Clock device tree configuration - Bootloader specific

Stable: 2 Rev: 1.0 Revision: 10.02.2021 – 10:11

Contents

1 Article purpose

This article describes the specific **RCC** internal peripheral configuration done by the first stage bootloader:

- TF-A for the Trusted boot chain
- U-Boot SPL DDR interactive mode for the DDR tuning tool

Regarding OP-TEE when it is embedded in the device, OP-TEE OS is booted by TF-A BL2, it is booted by TF-A BL2 bootstage. OP-TEE relies on TF-A BL2 bootstage for the RCC clock tree initial configuration. This article explicitily mentions OP-TEE when in information applies to OP-TEE secure world configuration.

This article explains how to configure the clock tree in the RCC at boot time.
You can then refer to the clock device tree configuration article to understand how to derive each internal peripheral clock tree in Linux®OS from the RCC clock tree.

The configuration is performed using the device tree mechanism that provides a hardware description of the RCC peripheral.

This clock tree is only used in the device tree of the boot chain FSBL; so in the TF-A device tree for OpenSTLinux official delivery (or in SPL only for the DDR tuning tool).

Even if the clock tree information is also present in the U-Boot device tree, it is not used during boot by this SSBL.

2 DT bindings documentation

The bootloader clock device tree bindings correspond to the vendor clock DT bindings used by the clk-stm32mp1 driver of the FSBL (TF-A or U-Boot SPL for DDR interactive mode), it is based on:

- binding described in Clock_device_tree_configuration
- bootloader specific properties described in #DT configuration

This binding document explains how to write the device tree files for clocks on the bootloader side:

- TF-A: *tf-a/docs/devicetree/bindings/clock/st,stm32mp1-rcc.txt*"[1]
- U-Boot SPL for DDR interactive mode: *doc/device-tree-bindings/clock/st,stm32mp1.txt*[2]

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 STM32MP1 clock nodes are located in *stm32mp151.dtsi*[3] (see Device tree for more explanations):

- fixed-clock defined in clock node
- RCC node for #STM32MP1 clock node: clock generation and distribution.

```
/ {
...
        clocks {
                clk_hse: clk-hse {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <24000000>;
                };

...
          };
...
        soc {
...
                rcc: rcc@50000000 {
                        compatible = "st,stm32mp1-rcc", "syscon";
                        reg = <0x50000000 0x1000>;
                        #clock-cells = <1>;
                        #reset-cells = <1>;
                        interrupts = <GIC_SPI 5 IRQ_TYPE_LEVEL_HIGH>;
                };
...

        };

};
```

Please refer to clock device tree configuration for the bindings common with Linux® kernel.

3.2 DT configuration (board level)

3.2.1 Clock node

Note: this section applies to OP-TEE that gets input clocks frequency value from the device tree description it boots upon.

The clock tree is also based on five fixed clocks in the clock node. They are used to define the state of associated STM32MP1 oscillators:

- clk-lsi
- clk-lse

- clk-hsi
- clk-hse
- clk-csi

Please refer to clock device tree configuration for detailed information.

At boot time, the clock tree initialization performs the following tasks:

- enabling of the oscillators present in the device tree and not disabled (node with status="disabled"),
- disabling of the HSI oscillator if the node is absent or disabled (HSI is always activated by the ROM code).

3.2.1.1 *Optional properties for "clk-lse" and "clk-hse" external oscillators*

For external oscillator HSE and LSE, the default clock configuration is an external crystal/ceramic resonator.

Four optional fields are supported:

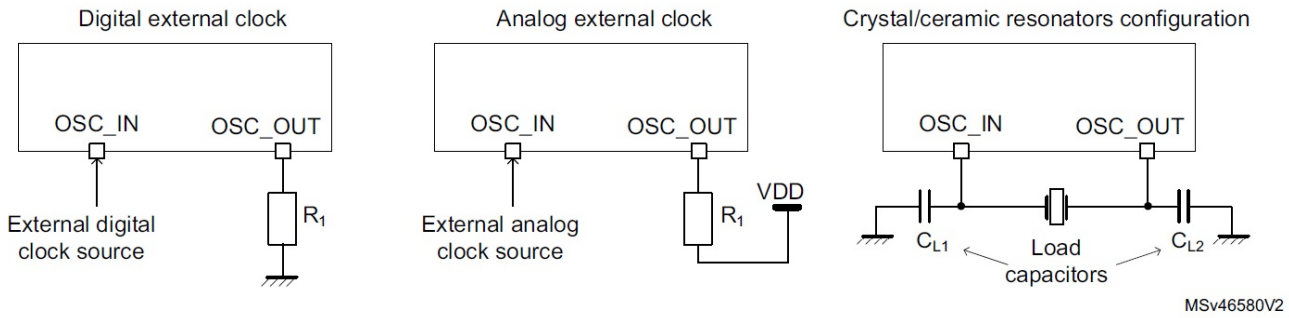
- "st,bypass" configures the external analog clock source (set HSEBYP, LSEBYP),
- "st,digbypass" configures the external digital clock source (set DIGBYP and HSEBYP, LSEBYP),
- "st,css" activates the clock security system (HSECSSON, LSECSSON),
- "st,drive" (LSE only) contains the value of the drive for the oscillator (see LSEDRV_ defined in the file *stm32mp1-clksrc.h*[4]).

3.2.1.2 *DT configuration for HSE*

The HSE can accept an external crystal/ceramic or external clock source on OSC_IN, digital or analog : the user needs to select the correct frequency and the correct configuration in the device tree, corresponding to the hardware setup.

All the ST boards are using a digital external clock configuration (so device tree with = st,digbypass).

For example with the same 24MHz frequency, we have 3 configurations:



- Digital external clock = st,digbypass

```
/ {
        clocks {
                clk_hse: clk-hse {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <24000000>;
                        st,digbypass;
                };
};
```

- Analog external clock = st,bypass

```
/ {
        clocks {
                clk_hse: clk-hse {
                        #clock-cells = <0>;
```

```
                                compatible = "fixed-clock";
                                clock-frequency = <24000000>;
                                st,bypass;
                        };
        };
```

- Crystal/ ceramic resonators configuration

```
        / {
                clocks {
                        clk_hse: clk-hse {
                                #clock-cells = <0>;
                                compatible = "fixed-clock";
                                clock-frequency = <24000000>;
                        };
        };
```

3.2.1.3 DT configuration for LSE

Below an example of LSE on board file with 32,768kHz crystal resonators, the drive set to medium high and with activated clock security system.

```
        / {
                clocks {
                        clk_lse: clk-lse {
                                #clock-cells = <0>;
                                compatible = "fixed-clock";
                                clock-frequency = <32768>;
                                st,css;
                                st,drive = <LSEDRV_MEDIUM_HIGH>;
                        };
        };
```

3.2.1.4 Optional property for "clk-hsi" internal oscillator

The HSI clock frequency is internally fixed to 64 MHz for the STM32MP15 devices.

In the device tree, clk-hsi is the clock after HSIDIV divider (more information on clk_hsi can be found in the RCC chapter in the reference manual).
As a result the frequency of this fixed clock is used to compute the expected HSIDIV for the clock tree initialization.

Below an example with HSIDIV = 1/1:

```
        / {
                clocks {
                        clk_hsi: clk-hsi {
                                #clock-cells = <0>;
                                compatible = "fixed-clock";
                                clock-frequency = <64000000>;
                        };
        };
```

Below an example with HSIDIV = 1/2:

```
        / {
                clocks {
                        clk_hsi: clk-hsi {
                                #clock-cells = <0>;
                                compatible = "fixed-clock";
                                clock-frequency = <32000000>;
                        };
        };
```

3.2.1.5 *Clock node example*

Note: this section applies to OP-TEE OS clock drivers.

An example of clocks node with:

- all oscillators switched on (HSE, HSI, LSE, LSI, CSI)
- HSI at 64MHZ (HSIDIV = 1/1)
- HSE using a digital external clock at 24MHz
- LSE using an external crystal at 32.768kHz (the typical frequency)

We highlight the customized parts:

```
/ {
        clocks {
                clk_hse: clk-hse {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <24000000>;
                        st,digbypass;
                };

                clk_hsi: clk-hsi {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <64000000>;
                };

                clk_lse: clk-lse {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <32768>;
                };

                clk_lsi: clk-lsi {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <32000>;
                };

                clk_csi: clk-csi {
                        #clock-cells = <0>;
                        compatible = "fixed-clock";
                        clock-frequency = <4000000>;
                };
        };
};
```

So the resulting board device tree, based on SoC device tree "stm32mp151.dtsi", is :

```
#include "stm32mp151.dtsi"
&clk_hse {
        clock-frequency = <24000000>;
        st,digbypass;
};

&clk_hsi {
        clock-frequency = <64000000>;
};

&clk_lse {
        clock-frequency = <32768>;
};
```

It is the configuration used by TF-A for STM32MP15 boards.

3.2.2 STM32MP1 clock node

Please refer to clock device tree configuration for information on how to specify the number of cells in a clock specifier.

The bootloader performs a global clock initialization, as described below. The information related to a given board can be found in the board specific device tree files listed in clock node.

The bootloader uses other properties for RCC node ("st,stm32mp1-rcc" compatible):

- secure-status: related to TZEN bit configuration in RCC security register that allows to restrict RCC and PWR registers write access
- st,clksrc: clock source configuration array
- st,clkdiv: clock divider configuration array
- st,pll: specific PLL configuration
- st,pkcs: peripheral kernel clock distribution configuration array.

All the available clocks are defined as preprocessor macros in *stm32mp1-clks.h*[5] and can be used in device tree sources.

Note: this section partially applies to OP-TEE OS clock drivers in that OP-TEE OS clock drivers consider only property *secure-status* over those listed above.

3.2.2.1 Defining clock source distribution with st,clksrc property

This property can be used to configure the clock distribution tree. When used, it must describe the whole distribution tree.

There are nine clock source selectors for the STM32MP15 devices. They must be configured in the following order: MPU, AXI, MCU, PLL12, PLL3, PLL4, RTC, MCO1, and MCO2.

The clock source configuration values are defined by the CLK_<NAME>_<SOURCE> macros located in *stm32mp1-clksrc.h*[4].

Example:

```
        st,clksrc = <
                CLK_MPU_PLL1P
                CLK_AXI_PLL2P
                CLK_MCU_PLL3P
                CLK_PLL12_HSE
                CLK_PLL3_HSE
                CLK_PLL4_HSE
                CLK_RTC_LSE
                CLK_MCO1_DISABLED
                CLK_MCO2_DISABLED
        >;
```

3.2.2.2 *Defining clock dividers with st,clkdiv property*

This property can be used to configure the value of the clock main dividers. When used, it must describe the whole clock divider tree.

There are 11 dividers values for the STM32MP15 devices. They must be configured in the following order: MPU, AXI, MCU, APB1, APB2, APB3, APB4, APB5, RTC, MCO1 and MCO2.

Each divider value uses the DIV coding defined in the RCC associated register, RCC_xxxDIVR. In most cases, this value is the following:

- 0x0: not divided
- 0x1: division by 2
- 0x2: division by 4
- 0x3: division by 8
- ...

Note that the coding differs for RTC MCO1 and MCO2:

- 0x0: not divided
- 0x1: division by 2
- 0x2: division by 3
- 0x3: division by 4
- ...

Example:

```
        st,clkdiv = <
                1 /*MPU*/
                0 /*AXI*/
                0 /*MCU*/
                1 /*APB1*/
                1 /*APB2*/
                1 /*APB3*/
                1 /*APB4*/
                2 /*APB5*/
                23 /*RTC*/
                0 /*MCO1*/
                0 /*MCO2*/
        >;
```

3.2.2.3 *Defining peripheral PLL frequencies with st,pll property*

This property can be used to configure PLL frequencies.

The PLL children nodes for PLL1 to PLL4 (see reference manual for details) are associated with an index from 0 to 3 (st,pll@0 to st,pll@3).

PLL2, PLL3 or PLL4 are off when their associated nodes are absent or deactivated.

The configuration of PLL1, the source clock of Cortex-A7 core, with st,pll@0 node, is optional as TF-A automatically selects the most suitable operating point for the platform (please refer to How to change the CPU frequency). The node st,pll@0 node should be absent; it is only used if you want to override the PLL1 properties computed by TF-A (clock spreading for example).

Below the available properties for each PLL node:

- cfg contains the PLL configuration parameters in the following order: DIVM, DIVN, DIVP, DIVQ, DIVR, output.

  DIVx values are defined as in RCC:

  - 0x0: bypass (division by 1)

- 0x1: division by 2
- 0x2: division by 3
- 0x3: division by 4
- ...

Output contains a bitfield for each output value (1:ON / 0:OFF)

- BIT(0) output P : DIVPEN
- BIT(1) output Q : DIVQEN
- BIT(2) output R : DIVREN

Note: PQR(p,q,r) macro can be used to build this value with p, q, r = 0 or 1.

- frac: fractional part of the multiplication factor (optional, when absent PLL is in integer mode).
- csg contains the clock spreading generator parameters (optional) in the following order: MOD_PER, INC_STEP and SSCG_MODE.

MOD_PER: modulation period adjustment

INC_STEP: modulation depth adjustment

SSCG_MODE: Spread spectrum clock generator mode, defined in *stm32mp1-clksrc.h*[4]:

- SSCG_MODE_CENTER_SPREAD = 0
- SSCG_MODE_DOWN_SPREAD = 1

Example:

```
        pll2: st,pll@1 {
                   compatible = "st,stm32mp1-pll";
                   reg = <1>;
                cfg = < 1 43 1 0 0 PQR(0,1,1) >;
                csg = < 10 20 1 >;
        };
        pll3: st,pll@2 {
                   compatible = "st,stm32mp1-pll";
                   reg = <2>;
                cfg = < 2 85 3 13 3 0 >;
                csg = < 10 20 SSCG_MODE_CENTER_SPREAD >;
        };
        pll4: st,pll@3 {
                   compatible = "st,stm32mp1-pll";
                   reg = <3>;
                cfg = < 2 78 4 7 9 3 >;
        };
```

3.2.2.4 *Defining peripheral kernel clock tree distribution with st,pkcs property*

This property can be used to configure the peripheral kernel clock selection.

It is a list of peripheral kernel clock source identifiers defined by the CLK_<KERNEL-CLOCK>_<PARENT-CLOCK> macros in the *stm32mp1-clksrc.h*[4] header file.

st,pkcs may not list all the kernel clocks. No specific order is required.

Example:

```
        st,pkcs = <
                CLK_STGEN_HSE
                CLK_CKPER_HSI
```

```
                CLK_USBPHY_PLL2P
                CLK_DSI_PLL2Q
                CLK_I2C46_HSI
                CLK_UART1_HSI
                CLK_UART24_HSI
        >;
```

3.2.2.5 *HSI and CSI clocks calibration*

Note: this section applies to OP-TEE OS clock calibration support.

The calibration is an optional feature that can be enabled from the device tree. It allows requesting the HSI or CSI clock calibration by several means:

- SiP SMC service
- Periodic calibration every X seconds
- Interrupt raised by the MCU

This feature requires that a hardware timer is assigned to the calibration sequence.

A dedicated interrupt must be defined using "mcu_sev" name to start a calibration on detection of an interrupt raised by the MCU.

- st,hsi-cal: used to enable HSI clock calibration feature.
- st,csi-cal; used to enable CSI clock calibration feature.
- st,cal-sec: used to enable periodic calibration at specified time intervals from the secure monitor. The time interval must be given in seconds. If not specified, a calibration is only processed for each incoming request.

Example:

```
        &rcc {
                st,hsi-cal;
                st,csi-cal;
                st,cal-sec = <15>;
                secure-interrupts = <GIC_SPI 144 IRQ_TYPE_LEVEL_HIGH>,
                                    <GIC_SPI 145 IRQ_TYPE_LEVEL_HIGH>;
                interrupt-names = "mcu_sev", "wakeup";
        };
```

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:

- docs/devicetree/bindings/clock/st,stm32mp1-rcc.txt TF-A clock binding information file
- doc/device-tree-bindings/clock/st,stm32mp1.txt U-Boot SPL for DDR interactive mode clock binding information file
- fdts/stm32mp151.dtsi (for TF-A), arch/arm/dts/stm32mp15-no-scmi.dtsi (for U-Boot SPL for DDR interactive mode): STM32MP151 device tree files
- 4.04.14.24.3 include/dt-bindings/clock/stm32mp1-clksrc.h (for TF-A), include/dt-bindings/clock/stm32mp1-clksrc.h (for U-Boot SPL for DDR interactive mode): STM32MP1 DT bindings clock source files
- include/dt-bindings/clock/stm32mp1-clks.h (for TF-A), include/dt-bindings/clock/stm32mp1-clks.h (for U-Boot SPL for DDR interactive mode): STM32MP1 DT bindings clock identifier files

Doubledata rate (memory domain)

Open Portable Trusted Execution Environment

Operating System

Trusted Firmware for Arm® Cortex®-A

Boot Loader stage 2

Reset and Clock Control

Linux® is a registered trademark of Linus Torvalds.

First Stage Boot Loader

Secondary Program Loader, *Also known as* ***U-Boot SPL***

Second Stage Boot Loader

Device Tree

Generic Interrupt Controller

Serial Peripheral Interface

High Speed Internal oscillator (STM32 clock source) or High Speed Synchronous Serial Interface (MIPI® Alliance standard)

Read Only Memory

High Speed External oscillator (STM32 clock source)

Low Speed External oscillator (STM32 clock source)

Low Speed Internal oscillator (STM32 clock source)

Multi Speed Internal oscillator (STM32 clock source)

Microprocessor Unit

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.)

Real Time Clock

Cortex®

System Time Generator

Display Serial Interface (MIPI® Alliance standard)

Silicon Provider

Secure Monitor Call