral when it is assigned to the Linux[®]OS. In that case, it is controlled by the Dmaengine overview.

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

The hardware description is a combination of:

- STM32 DMA peripheral
- and STM32 DMA client

2 DT bindings documentation

Complete device tree bindings can be found at this location: ^[1].

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)

At device level, the DMA declared is declared as follows:

```

dma2: dma@48001000 {
    compatible = "st,stm32-dma";
    reg = <0x48001000 0x400>;
    interrupts = <GIC_SPI 56 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 57 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 58 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 59 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 60 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 68 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 69 IRQ_TYPE_LEVEL_HIGH>,
                <GIC_SPI 70 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&rcc DMA2>;
    #dma-cells = <4>;
    st,mem2mem;
    dma-requests = <8>;
    dmas = <&mdma1 8 0x11 0x1200000a 0x48001008 0x00000020 1>,
          <&mdma1 9 0x11 0x1200000a 0x48001008 0x00000800 1>,
          <&mdma1 10 0x11 0x1200000a 0x48001008 0x00200000 1>,
          <&mdma1 11 0x11 0x1200000a 0x48001008 0x08000000 1>,
          <&mdma1 12 0x11 0x1200000a 0x4800100c 0x00000020 1>,
          <&mdma1 13 0x11 0x1200000a 0x4800100c 0x00000800 1>,
          <&mdma1 14 0x11 0x1200000a 0x4800100c 0x00200000 1>,
          <&mdma1 15 0x11 0x1200000a 0x4800100c 0x08000000 1>;
    dma-names = "ch0", "ch1", "ch2", "ch3", "ch4", "ch5", "ch6", "ch7";
};

```



Template:DarkBlue

3.2 DT configuration (board level)

Do not change the DMA device tree at board level.



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/dma/stm32-dma.txt](#)

Operating System

Direct Memory Access

Device Tree

Generic Interrupt Controller

Serial Peripheral Interface

Permission error

Stable: 11.02.2019 - 11:21 / Revision: 18.01.2019 - 16:05

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups [ST_editors](#), [ST_readers](#), [Selected_editors](#), [sysop](#), [reviewer](#)

Permission error

Stable: 15.10.2019 - 14:06 / Revision: 15.10.2019 - 14:05

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups [ST_editors](#), [ST_readers](#), [Selected_editors](#), [sysop](#), [reviewer](#)

Permission error

Stable: 04.02.2020 - 07:47 / Revision: 04.02.2020 - 07:34



You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer

Permission error

Stable: 31.01.2020 - 13:04 / Revision: 31.01.2020 - 13:02

You do not have permission to read this page, for the following reason:

The action "Read pages" for the draft version of this page is only available for the groups ST_editors, ST_readers, Selected_editors, sysop, reviewer