Category:Clock

Contents

Stable: 17.06.2020 - 15:26 / Revision: 16.01.2020 - 07:57

Category:Clock

Stable: 17.06.2020 - 10:20 | Revision: 16.01.2020 - 07:37

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

This category groups together all articles related to the Linux® **clock** software framework.

It is recommended to first read the Clock overview article.

Pages in category "Clock"

The following 6 pages are in this category, out of 6 total.

- Clock device tree configuration
- Clock device tree configuration - Bootloader specific
- Clock overview
- Non secure RCC configuration
- SCMI device tree configuration
- SCMI overview

Stable: 01.12.2020 - 16:39 / Revision: 01.12.2020 - 16:25

A quality version of this page, approved on *1 December 2020*, was based off this revision.

Contents

1 Article purpose

This article explains how to configure the clocks in the STMP32MP1.

This article focuses on the non-secure world configuration with the device tree technology. Specific articles describe the bootlader stage clock configuration or the alternate ways when disabling the RCC TZEN hardening in non-secure RCC configuration article.

2 Clock device providers

There are 3 clock providers in the STM32MP1. Each are represented by node(s) in the device tree description.

- Fixed clocks, as input clocks with a fixed frequency.
- STM32MP1 RCC clocks, most of the system clocks.
- SCMI clocks, clocks that only the secure world can manipulate, and that the secure world exposes to non-secure world using the SCMI clock protocol and a client/server communication.

The node defines *#clock-cells = <1>;*.
In the device tree bindings, all clock providers must define a specific specifier for the number cells used with the clock device phandle, when referring to the clock. When *#clock-cells = 0*, the clock provider phandle does not need an extra argument. When *#clock-cells = 1*, the clock provider phandle is used with an argument. The STM32MP1 RCC clock and SCMI clock drivers both use stm32mp1 clock DT binding IDs defined in the STM32MP1 clock DT bindings[1].

2.1 STM32MP1 RCC clock device

The device tree defines the RCC clock controller 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.

For example, below is an extract from the Linux kernel and U-Boot device tree representation.

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

                clock-names = "hse", "hsi", "csi", "lse", "lsi";
                clocks = <&scmi0_clk CK_SCMI0_HSE>,
                         <&scmi0_clk CK_SCMI0_HSI>,
                         <&scmi0_clk CK_SCMI0_CSI>,
                         <&scmi0_clk CK_SCMI0_LSE>,
                         <&scmi0_clk CK_SCMI0_LSI>;
        };
```

Note in the file snipped above the STM32MP1 RCC clocks depend on system clocks registered by the SCMI device using the phandles *&scmi0_clk*. Declaring this dependency helps the Linux kernel and U-Boot boot loader to proprely handle the platform clocks.

2.2 SCMI clock device

The device tree defines SCMI clocks using *compatible = "arm,scmi"* nodes with subnodes specifying protocol@14 (*reg = <0x14>*). The node defines *#clock-cells = 1*. The node phandle (label *scmi1_clk* in example below) is used together with a clock ID, using macros *CK_SCMI1_** defined in the DT bindings[2]. One can refer to the article SCMI overview, chapter STM32MP15 SCMI clock and reset for more details.

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

                scmi1_clk: protocol@14 {
                        reg = <0x14>;
                        #clock-cells = <1>;
                };
        };
```

The Linux driver handles all STM32MP clocks with the Linux common clock framework. As does U-Boot with its generic clock udevice framework.

The configuration is performed using the device tree mechanism that provides a description of the fixed clocks if any, RCC peripheral clocks and SCMI clocks used.

3 DT bindings documentation

The RCC is a multi function device.

Each function is represented by a separate binding document:

- generic DT bindings[3] used by the Common Clock framework.
- vendor clock DT bindings[4] used by the clk-stm32mp1 driver: this binding document explains how to write device tree files for clocks.
- generic SCMI DT bindings[5] for the clock protocol support.

4 DT configuration

4.1 DT configuration (STM32 level)

The STM32MP1 Clock node is located in the *stm32mp151.dtsi*[6]. See Device tree for more explanations.

See the example of STM32MP1 RCC clock DT node above in this article, with *compatible = "st,stm32mp1-rcc-secure"*. The node specifies its dependency on the input clock using SCMI clock phandles *scmi0_clk*.

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

                clock-names = "hse", "hsi", "csi", "lse", "lsi";
                clocks = <&scmi0_clk CK_SCMI0_HSE>,
                         <&scmi0_clk CK_SCMI0_HSI>,
                         <&scmi0_clk CK_SCMI0_CSI>,
                         <&scmi0_clk CK_SCMI0_LSE>,
                         <&scmi0_clk CK_SCMI0_LSI>;
        };
```

4.2 DT configuration (board level)

If a Linux driver needs a clock, it has to be added in its DT node:

clocks = <handle> It is the list of the handles to use in the device tree to refer to the target clock instance. The can be several clocks listed, separated with coma character ',', i.e. *clocks = <handle>, <handle>, <handle>;*

In the clock provider node, property *#clock-cells* defines how many 32-bit IDs are to be used with the device phandle to identify the clock.
When the clock provider defines *#clock-cells = 0*, *<handle>* is the single device tree node phandle reference *<&phandle>*.
When the clock provider defines *#clock-cells = 1*, *<handle>* is a pair *<&phandel id>'*

- Example:

```
    usart3: serial@4000f000 {
        compatible = "st,stm32h7-usart";
        reg = <0x4000f000 0x400>;
        interrupts-extended = <&intc GIC_SPI 39 IRQ_TYPE_LEVEL_HIGH>,
                              <&exti 28 1>;
        clocks = <&rcc USART3_K>;
        power-domains = <&pd_core>;
    };

    usart1: serial@5c000000 {
        compatible = "st,stm32h7-usart";
        reg = <0x5c000000 0x400>;
        interrupts-extended = <&intc GIC_SPI 39 IRQ_TYPE_LEVEL_HIGH>,
                              <&exti 28 1>;
        clocks = <&scmi0_clk CK_SCMI0_USART1>;
        power-domains = <&pd_core>;
    };
```

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 and they 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:

- include/dt-bindings/clock/stm32mp1-clks.h STM32MP1 Clock DT bindings header file
- include/dt-bindings/clock/stm32mp1-clks.h STM32MP1 Reset DT bindings header file
- Documentation/devicetree/bindings/clock/clock-bindings.txt , Clock device tree bindings
- Documentation/devicetree/bindings/clock/st,stm32mp1-rcc.txt , STM32MP1 clock device tree bindings
- >Documentation/devicetree/bindings/arm/arm,scmi.txt SCMI DT bindings
- stm32mp151.dtsi STM32MP157C device tree file

Reset and Clock Control

System control and management interface

Device Tree

Linux® is a registered trademark of Linus Torvalds.

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

High Speed External oscillator (STM32 clock source)

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

Multi Speed Internal oscillator (STM32 clock source)

Low Speed External oscillator (STM32 clock source)

Low Speed Internal oscillator (STM32 clock source)

Generic Interrupt Controller

Serial Peripheral Interface

Stable: 09.12.2020 - 13:13 / Revision: 07.12.2020 - 12:45

A quality version of this page, approved on *9 December 2020*, was based off this revision.

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.

Warning

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:

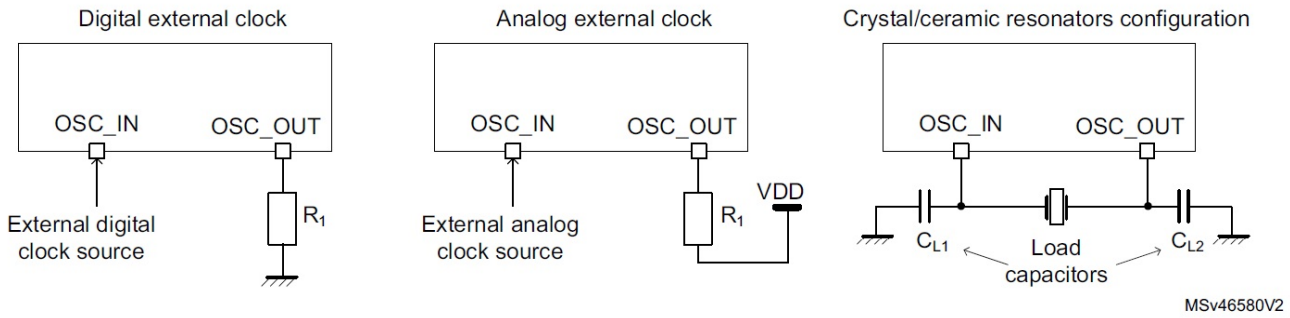
- "st,bypass" configures the external analog clock source (set HSEBYP, LSEBYP),
- "st,digbypass" configures the external digital clock source (set DIGBYP and HSEBYP, LSEBYP),
- "st,css" activates the clock security system (HSECSSON, LSECSSON),
- "st,drive" (LSE only) contains the value of the drive for 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

Stable: 15.10.2019 - 11:58 / Revision: 15.10.2019 - 11:57

A quality version of this page, approved on *15 October 2019*, was based off this revision.

This article gives information about the Linux® Clock framework.
Clocks are generally provided by internal/external oscillators or PLLs.
They can pass through a gate, a muxing, a divider or a multiplier.
All peripheral clocks are organized as a tree.

All these elements are managed by the Common Clock framework.

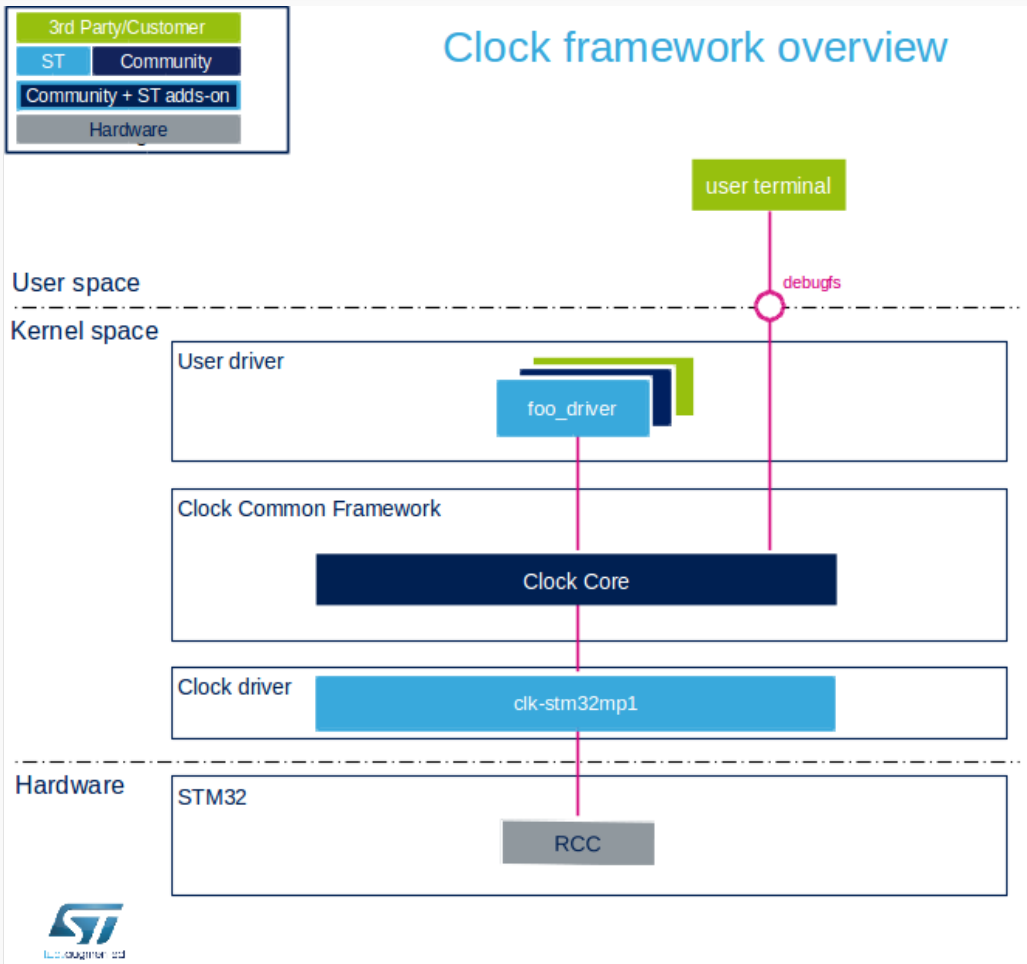
Contents

1 Framework purpose

The purpose of this article is to introduce the Common Clock Framework. It provides general information and, based on examples, explains how to use it.

Linux Common Clock framework offers a generic API for configuring and controlling the different system clocks.
Drivers can easily enable/disable clocks, change their frequency, change the parent of the clocks associated to peripherals without any knowledge of the clock characteristics.
All clock tree specificities (such as clock source selection, muxing, dividor and gate) are abstracted by the Common Clock framework and the associated system clock driver.

2 System overview



Clock Overview

2.1 Component description

- **foo driver:** any peripheral driver which needs to control and activate clock(s) associated to a given peripheral.
- **Clock core:** the Common Clock framework is the generic Linux kernel interface that controls the clock nodes available in the system.

  It features two generic interfaces:

  - The upper interface unifies the definition and control of the clocks for all Linux platforms. This agnostic API is used by peripheral drivers for configuring and controlling the clocks associated to a specific peripheral.

  - The lower interface allows the registration of platform specific functions in order to manage a platform specific clock tree.
- **clk-stm32mp1:** stm32mp1 specific clock driver that supports RCC clocks.
- **RCC:** reset and clock controler RCC.

2.2 API description

Documentation on the Common Clock framework can be found in the kernel documentation folder: https://www.kernel.org/doc/Documentation/driver-api/clk.rst

- Main kernel clock API
  - **devm_get_clk():** looks up and obtains from the device tree a managed reference to a clock producer, to a root clock or to a clock node.
  - **clk_prepare_enable():** selects a parent clock, configures the corresponding multiplexor and dividor, and enables the clock gating.
  - **clk_disable_unprepare():** unprepares and gate a clock.
  - **clk_get_rate():** obtains the current frequency (in Hz) for a given clock.
  - **clk_set_rate():** sets the frequency for a given clock. If a clock has several parents, the clock framework can change the parent in order to obtain a better frequency.
  - **clk_get_parent():** gets the parent clock source for a given clock
  - **clk_set_parent():** sets the parent clock source for a given clock
  - **...**

3 Configuration

3.1 Kernel configuration

By default the Common Clock framework is activated in the kernel configuration.

3.2 Device tree configuration

A detailed device tree configuration is described in Clock device tree configuration.

4 How to use the Common Clock framework

This paragraph describes how a standard peripheral driver can retrieve its clock configuration from the device tree and configure it.

The clocks associated to a given peripheral are declared in the device tree as described in https://www.kernel.org/doc/Documentation/devicetree/bindings/clock/clock-bindings.txt.

Specific platform define statements that abstract hardware clock offsets are defined in dt-bindings/clock/stm32mp1-clks.h header file.

Below an example of clock association to a foo driver:

```
foo: foo@adcdefgh {
       compatible = "foo-driver";
       ...
       clocks = <&rcc FOO_K>;
       ...
};
```

Before using the clock specified in the device tree node, the foo driver has to request it at probe execution.

```
static int foo_probe(struct platform_device *pdev)
{
        struct clk *clk;
        ...
        clk = devm_clk_get(&pdev->dev, NULL);
        if (IS_ERR(clk)) {
                ret = PTR_ERR(clk);
                dev_err(&pdev->dev, "clk get failed: %d\n", ret);
                goto err_master_put;
        }
        ...
}
```

The clock can then be configured and enabled using the Common Clock framework API.

If the peripheral needs several clocks, a name must be associated to each clock handle in the device tree node. The specified names must be used by the driver to retrieve the corresponding pointer for each clock.

Below an example of foo driver managing 2 clocks:

```
foo: foo@abcdefgh {
        compatible = "foo-driver";

        clocks = <&rcc FOO1_K>, <&rcc FOO2_K>;
        clock-names = "foo1", "foo2";
};
```

```
static int foo_probe(struct platform_device *pdev)
{
        priv->foo1clk = devm_clk_get(&pdev->dev, "foo1");
        if (IS_ERR(priv->foo1clk)) {
                ret = PTR_ERR(priv->foo1clk);
```

```
                        if (ret != -ENOENT) {
                                dev_err(&pdev->dev, "Can't get 'foo1' clock\n");
                                return ret;
                        }
                }
                  priv->foo2clk = devm_clk_get(&pdev->dev, "foo2")';
                if (IS_ERR(priv->foo2clk)) {
                        ret = PTR_ERR(priv->foo2clk);
                        if (ret != -ENOENT) {
                                dev_err(&pdev->dev, "Can't get 'foo2' clock\n");
                                return ret;
                        }
                }
          }
```

```
        ret = clk_prepare_enable(priv->foo1clk);
        if (ret < 0) {
                dev_err(dev, "foo1 clk enable failed\n");
                goto err_bclk_disable;
        }
```

5 How to trace and debug the framework

5.1 Tracing using dynamic debug

By default, no kernel log showing the clock activity. However the user can enable the dynamic debug for the clock framework driver:

```
Board $> dmesg -n8
Board $> echo "file drivers/clk/* +p" > /sys/kernel/debug/dynamic_debug/control
```

Refer to dynamic debug for more details.

5.1.1 How to monitor with debugfs

Information on clocks are available in Debugfs interface located in the '/sys/kernel/debug/clk' directory.

It helps viewing all the clocks registered in tree format.

Show clock tree:

```
Board $> cat /sys/kernel/debug/clk/clk_summary

   clock                         enable_cnt  prepare_cnt        rate   accuracy   phase
----------------------------------------------------------------------------------------
clk-ext-camera                          0            0    24000000          0 0
ck_usbo_48m                             1            1    48000000          0 0
   usbo_k                               1            1    48000000          0 0
ck_dsi_phy                              0            0           0          0 0
   dsi_k                                0            0           0          0 0
i2s_ckin                                0            0           0          0 0
clk-csi                                 0            0     4000000          0 0
   ck_csi                               0            0     4000000          0 0
      rng2_k                            0            0     4000000          0 0
      rng1_k                            0            0     4000000          0 0
clk-lsi                                 1            1       32000          0 0
   ck_lsi                               1            1       32000          0 0
      dac12_k                           0            0       32000          0 0
clk-lse                                 1            1       32768          0 0
   ck_lse                               1            1       32768          0 0
      ck_rtc                            1            1       32768          0 0
      cec_k                             0            0       32768          0 0
clk-hsi                                 1            1    64000000          0 0
   clk-hsi-div                          1            1    64000000          0 0
      ck_hsi                            2            2    64000000          0 0
         ck_mco1                        0            0    64000000          0 0
         uart8_k                        0            0    64000000          0 0
         uart7_k                        0            0    64000000          0 0
         uart6_k                        0            0    64000000          0 0
         uart5_k                        0            0    64000000          0 0
         uart4_k                        1            1    64000000          0 0
         usart3_k                       0            0    64000000          0 0
         usart2_k                       0            0    64000000          0 0
         usart1_k                       0            0    64000000          0 0
         i2c6_k                         0            0    64000000          0 0
         i2c4_k                         1            1    64000000          0 0
         i2c5_k                         0            0    64000000          0 0
         i2c3_k                         0            0    64000000          0 0
         i2c2_k                         0            0    64000000          0 0
```

```
         i2c1_k                    0          0    64000000          0 0
         spi6_k                    0          0    64000000          0 0
         spi5_k                    0          0    64000000          0 0
         spi4_k                    0          0    64000000          0 0
clk-hse                            1          1    24000000          0 0
   ck_hse                          6          6    24000000          0 0
      ck_hse_rtc                   0          0     1000000          0 0
      stgen_k                      1          1    24000000          0 0
      usbphy_k                     1          1    24000000          0 0
      ck_per                       0          0    24000000          0 0
         adc12_k                   0          0    24000000          0 0
      ref4                         1          1    24000000          0 0
         pll4                      1          1   508000000          0 0
            pll4_r                 0          0    56444445          0 0
            pll4_q                 1          1    50800000          0 0
               ltdc_px             1          1    50800000          0 0
               dsi_px              0          0    50800000          0 0
               fdcan_k             0          0    50800000          0 0
            pll4_p                 0          0    56444445          0 0
      ref3                         1          1    24000000          0 0
         pll3                      2          3   786431640          0 0
            pll3_r                 1          1    98303955          0 0
               sdmmc3_k            0          0    98303955          0 0
               sdmmc2_k            0          0    98303955          0 0
               sdmmc1_k            1          1    98303955          0 0
            pll3_q                 0          3    49151978          0 0
               adfsdm_k            0          0    49151978          0 0
               sai4_k              0          1    49151978          0 0
               sai3_k              0          0    49151978          0 0
               sai2_k              0          2    49151978          0 0
               sai1_k              0          0    49151978          0 0
               spi3_k              0          0    49151978          0 0
               spi2_k              0          0    49151978          0 0
               spi1_k              0          0    49151978          0 0
               spdif_k             0          1    49151978          0 0
            pll3_p                 1          1   196607910          0 0
               ck_mcu              6         19   196607910          0 0
                  dfsdm_k          0          1   196607910          0 0
                  gpiok            0          1   196607910          0 0
                  gpioj            0          1   196607910          0 0
                  gpioi            0          1   196607910          0 0
                  gpioh            0          1   196607910          0 0
                  gpiog            0          1   196607910          0 0
                  gpiof            0          1   196607910          0 0
                  gpioe            0          1   196607910          0 0
                  gpiod            0          1   196607910          0 0
                  gpioc            0          1   196607910          0 0
                  gpiob            0          1   196607910          0 0
                  gpioa            0          1   196607910          0 0
                  ipcc             2          2   196607910          0 0
                  hsem             0          0   196607910          0 0
                  crc2             0          0   196607910          0 0
                  rng2             0          0   196607910          0 0
                  hash2            0          0   196607910          0 0
                  cryp2            0          0   196607910          0 0
                  dcmi             0          0   196607910          0 0
                  sdmmc3           0          0   196607910          0 0
                  usbo             0          0   196607910          0 0
                  adc12            0          0   196607910          0 0
                  dmamux           1          1   196607910          0 0
                  dma2             1          1   196607910          0 0
                  dma1             1          1   196607910          0 0
                  pclk3            1          1    98303955          0 0
                     lptim5_k      0          0    98303955          0 0
                     lptim4_k      0          0    98303955          0 0
                     lptim3_k      0          0    98303955          0 0
                     lptim2_k      0          0    98303955          0 0
```

```
      hdp                0          0    98303955          0 0
      pmbctrl            0          0    98303955          0 0
      tmpsens            0          0    98303955          0 0
      vref               0          0    98303955          0 0
      syscfg             1          1    98303955          0 0
      sai4               0          0    98303955          0 0
      lptim5             0          0    98303955          0 0
      lptim4             0          0    98303955          0 0
      lptim3             0          0    98303955          0 0
      lptim2             0          0    98303955          0 0
   pclk2                 0          0    98303955          0 0
      fdcan              0          0    98303955          0 0
      dfsdm              0          0    98303955          0 0
      sai3               0          0    98303955          0 0
      sai2               0          0    98303955          0 0
      sai1               0          0    98303955          0 0
      usart6             0          0    98303955          0 0
      spi5               0          0    98303955          0 0
      spi4               0          0    98303955          0 0
      spi1               0          0    98303955          0 0
      tim17              0          0    98303955          0 0
      tim16              0          0    98303955          0 0
      tim15              0          0    98303955          0 0
      tim8               0          0    98303955          0 0
      tim1               0          0    98303955          0 0
      ck2_tim            0          0   196607910          0 0
         tim17_k          0          0   196607910           0 0
         tim16_k          0          0   196607910           0 0
         tim15_k          0          0   196607910           0 0
         tim8_k           0          0   196607910          0 0
         tim1_k           0          0   196607910          0 0
   pclk1                 0          2    98303955          0 0
      lptim1_k           0          0    98303955          0 0
      mdio               0          0    98303955          0 0
      dac12              0          1    98303955          0 0
      cec                0          0    98303955          0 0
      spdif              0          0    98303955          0 0
      i2c5               0          0    98303955          0 0
      i2c3               0          0    98303955          0 0
      i2c2               0          0    98303955          0 0
      i2c1               0          0    98303955          0 0
      uart8              0          0    98303955          0 0
      uart7              0          0    98303955          0 0
      uart5              0          0    98303955          0 0
      uart4              0          0    98303955          0 0
      usart3             0          0    98303955          0 0
      usart2             0          0    98303955          0 0
      spi3               0          0    98303955          0 0
      spi2               0          0    98303955          0 0
      lptim1             0          0    98303955          0 0
      tim14              0          0    98303955          0 0
      tim13              0          0    98303955          0 0
      tim12              0          0    98303955          0 0
      tim7               0          0    98303955          0 0
      tim6               0          0    98303955          0 0
      tim5               0          0    98303955          0 0
      tim4               0          0    98303955          0 0
      tim3               0          0    98303955          0 0
      tim2               0          0    98303955          0 0
      ck1_tim            0          1   196607910          0 0
         tim14_k          0          0   196607910           0 0
         tim13_k          0          0   196607910           0 0
         tim12_k          0          0   196607910           0 0
         tim7_k           0          0   196607910          0 0
         tim6_k           0          1   196607910          0 0
         tim5_k           0          0   196607910          0 0
         tim4_k           0          0   196607910          0 0
```

```
                        tim3_k              0            0   196607910          0 0
                        tim2_k              0            0   196607910          0 0
      ref1                                  2            2    24000000          0 0
         pll2                               2            2   533000000          0 0
            pll2_r                          1            1   533000000          0 0
            pll2_q                          0            0   533000000          0 0
               gpu_k                        0            0   533000000          0 0
            pll2_p                          1            1   266500000          0 0
               ck_axi                       9           10   266500000          0 0
                  ck_trace                  0            0   133250000          0 0
                  ck_sys_dbg                0            0   266500000          0 0
                  qspi_k                    1            1   266500000          0 0
                  fmc_k                     0            0   266500000          0 0
                  ethstp                    0            0   266500000          0 0
                  usbh                      1            1   266500000          0 0
                  crc1                      0            0   266500000          0 0
                  sdmmc2                    0            0   266500000          0 0
                  sdmmc1                    0            0   266500000          0 0
                  qspi                      0            0   266500000          0 0
                  fmc                       0            0   266500000          0 0
                  ethmac                    1            1   266500000          0 0
                  ethrx                     1            1   266500000          0 0
                  ethtx                     1            1   266500000          0 0
                  gpu                       0            0   266500000          0 0
                  mdma                      1            1   266500000          0 0
                  bkpsram                   0            0   266500000          0 0
                  rng1                      0            0   266500000          0 0
                  hash1                     0            0   266500000          0 0
                  cryp1                     0            0   266500000          0 0
                  gpioz                     0            1   266500000          0 0
                  tzc2                      0            0   266500000          0 0
                  tzc1                      0            0   266500000          0 0
                  pclk5                     1            1    66625000          0 0
                     stgen                  0            0    66625000          0 0
                     bsec                   0            0    66625000          0 0
                     iwdg1                  0            0    66625000          0 0
                     tzpc                   0            0    66625000          0 0
                     rtcapb                 2            2    66625000          0 0
                     usart1                 0            0    66625000          0 0
                     i2c6                   0            0    66625000          0 0
                     i2c4                   0            0    66625000          0 0
                     spi6                   0            0    66625000          0 0
                  pclk4                     1            1   133250000          0 0
                     stgenro                0            0   133250000          0 0
                     usbphy                 0            0   133250000          0 0
                     iwdg2                  1            1   133250000          0 0
                     dsi                    0            0   133250000          0 0
                     ltdc                   0            0   133250000          0 0
         pll1                               1            1   650000000          0 0
            pll1_p                          1            1   650000000          0 0
               ck_mpu                       1            1   650000000          0 0
                  ck_mco2                   0            0   650000000          0 0
      clk-hse-div2                          0            0    12000000          0 0
ethptp_k                                    0            0           0          0 0
ethck_k
```

In addition, each clock has a dedicated directory in which you can find the information such as prepare count, enable count, rate and accuracy:

```
Board $>/sys/kernel/debug/clk# cd usart2_k
Board $>/sys/kernel/debug/clk/usart2_k# ls
clk_accuracy      clk_flags           clk_phase             clk_prepare_count
clk_enable_count  clk_notifier_count  clk_possible_parents  clk_rate
```

Additional information, such as flags, notifier count and possible parents (for clocks with multiple parents) are also available. Just execute a 'cat' command to display them.

```
Board $>:/sys/kernel/debug/clk/usart2_k# cat clk_possible_parents
pclk1 pll4_q ck_hsi ck_csi ck_hse
```

6 Source code location

stm32mp1 clock driver source [1]
clock dt-binding interface [2]

7 To go further

See "how to build a clock tree" in STM32MP15_clock_tree.

8 References

- drivers/clk/clk-stm32mp1.c
- include/dt-bindings/clock/stm32mp1-clks.h

Linux® is a registered trademark of Linus Torvalds.

Application programming interface

Reset and Clock Control

Debug File System (See https://en.wikipedia.org/wiki/Debugfs for more details)

Stable: Not stable / Revision: 19.02.2021 - 07:41

Invalid target: no reviewed revision corresponds to the given ID.

Return to Non secure RCC configuration.

Stable: Not stable / Revision: 07.01.2021 - 13:50

Invalid target: no reviewed revision corresponds to the given ID.

Return to SCMI device tree configuration.

Stable: 20.07.2020 - 09:39 / Revision: 20.07.2020 - 09:37

A quality version of this page, approved on *20 July 2020*, was based off this revision.

Contents

1 Article Purpose

The purposes of this article is to explain the SCMI services in STM32MPU.

SCMI stands for System Control and Magagement Interface. It refers to Arm SCMI specification[1]. SCMI is a message driven interface where specific SCMI protocols expose statndard service for system resources.

STM32MP15 uses SCMI to abstrat RCC resources, as clock and reset controllers, assigned to secure world excluvie access by RCC TZEN configuration. Secure world embeds a SCMI server that exposes clock and reset controllers. Non-secure world uses SCMI drivers as SCMI clock driver *clk-scmi.c*[2] and SCMI reset driver *reset-scmi.c*[3] to handle the resource in the system.

2 What is SCMI and what is it used for

SCMI is a protocol based on message exchanged. It allows a resource master to expose device services to a client entity.

For example, the CPU clock cannot be freely manipulated by the non-secure world. Yet, Linux kernel executing in non-secure world may need to get or set the DDR frequency. Implementing a SCMI agent in non-secure world (Linux kernel, U-Boot bootloader) and a SCMI server in secure firmware (TF-A, OP-TEE) allows non-secure Linux kernel and U-Boot to natively discover the DDR clock as a standard clock device.

In the example, applied to STM32MP15, enabling security on the DDR clock is also related to enable the security on DDR controllers interfaces. DDR clock secure hardening is enabled from RCC interface. DDR controller access permissions are configure through ETZPC. STM32MP15 software release ensures platform configuration consistency.

3 STM32MP15 SCMI overview

SCMI on STM32MP15 is used to create server/client paths between secure and non-secure world to offer device driver services for the SoC resource when some firewalls configuration are hardened.

In STM32MP1 software release, the secure world, being TF-A or OP-TEE, embeds a SCMI server that exposes clock and reset controllers that non-secure world cannot access straight due to the configuration loaded in STM32MP1 RCC security hardening .

In STM32MP15 software release, secure world exposes some clock and reset controllers to non-secure world. Secure world can be TF-A or OP-TEE. They implement features from the SCMI specification v2.0 [4]. TF-A and OP-TEE supports necessary features from SCMI Base: clock, reset, domain protocols. These clock and reset controllers are related the SoC configuration that grants write access to specific RCC interface register. As stated in RCC security hardening, RCC TZEN restricts certain resource accesses, RCC MCKPROT restricts even more resource accesses.

Since there are 2 levels of secure hardening, STM32MP1 exposes system resources over 2 channel, represented by 2 agents in non-secure world. SCMI agent 0 discovers clocks and reset hardened per RCC TZEN enabling. SCMI agent 1 discovers clocks hardended upon RCC MCKPROT enabling.

4 STM32MP15 SCMI clocks and reset

In STM32MP15, SCMI exposes resource for the peripheral interfaces listed below. Note that when a related device is assigned to secure world by system configuration (see How to assign an internal peripheral to a runtime context for further information), the SCMI controls, even if exposed, will not allow non-secure world to effectively access the resource.

The SCMI clock protocol phandle together with the clock ID, supported values are defined in STM32MP1 DT clock bindings[5], forms a unique identifer for the clock resources in Linux and U-Boot executable.

The SCMI reset domain protocol phandle together with the agent reset domain ID, supported values are defined in STM32MP1 DT reset bindings[6], forms a unique identifer for the reset controller resources.

- SCMI exposes CRYP1 clock with ID CK_SCMI0_CRYP1 to non-secure agent 0.

SCMI exposes CRYP1 reset with ID RST_SCMI0_CRYP1 to non-secure agent 0.

- SCMI exposes HASH1 clock with ID CK_SCMI0_HASH1 to non-secure agent 0.

SCMI exposes HASH1 reset with ID RST_SCMI0_CRYP1 to non-secure agent 0.

- SCMI exposes RNG1 clock with ID CK_SCMI0_RNG1 to non-secure agent 0.

SCMI exposes RNG1 reset with ID RST_SCMI0_RNG1 to non-secure agent 0.

- SCMI exposes BSEC clock with ID CK_SCMI0_BSEC to non-secure agent 0.
- SCMI exposes GPIOZ bank clock with ID CK_SCMI0_GPIOZ to non-secure agent 0.

SCMI exposes GPIOZ bank reset with ID RST_SCMI0_GPIOZ to non-secure agent 0.

- SCMI exposes I2C4 and I2C6 clock with ID CK_SCMI0_I2Cx to non-secure agent 0.

SCMI exposes I2C4 and I2C6 reset with ID RST_SCMI0_I2Cx to non-secure agent 0.

- SCMI exposes SPI6 clock with ID CK_SCMI0_SPI6 to non-secure agent 0.

SCMI exposes SPI6 reset with ID RST_SCMI0_SPI6 to non-secure agent 0.

- SCMI exposes USART1 clock with ID CK_SCMI0_USART1 to non-secure agent 0.

SCMI exposes USART1 reset with ID RST_SCMI0_USART1 to non-secure agent 0.

- SCMI exposes IWDG1 bus clock with ID CK_SCMI0_IWDG1 to non-secure agent 0.
- SCMI exposes RTC clock controller with ID CK_SCMI0_RTC to non-secure agent 0.
- SCMI exposes MDMA reset controller with ID CK_SCMI0_MDMA to non-secure agent 0.
- SCMI exposes MCU clock to non-secure agent 1 with ID CK_SCMI1_MCU.

SCMI exposes MCU reset to non-secure agent 0 with ID RST_SCMI0_MCU.

They can be used for coprocessor startup whether via Linux remoteproc framework or U-Boot bootloader.

- SCMI exposes internal root clocks LSI, HSI and CSI to non-secure agent 0

with IDs CLK_SCMI0_LSI, CLK_SCMI0_HSI and CLK_SCMI0_CSI.

- SCMI exposes external root clocks LSE and HSE to non-secure agent 0

with IDs CK_SCMI0_HSE and CK_SCMI0_LSE.

- SCMI exposes PLL2Q and PLL2R to non-secure agent 0

with IDs CK_SCMI0_PLL2_Q and CK_SCMI0_PLL2_R.

- SCMI exposes PLL3Q and PLL3R to non-secure agent 1

with IDs CK_SCMI1_PLL3_Q and CK_SCMI1_PLL3_R.

5 STM32MP15 SCMI agents

Among all clocks and resets exposed by STM32MP15, see the list in the previous section, all but MCU clock and PLL3Q and PLL3R clocks are related to RCC TZEN hardening, the 3 later being related to RCC MCKPROT hardening.

When RCC TZEN hardening is enabled, non-secure world shall rely on SCMI agent 0 to access the desired resources.

When RCC MCKPROT hardening is enabled, non-secure world shall rely on SCMI agent 1 to access the desired resources.

Each agent interface uses a specific shared memory for SCMI message exchanges. A specific notification layer is also used between non-secure and secure worlds. The device tree technology is used the define the SCMI message transport configuration, in non-secure side. In secure world, the transport means are built-in firmware image. The shared memories used are all located at the top (high addresses) is SYSRAM internal memory.

6 Device Tree description

The SCMI services device tree bindings are defined in the Linux kernel source tree in Arm SCMI DT bindings[7] documentation.

6.1 SCMI agent nodes

Each SCMI agent is a subnode of the firmware node in the device tree description.

SCMI devices are described in the *firmware* node of the device tree, with *compatible = "arm,scmi"*.

```
                scmi-agent {
                        compatible = "arm,scmi";

                        (...) /* SCMI transport properties */

                        /* Below are the discovered SCMI protocols */
                        protocol@X {
                                reg = <0xX>;
                                (...)
                        };
                        protocol@Y {
                                reg = <0xY>;
                                (...)
                        };
                };
```

6.2 STM32MP1 SCMI Transport

SCMI framework support several transport layers for message exchange. SCMI uses a transport built on *mmio-sram* shared memory instance with a *smc-mbox* mailbox device for message notification.

Shared memory device with *mmio-sram*. Device tree bindings offers compatible *"mmio-sram"* devices to described physical memory. It is supported by U-Boot and Linux kernel. In STM32MP1 memory layout, 2 shared memories in SYSRAM internal RAM are reserved for non-secure SCMI communication. In exmample below, phandle *scmi0_shm* refers to SYSRAM range [0x2FFFF000 0x2FFFF07F] used by SCMI agent 0 for client/server communication.

```
        sram@2ffff000 {
                compatible = "mmio-sram";
                reg = <0x2ffff000 0x1000>;
                #address-cells = <1>;
                #size-cells = <1>;
                ranges = <0 0x2ffff000 0x1000>;

                scmi0_shm: scmi_shm@0 {
                        reg = <0 0x80>;
                };
        };
```

Mailbox device: ad-hoc *arm,smc-mbox*.
STM32MP1 software release uses a SMC mailbox described with a "arm,smc-mbox" compatible node. Property *arm,func-id* defines the function ID used to invoke secure world for a specific SCMI agent/server interface.

```
        scmi0_mbox: mailbox-0 {
                #mbox-cells = <0>;
                compatible = "arm,smc-mbox";
                arm,func-id = <0x82002000>;
        };
```

In example below, SCMI agent 0 uses *scmi0_mbox* mailbox and *scmi0_shm* as shared memory.

```
        firmware {
                scmi0: scmi-0 {
                        compatible = "arm,scmi";
                        (...)
                        mboxes = <&scmi0_mbox 0>;
                        mbox-names = "txrx";
                        shmem = <&scmi0_shm>;
                        (...)
                };
        };
```

6.3 STM32MP15 SCMI device tree nodes

STM32MP15 exposes SCMI over 2 devices in the device tree description. One device per supported SCMI agents, hence two: one for the clocks and reset controllers related to RCC TZEN hardening configuration, and one for the clock controllers related to RCC MCKPROT hardening configuration.

As defined in Arm SCMI DT bindings[8], a SCMI agent node can contain one or more supported SCMI protocols. Each are described in a specific subnode of the agent node. STM32MP15 implements SCMI Clock protocol (ID 0x14) that is a clock provider device and SCMI Reset Domain protocol (ID 0x16) that is a reset controller provider device.

Example below shows a configuration supported in STM32MP15 software release.

```
        /* Shared memory used for SCMI agent/server communication */
        scmi_sram: sram@2ffff000 {
                compatible = "mmio-sram";
                reg = <0x2ffff000 0x1000>;
                #address-cells = <1>;
                #size-cells = <1>;
                ranges = <0 0x2ffff000 0x1000>;

                scmi0_shm: scmi_shm@0 {
                        reg = <0 0x80>;
                };
                scmi1_shm: scmi_shm@200 {
                        reg = <0x200 0x80>;
                };
        };

        /* Mailbox device used for SCMI agent/server communication */
        scmi0_mbox: mailbox-0 {
                #mbox-cells = <0>;
                compatible = "arm,smc-mbox";
                arm,func-id = <0x82002000>;
        };
        scmi1_mbox: mailbox-1 {
                #mbox-cells = <0>;
                compatible = "arm,smc-mbox";
                arm,func-id = <0x82002001>;
        };
```

```
/* SCMI agent nodes, in the firmware node */
firmware {
        scmi0: scmi-0 {
                compatible = "arm,scmi";
                #address-cells = <1>;
                #size-cells = <0>;

                status = "okay"; /* To enable upon RCC[TZEN] */
                mboxes = <&scmi0_mbox 0>;
                mbox-names = "txrx";
                shmem = <&scmi0_shm>;

                scmi0_clk: protocol@14 {
                        reg = <0x14>;
                        #clock-cells = <1>;
                };
                scmi0_reset: protocol@16 {
                        reg = <0x16>;
                        #reset-cells = <1>;
                };
        };

        scmi1: scmi-1 {
                compatible = "arm,scmi";
                #address-cells = <1>;
                #size-cells = <0>;

                status = "disabled"; /* To enable upon RCC[MCKPROT] */
                mboxes = <&scmi1_mbox 0>;
                mbox-names = "txrx";
                shmem = <&scmi1_shm>;

                scmi1_clk: protocol@14 {
                        reg = <0x14>;
                        #clock-cells = <1>;
                };
        };
```

7 How to go further

One can follow SCMI developments through TF-A mailing list[9] and OP-TEE OS issues[10]and pull requests[11]forums.

8 References

- https://developer.arm.com/architectures/system-architectures/software-standards/scmi
- Linux drivers/clk/clk-scmi.c
- Linux drivers/reset/reset-scmi.c
- https://developer.arm.com/docs/den0056/latest
- include/dt-bindings/clock/stm32mp1-clks.h stm32mp1 DT clocks bindings
- include/dt-bindings/reset/stm32mp1-resets.h stm32mp1 DT reset bindings
- Documentation/devicetree/bindings/arm/arm,scmi.txt Arm SCMI DT bindings
- Documentation/devicetree/bindings/arm/arm,scmi.txt Arm SCMI DT bindings
- https://lists.trustedfirmware.org/mailman/listinfo/tf-a
- https://github.com/OP-TEE/optee_os/issues
- https://github.com/OP-TEE/optee_os/pulls

System control and management interface

Central processing unit

Doubledata rate (memory domain)

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

Trusted Firmware for Arm Cortex-A

Open Portable Trusted Execution Environment

Reset and Clock Control

Device Tree

Boot and Security and OTP control

Real Time Clock

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.)

Low Speed Internal oscillator (STM32 clock source)

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

Multi Speed Internal oscillator (STM32 clock source)

High Speed External oscillator (STM32 clock source)

Low Speed External oscillator (STM32 clock source)

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Secure Monitor Call

Operating System