HASH device tree configuration

Contents

Stable: 13.05.2020 - 08:37 / Revision: 13.05.2020 - 08:36

Stable: 18.06.2020 - 00.01 - Revision: 18.06.2020 - 00.00

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the HASH peripheral, used by the STM32 HASH 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

The HASH is represented by the STM32 HASH device tree bindings[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)

The HASH node is declared in stm32mp151.dtsi[2]. It describes the hardware register address, clock, interrupt, reset and dma.

```
 hash1: hash@54002000 {                                                    Comments
         compatible = "st,stm32f756-hash";
         reg = <0x54002000 0x400>;                                          -->
Register location and length
         interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>;                     --> The
interrupt number used
         clocks = <&scmi0_clk CK_SCMI0_HASH1>;
         resets = <&scmi0_reset RST_SCMI0_HASH1>;
         dmas = <&mdma1 31 0x10 0x1000A02 0x0 0x0 0x0>;                     --> DMA
specifiers[3]
         dma-names = "in";
         dma-maxburst = <2>;
         status = "disabled";
 };
```

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)

This part is used to enable the HASH used on a board which is done by setting the **status** property to **okay**.

3.3 DT configuration examples

```
&hash1 {
        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:

- Device tree bindings
- STM32MP151 device tree file
- Documentation/devicetree/bindings/dma/stm32-mdma.txt , STM32 MDMA controller

Linux® is a registered trademark of Linus Torvalds.

Operating System

Device Tree

Generic Interrupt Controller

Serial Peripheral Interface

Direct Memory Access

Stable: 19.10.2020 - 09:54 / Revision: 19.10.2020 - 09:52

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

Return to Crypto API overview.

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: 12.02.2020 - 16:46 / Revision: 12.02.2020 - 16:44

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

Return to HASH internal peripheral.

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

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

Return to STM32CubeMX.