



HASH device tree configuration



Contents

1. HASH device tree configuration	3
2. Crypto API overview	8
3. Device tree	13
4. HASH internal peripheral	18
5. How to assign an internal peripheral to a runtime context	23
6. STM32CubeMX	28



A quality version of this page, approved on 13 May 2020, was based off this revision.

Contents

1 Article purpose	4
2 DT bindings documentation	5
3 DT configuration	6
3.1 DT configuration (STM32 level)	6
3.2 DT configuration (board level)	6
3.3 DT configuration examples	6
4 How to configure the DT using STM32CubeMX	7
5 References	8



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.

<pre>hash1: hash@54002000 { compatible = "st,stm32f756-hash"; reg = <0x54002000 0x400>; interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>; clocks = <&scmi0_clk CK_SCMI0_HASH1>; resets = <&scmi0_reset RST_SCMI0_HASH1>; dmas = <&mdma1 3I 0x10 0x1000A02 0x0 0x0 0x0>; dma-names = "in"; dma-maxburst = <2>; status = "disabled"; };</pre>	<p>Comments</p> <p>--></p> <p>--> The</p> <p>--> DMA</p>
--	--

Register location and length

interrupt number used

specifiers^[3]



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 7 Revision: 19.10.2020 - 09:52

Contents

1 Article purpose	9
2 DT bindings documentation	10
3 DT configuration	11
3.1 DT configuration (STM32 level)	11
3.2 DT configuration (board level)	11
3.3 DT configuration examples	11
4 How to configure the DT using STM32CubeMX	12
5 References	13



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.

<pre>hash1: hash@54002000 { compatible = "st,stm32f756-hash"; reg = <0x54002000 0x400>; interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>; clocks = <&scmi0_clk CK_SCMI0_HASH1>; resets = <&scmi0_reset RST_SCMI0_HASH1>; dmas = <&mdma1 3I 0x10 0x1000A02 0x0 0x0 0x0>; dma-names = "in"; dma-maxburst = <2>; status = "disabled"; };</pre>	<p>Comments</p> <p>--></p> <p>--> The</p> <p>--> DMA</p>
--	--

Register location and length

interrupt number used

specifiers^[3]



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: **Not stable** / Revision: 29.09.2021 - 15:37

Contents

1 Article purpose	14
2 DT bindings documentation	15
3 DT configuration	16
3.1 DT configuration (STM32 level)	16
3.2 DT configuration (board level)	16
3.3 DT configuration examples	16
4 How to configure the DT using STM32CubeMX	17
5 References	18



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.

<pre>hash1: hash@54002000 { compatible = "st,stm32f756-hash"; reg = <0x54002000 0x400>; interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>; clocks = <&scmi0_clk CK_SCMI0_HASH1>; resets = <&scmi0_reset RST_SCMI0_HASH1>; dmas = <&mdma1 3I 0x10 0x1000A02 0x0 0x0 0x0>; dma-names = "in"; dma-maxburst = <2>; status = "disabled"; };</pre>	<p>Comments</p> <p>--></p> <p>--> The</p> <p>--> DMA</p>
--	--

Register location and length

interrupt number used

specifiers^[3]



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: 12.02.2020 - 16:48 / Revision: 12.02.2020 - 16:44

Contents

1 Article purpose	19
2 DT bindings documentation	20
3 DT configuration	21
3.1 DT configuration (STM32 level)	21
3.2 DT configuration (board level)	21
3.3 DT configuration examples	21
4 How to configure the DT using STM32CubeMX	22
5 References	23



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.

<pre>hash1: hash@54002000 { compatible = "st,stm32f756-hash"; reg = <0x54002000 0x400>; interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>; clocks = <&scmi0_clk CK_SCMI0_HASH1>; resets = <&scmi0_reset RST_SCMI0_HASH1>; dmas = <&mdma1 3I 0x10 0x1000A02 0x0 0x0 0x0>; dma-names = "in"; dma-maxburst = <2>; status = "disabled"; };</pre>	<p>Comments</p> <p>--></p> <p>--> The</p> <p>--> DMA</p>
--	--

Register location and length

interrupt number used

specifiers^[3]



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: 08.03.2021 - 16:137 Revision: 16.02.2021 - 17:11

Contents

1 Article purpose	24
2 DT bindings documentation	25
3 DT configuration	26
3.1 DT configuration (STM32 level)	26
3.2 DT configuration (board level)	26
3.3 DT configuration examples	26
4 How to configure the DT using STM32CubeMX	27
5 References	28



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.

<pre>hash1: hash@54002000 { compatible = "st,stm32f756-hash"; reg = <0x54002000 0x400>; interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>; clocks = <&scmi0_clk CK_SCMI0_HASH1>; resets = <&scmi0_reset RST_SCMI0_HASH1>; dmas = <&mdma1 31 0x10 0x1000A02 0x0 0x0 0x0>; dma-names = "in"; dma-maxburst = <2>; status = "disabled"; };</pre>	<p>Comments</p> <p>--></p> <p>--> The</p> <p>--> DMA</p>
--	--

Register location and length

interrupt number used

specifiers^[3]



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: 23.09.2020 - 13:22.7 Revision: 12.06.2020 - 13:25

Contents

1 Article purpose	29
2 DT bindings documentation	30
3 DT configuration	31
3.1 DT configuration (STM32 level)	31
3.2 DT configuration (board level)	31
3.3 DT configuration examples	31
4 How to configure the DT using STM32CubeMX	32
5 References	33



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.

<pre>hash1: hash@54002000 { compatible = "st,stm32f756-hash"; reg = <0x54002000 0x400>; interrupts = <GIC_SPI 80 IRQ_TYPE_LEVEL_HIGH>; clocks = <&scmi0_clk CK_SCMI0_HASH1>; resets = <&scmi0_reset RST_SCMI0_HASH1>; dmas = <&mdma1 31 0x10 0x1000A02 0x0 0x0 0x0>; dma-names = "in"; dma-maxburst = <2>; status = "disabled"; };</pre>	<p>Comments</p> <p>--></p> <p>--> The</p> <p>--> DMA</p>
--	--

Register location and length

interrupt number used

specifiers^[3]



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