Reset device tree configuration

A quality version of this page, accepted on *17 July 2020*, was based off this revision.

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the RCC peripheral used by the reset-stm32mp1 Linux driver and by the Reset framework.

When RCC TZEN or MCKPROT security hardening is enabled, Linux RCC reset driver is not the only reset controller provider. The SCMI server embedded in the secure firmware is also a reset controller provider through Linux SCMI reset device driver. See more information in section STM32MP15 SCMI reset domains below.

2 Reset controller providers

There are 2 reset controller providers in STM32MP1. Each are represented by node(s) in the device tree description. Their node must define a value for specifier *#reset-cells*.

- STM32MP1 RCC reset controllers, most of the system reset controllers actually.
- SCMI reset domains are reset controllers registered by the SCMI drivers.

The STM32MP1 uses SCMI reset domains to abstract RCC secure reset controllers.

This article describes the device configuration where RCC TZEN security hardening is enabled. Specificities to explore when RCC TZEN hardening is disabled are discussed in non-secure RCC configuration article.

3 DT bindings documentation

The Reset device tree bindings are composed of:

- generic DT bindings[1] used by the Reset framework.
- vendor Reset DT bindings[2] used by the reset-stm32mp1 driver: this binding document explains how to write device tree files for reset.
- generic SCMI DT bindings[3] used by the SCMI reset domain protocol support.

4 DT configuration

4.1 DT configuration (STM32 level)

The device tree description of the STM32 SoC includes reset controllers exposed by RCC reset controller device driver and SCMI reset domain device driver.

4.1.1 STM32MP1 RCC Reset node

The device tree defines the RCC reset controllers device as a node with *compatible = "st,stm32mp1-rcc"* or *"st,stm32mp1-rcc-secure"* node.

- *"st,stm32mp1-rcc-secure"* complies with configuration where RCC TZEN secure hardening is enabled.
- *"st,stm32mp1-rcc"* complies with configuration where RCC TZEN secure hardening is disabled.

The node defines *#reset-cells = <1>;*.
STM32MP1 RCC Reset controllers are identified by a single 32-bit ID. Valid values for the IDs are defined in STM32MP1 Reset DT bindings[4].

The STM32MP1 RCC Reset node is the same node as the STM32MP1 RCC Clock (they share same hardware IP) and is located in the *stm32mp151.dtsi*[5], hence one can also see *#clock-cells = <1>;* in the node. See the Device tree for further explanation.

```
rcc: rcc@50000000 {
       compatible = "st,stm32mp1-rcc-secure", "st,stm32mp1-rcc", "syscon";
        #clock-cells = <1>;
        #reset-cells = <1>;
        reg = <0x50000000 0x1000>;
        ...
};
```

This device tree part is related to STM32MP1 microprocessors. It must be kept as-is, without being modified by the end-user.

4.1.2 SCMI Reset Domain node

The device tree defines SCMI reset domains using *compatible = "arm,scmi"* nodes with subnodes specifying protocol@16 (*reg = <0x16>*) together with specifier '#reset-cells = 1. *The reset consumer uses node phandle (*scmi0_reset *in example below) together with a reset ID based macros* RST_SCMIx_* *defined in STM32MP1 Reset DT bindings[6] to identify a SCMI reset domain related to an agent interface.*

```
scmi-0 {
        compatible = "arm,scmi";
        #address-cells = <1>;
        #size-cells = <0>;

        scmi0_reset: protocol@16 {
               reg = <0x16>;
               #reset-cells = <1>;
       };
};
```

This device tree part is related to STM32MP1 microprocessors. It must be kept as-is, without being modified by the end-user.

4.2 DT configuration (board level)

If a Linux driver needs a reset signal, it should be declared in its DT node as shown below:
resets = <phandle> : List of phandle and reset specifier pairs, one pair for each reset signal that affects the device, or that the device manages.

- Example:

```
i2c2: i2c@40013000 {
        compatible = "st,stm32f7-i2c";
        reg = <0x40013000 0x400>;
        clocks = <&rcc I2C2_K>;
        resets = <&rcc I2C2_R>;
       ...
};

i2c4: i2c@5c002000 {
        compatible = "st,stm32f7-i2c";
        reg = <0x5c002000 0x400>;
        clocks = <&scmi0_clk CK_SCMI0_I2C4>;
        resets = <&scmi0_reset RST_SCMI0_I2C4>;
       ...
};
```

5 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MP1 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, which are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

6 References

Please refer to the following links for additional information:

- Documentation/devicetree/bindings/reset/reset.txt , Reset device tree bindings
- Documentation/devicetree/bindings/reset/st%2Cstm32mp1-rcc.txt , STM32MP1 Reset device tree bindings
- Documentation/devicetree/bindings/arm/arm,scmi.txt SCMI DT bindings
- Documentation/devicetree/bindings/reset/st%2Cstm32mp1-rcc.txt STM32MP1 Reset DT bindings
- stm32mp151.dtsi STM32MP151 device tree file
- Documentation/devicetree/bindings/reset/st%2Cstm32mp1-rcc.txt STM32MP1 Reset DT bindings

Operating System

Reset and Clock Control

System control and management interface

Device Tree