

Clock device tree configuration - Bootloader specific

Stable: 15.10.2019 - 17:29 / Revision: 15.10.2019 - 17:26

Contents

1 Article purpose	1
2 DT bindings documentation	1
3 DT configuration	2
3.1 DT configuration (STM32 level)	2
3.2 DT configuration (board level)	3
3.2.1 Clock node	3
3.2.1.1 Optional properties for "clk-lse" and "clk-hse" external oscillators	3
3.2.1.2 Optional property for "clk-hsi" internal oscillator	4
3.2.2 STM32MP1 clock node	4
3.2.2.1 Defining clock source distribution with st,clksrc property	5
3.2.2.2 Defining clock dividers with st,clkdiv property	5
3.2.2.3 Defining peripheral PLL frequencies with st,pll property	6
3.2.2.4 Defining peripheral kernel clock tree distribution with st,pkcs property	7
3.2.2.5 HSI and CSI clock calibration	8
4 How to configure the DT using STM32CubeMX	8
5 References	9

1 Article purpose

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



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.

2 DT bindings documentation

The bootloader clock device tree bindings are composed of the vendor clock DT bindings used by the clk-stm32mp1 [TF-A](#) and by the [U-Boot SPL](#) driver.

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

- **TF-A:** `<TF-A directory>/docs/devicetree/bindings/clock/st,stm32mp1-rcc.txt`
- **U-Boot SPL:** `doc/device-tree-bindings/clock/st,stm32mp1.txt`^[1]



Please note that the upstreaming of this **TF-A** file is in progress

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 `stm32mp157c.dtsi`^[2] (see [Device tree](#) for more explanations):

- fixed-clock define in [clock node](#)
- **RCC** node for **#STM32MP1 Clock node**: clock generation and distribution.

Please refer to [clock device tree configuration](#) for the bindings common with Linux[®] kernel.

```

/ {
...
    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>;
        };
...
    };
};

```

3.2 DT configuration (board level)

3.2.1 Clock node

The clock tree is also based on five fixed clocks in the clock node. They are used to define the state of associated ST32MP1 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,
- disabling of the HSI oscillator if the node is absent (always activated by bootROM).

This information is located in the following files:

- STM32MP157C-ED1 and STM32MP157C-EV1:
 - TF-A: *fdts/stm32mp157c-ed1.dts*^[3]
 - U-Boot SPL: *arch/arm/dts/stm32mp157c-ed1-u-boot.dtsi*^[4]
- STM32MP157X-DK1 and STM32MP157X-DK2:
 - TF-A: *<TF-A directory>/fdts/stm32mp157a-dk1.dts*.
 - U-Boot SPL: *<U-Boot directory>/arch/arm/dts/stm32mp157a-dk1-u-boot.dtsi*.



Please note that the upstreaming of these two TF-A / U-Boot SPL files is in progress

Optional properties are described below.

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

Four optional fields are supported:

- "st,bypass" configures the oscillator bypass mode (HSEBYP, LSEBYP),
- "st,digbypass" configures the bypass mode as full-swing digital signal (DIGBYP),
- "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*^[5]).

Below an example of board file:

```

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

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

```

3.2.1.2 Optional property for "clk-hsi" internal oscillator

The HSI clock frequency is internally fixed to 64 MHz for 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>;
        };
    };
};

```

3.2.2 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:

- `secure-status`: related to TZEN bit configuration in RCC_TZCR register that allows to restrict RCC register 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*^[6] and can be used in device tree sources.

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*^[5].

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.

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). PLLx is off when the associated node is absent.

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*^[5]:

- SSCG_MODE_CENTER_SPREAD = 0
- SSCG_MODE_DOWN_SPREAD = 1

Example:

```
st,pll@0 {
    cfg = < 1 53 0 0 0 1 >;
    frac = < 0x810 >;
};
st,pll@1 {
    cfg = < 1 43 1 0 0 PQR(0,1,1) >;
    csg = < 10 20 1 >;
};
st,pll@2 {
    cfg = < 2 85 3 13 3 0 >;
    csg = < 10 20 SSCG_MODE_CENTER_SPREAD >;
};
st,pll@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*^[5] 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 clock calibration

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 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:

1. ↑ [doc/device-tree-bindings/clock/st,stm32mp1.txt](#) U-Boot SPL clock binding information file
2. ↑ [fdts/stm32mp157c.dtsi](#) (for TF-A), [arch/arm/dts/stm32mp157c.dtsi](#) (for U-Boot SPL): STM32MP157C device tree files
3. ↑ [fdts/stm32mp157c-ed1.dts](#) STM32MP157C-ED1 device tree file for TF-A
4. ↑ [arch/arm/dts/stm32mp157c-ed1-u-boot.dtsi](#) STM32MP157C-ED1 device tree file for U-Boot SPL
5. ↑ [5.05.15.25.3](#) [include/dt-bindings/clock/stm32mp1-clksrc.h](#) (for TF-A), [include/dt-bindings/clock/stm32mp1-clksrc.h](#) (for U-Boot SPL): STM32MP1 DT bindings clock source files
6. ↑ [include/dt-bindings/clock/stm32mp1-clks.h](#) (for TF-A), [include/dt-bindings/clock/stm32mp1-clks.h](#) (for U-Boot SPL): STM32MP1 DT bindings clock identifier files

Operating System

Device Tree

Trusted Firmware for Arm Cortex-A

Generic Interrupt Controller

Serial Peripheral Interface

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

Reset and Clock Control

Microprocessor Unit

Microcontroller Unit

Real Time Clock

System Time Generator

Display Serial Interface (MIPI[®] Alliance standard)

Silicon Provider - NEW

Secure Monitor Call