ETZPC device tree configuration

Contents

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

1 Article purpose

This article explains how to configure the ETZPC internal peripheral.

The configuration is performed using the device tree mechanism that provides a hardware description of the ETZPC peripheral, used by TF-A or OP-TEE.

2 DT bindings documentation

The ETZPC device tree bindings [1] describe all the required and optional properties.

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 ETZPC node is in the stm32mp151.dtsi [2] file. It describes the hardware register address, clock and status.

```
etzpc: etzpc@5C007000 {
        compatible = "st,stm32-etzpc";
        reg = <0x5C007000 0x400>;
        clocks = <&rcc TZPC>;
        status = "disabled";
        secure-status = "okay";
};
```

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)

The ETZPC node in the board dedicated device tree file is used to configure the status of securable peripherals. The "st, decprot" property must only contain the list of peripherals for which the user wants a different status than the one configured by default in the ETZPC. Refer to the ETZPC chapter of the reference manual [3] for more details.

To fill the "st,decprot" property, a DECPROT helper macro is provided. Its definition is in a dedicated header file [4]. It includes three parameters: the peripheral ID in the ETZPC list of peripherals, the domain, and a lock status. Additional information on these parameters can be found in the device tree bindings documentation [1].

3.3 DT configuration examples

Below an example of peripheral configuration:

```
&etzpc {
        st,decprot = <
                DECPROT(STM32MP1_ETZPC_USART1_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
                DECPROT(STM32MP1_ETZPC_SPI6_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
                DECPROT(STM32MP1_ETZPC_I2C4_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
                DECPROT(STM32MP1_ETZPC_I2C6_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
                DECPROT(STM32MP1_ETZPC_RNG1_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
                DECPROT(STM32MP1_ETZPC_HASH1_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
                DECPROT(STM32MP1_ETZPC_CRYP1_ID, DECPROT_NS_RW, DECPROT_UNLOCK)
        >;
};
```

By default these peripherals are secure only. They can then be configured to be used by the non-secure world in read and write modes.

Below another example of peripheral assignment (here the RNG2) to the MCU. By default this peripheral is assigned to the MPU non-secure world.

```
&etzpc {
        st,decprot = <
                DECPROT(STM32MP1_ETZPC_RNG2_ID, DECPROT_MCU_ISOLATION, DECPROT_UNLOCK)
        >;
};
```

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:

- 1.01.1 docs/devicetree/bindings/soc/st,stm32-etzpc.txt
- fdts/stm32mp151.dtsi
- STM32MP15 reference manuals
- include/dt-bindings/soc/st,stm32-etzpc.h

Extended TrustZone Protection Controller

Device Tree

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

Microprocessor Unit