SDMMC device tree configuration

Contents

SDMMC device tree configuration

Stable: 11.02.2019 - 11:21 / Revision: 21.01.2019 - 13:52

A quality version of this page, accepted on *11 February 2019*, was based off this revision.

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the SDMMC peripheral, used by the STM32 SDMMC Linux driver and by the MMC framework.

2 DT bindings documentation

The SDMMC device tree bindings are composed of:

- generic MMC device tree bindings [1].
- SDMMC MMC/SD/SDIO interface bindings [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 SDMMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
   sdmmc1: sdmmc@58005000 {                                      Comments
       compatible = "arm,pl18x", "arm,primecell";
       arm,primecell-periphid = <0x00253180>;
       reg = <0x58005000 0x1000>,                                --> The controller
register location
             <0x58006000 0x1000>;                                --> The delay block
register location
       interrupts = <GIC_SPI 49 IRQ_TYPE_LEVEL_HIGH>;            --> The interrupt number
used
       interrupt-names = "cmd_irq";
       clocks = <&rcc SDMMC1_K>;
       clock-names = apb_pclk
       resets = <&rcc SDMMC1_R>;
       status = "disabled";
   };
```

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SDMMC peripheral may connect to one SD card, one *e*•MMC™ device or one SDIO card.

```
   &sdmmc1{                                                      Comments
       pinctrl-names = "default", "opendrain", "sleep";          --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       st,neg-edge;                                              --> Generate data and
command on sdmmc clock falling edge
       st,sig-dir;                                               --> Allow to select
direction polarity of an external transceiver
       st,use-ckin;                                              --> Use sdmmc_ckin pin
from an external transceiver to sample the receive data
       bus-width = <4>;                                          --> Number of data lines,
can be 1, 4 or 8
       vmmc-supply = <&vdd_sd>;                                  --> Supply node for
card's power
       vqmmc-supply = <&sd_switch>;                              --> Supply node for IO
line power
       status = "okay";                                          --> Enable the node
   };
```

Below optional properties have to be used when an external transceiver is connected:

- st,sig-dir: This property allows to select external transceiver direction signals polarity. When this property is set, the voltage transceiver IOs are driven as output when the direction signals are high. Without setting this property, the voltage transceiver IOs are driven as output when the direction signals are low.

- st,use-ckin: By setting this property, the sdmmc_ckin pin from an external transceiver is used to sample the receive data.

3.3 DT configuration examples

Below example shows how to configure the SDMMC when an *e*•MMC™ is connected with 8 data lines [4].

```
   &sdmmc2{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc2_b4_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-1 = <&sdmmc2_b4_od_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-2 = <&sdmmc2_b4_sleep_pins_a &sdmmc2_dir_sleep_pins_a>;
       non-removable;                                        --> Non-removable slot,
assume always present
       no-sd;                                                --> Avoid to send SD
command during initialization
       no-sdio;                                              --> Avoid to send SDIO
command during initialization
       st,neg-edge;
       bus-width = <8>;
       vmmc-supply = <&v3v3>;
       vqmmc-supply = <&vdd>;
       mmc-ddr-3_3v;                                         --> Host supports e•MMC™ D
DR 3.3V
       status = "okay";
   };
```

Below example shows how to configure the SDMMC to SD card (4 data lines) with an external transceiver [4].

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       broken-cd;                                            --> use polling mode for
card detection
       st,neg-edge;
       st,sig-dir;
       st,use-ckin;
       bus-width = <4>;
       sd-uhs-sdr12;                                         --> sd modes supported [1]
       sd-uhs-sdr25;
       sd-uhs-sdr50;
       sd-uhs-ddr50;
       sd-uhs-sdr104;
       vmmc-supply = <&vdd_sd>;
       vqmmc-supply = <&sd_switch>;
       status = "okay";
   };
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- 1.01.1 Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/mmc/mmci.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- 4.04.1 arch/arm/boot/dts/stm32mp157c-ed1.dts

Operating System

MultimediaCard

Device Tree

Secure digital

Generic Interrupt Controller

Serial Peripheral Interface

SD memory card (https://www.sdcard.org)

SDIO is an SD-size card with extended input/output functions

input/output

Doubledata rate (memory domain)

SDMMC device tree configuration

Stable: 14.05.2020 - 07:13 / Revision: 14.05.2020 - 07:12

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the SDMMC peripheral, used by the STM32 SDMMC Linux driver and by the MMC framework.

2 DT bindings documentation

The SDMMC device tree bindings are composed of:

- generic MMC device tree bindings [1].
- SDMMC MMC/SD/SDIO interface bindings [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 SDMMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
   sdmmc1: sdmmc@58005000 {                                  Comments
       compatible = "arm,pl18x", "arm,primecell";
       arm,primecell-periphid = <0x00253180>;
       reg = <0x58005000 0x1000>,                            --> The controller
register location
             <0x58006000 0x1000>;                            --> The delay block
register location
       interrupts = <GIC_SPI 49 IRQ_TYPE_LEVEL_HIGH>;        --> The interrupt number
used
       interrupt-names = "cmd_irq";
       clocks = <&rcc SDMMC1_K>;
       clock-names = apb_pclk
       resets = <&rcc SDMMC1_R>;
       status = "disabled";
   };
```

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SDMMC peripheral may connect to one SD card, one *e*•MMC™ device or one SDIO card.

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";      --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       st,neg-edge;                                          --> Generate data and
command on sdmmc clock falling edge
       st,sig-dir;                                           --> Allow to select
direction polarity of an external transceiver
       st,use-ckin;                                          --> Use sdmmc_ckin pin
from an external transceiver to sample the receive data
       bus-width = <4>;                                      --> Number of data lines,
can be 1, 4 or 8
       vmmc-supply = <&vdd_sd>;                              --> Supply node for
card's power
       vqmmc-supply = <&sd_switch>;                          --> Supply node for IO
line power
       status = "okay";                                      --> Enable the node
   };
```

Below optional properties have to be used when an external transceiver is connected:

- st,sig-dir: This property allows to select external transceiver direction signals polarity. When this property is set, the voltage transceiver IOs are driven as output when the direction signals are high. Without setting this property, the voltage transceiver IOs are driven as output when the direction signals are low.
- st,use-ckin: By setting this property, the sdmmc_ckin pin from an external transceiver is used to sample the receive data.

3.3 DT configuration examples

Below example shows how to configure the SDMMC when an *e*•MMC™ is connected with 8 data lines [4].

```
   &sdmmc2{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc2_b4_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-1 = <&sdmmc2_b4_od_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-2 = <&sdmmc2_b4_sleep_pins_a &sdmmc2_dir_sleep_pins_a>;
       non-removable;                                        --> Non-removable slot,
assume always present
       no-sd;                                                --> Avoid to send SD
command during initialization
       no-sdio;                                              --> Avoid to send SDIO
command during initialization
       st,neg-edge;
       bus-width = <8>;
       vmmc-supply = <&v3v3>;
       vqmmc-supply = <&vdd>;
       mmc-ddr-3_3v;                                         --> Host supports e•MMC™ D
DR 3.3V
       status = "okay";
   };
```

Below example shows how to configure the SDMMC to SD card (4 data lines) with an external transceiver [4].

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       broken-cd;                                            --> use polling mode for
card detection
       st,neg-edge;
       st,sig-dir;
       st,use-ckin;
       bus-width = <4>;
       sd-uhs-sdr12;                                         --> sd modes supported [1]
       sd-uhs-sdr25;
       sd-uhs-sdr50;
       sd-uhs-ddr50;
       sd-uhs-sdr104;
       vmmc-supply = <&vdd_sd>;
       vqmmc-supply = <&sd_switch>;
       status = "okay";
   };
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- 1.01.1 Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/mmc/mmci.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- 4.04.1 arch/arm/boot/dts/stm32mp157c-ed1.dts

Operating System

MultimediaCard

Device Tree

Secure digital

Generic Interrupt Controller

Serial Peripheral Interface

SD memory card (https://www.sdcard.org)

SDIO is an SD-size card with extended input/output functions

input/output

Doubledata rate (memory domain)

SDMMC device tree configuration

Stable: 14.05.2020 - 09:26 / Revision: 14.05.2020 - 09:25

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the SDMMC peripheral, used by the STM32 SDMMC Linux driver and by the MMC framework.

2 DT bindings documentation

The SDMMC device tree bindings are composed of:

- generic MMC device tree bindings [1].
- SDMMC MMC/SD/SDIO interface bindings [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 SDMMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
   sdmmc1: sdmmc@58005000 {                                  Comments
       compatible = "arm,pl18x", "arm,primecell";
       arm,primecell-periphid = <0x00253180>;
       reg = <0x58005000 0x1000>,                            --> The controller
register location
             <0x58006000 0x1000>;                            --> The delay block
register location
       interrupts = <GIC_SPI 49 IRQ_TYPE_LEVEL_HIGH>;        --> The interrupt number
used
       interrupt-names = "cmd_irq";
       clocks = <&rcc SDMMC1_K>;
       clock-names = apb_pclk
       resets = <&rcc SDMMC1_R>;
       status = "disabled";
   };
```

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SDMMC peripheral may connect to one SD card, one *e*•MMC™ device or one SDIO card.

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";      --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       st,neg-edge;                                          --> Generate data and
command on sdmmc clock falling edge
       st,sig-dir;                                           --> Allow to select
direction polarity of an external transceiver
       st,use-ckin;                                          --> Use sdmmc_ckin pin
from an external transceiver to sample the receive data
       bus-width = <4>;                                      --> Number of data lines,
can be 1, 4 or 8
       vmmc-supply = <&vdd_sd>;                              --> Supply node for
card's power
       vqmmc-supply = <&sd_switch>;                          --> Supply node for IO
line power
       status = "okay";                                      --> Enable the node
   };
```

Below optional properties have to be used when an external transceiver is connected:

- st,sig-dir: This property allows to select external transceiver direction signals polarity. When this property is set, the voltage transceiver IOs are driven as output when the direction signals are high. Without setting this property, the voltage transceiver IOs are driven as output when the direction signals are low.
- st,use-ckin: By setting this property, the sdmmc_ckin pin from an external transceiver is used to sample the receive data.

3.3 DT configuration examples

Below example shows how to configure the SDMMC when an *e*•MMC™ is connected with 8 data lines [4].

```
   &sdmmc2{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc2_b4_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-1 = <&sdmmc2_b4_od_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-2 = <&sdmmc2_b4_sleep_pins_a &sdmmc2_dir_sleep_pins_a>;
       non-removable;                                        --> Non-removable slot,
assume always present
       no-sd;                                                --> Avoid to send SD
command during initialization
       no-sdio;                                              --> Avoid to send SDIO
command during initialization
       st,neg-edge;
       bus-width = <8>;
       vmmc-supply = <&v3v3>;
       vqmmc-supply = <&vdd>;
       mmc-ddr-3_3v;                                         --> Host supports e•MMC™ D
DR 3.3V
       status = "okay";
   };
```

Below example shows how to configure the SDMMC to SD card (4 data lines) with an external transceiver [4].

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       broken-cd;                                            --> use polling mode for
card detection
       st,neg-edge;
```

```
        st,sig-dir;
        st,use-ckin;
        bus-width = <4>;
        sd-uhs-sdr12;                                             --> sd modes supported [1]
        sd-uhs-sdr25;
        sd-uhs-sdr50;
        sd-uhs-ddr50;
        sd-uhs-sdr104;
        vmmc-supply = <&vdd_sd>;
        vqmmc-supply = <&sd_switch>;
        status = "okay";
    };
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- 1.01.1 Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/mmc/mmci.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- 4.04.1 arch/arm/boot/dts/stm32mp157c-ed1.dts

Operating System

MultimediaCard

Device Tree

Secure digital

Generic Interrupt Controller

Serial Peripheral Interface

SD memory card (https://www.sdcard.org)

SDIO is an SD-size card with extended input/output functions

input/output

Doubledata rate (memory domain)

SDMMC device tree configuration

Stable: 04.02.2020 - 07:47 / Revision: 04.02.2020 - 07:34

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the SDMMC peripheral, used by the STM32 SDMMC Linux driver and by the MMC framework.

2 DT bindings documentation

The SDMMC device tree bindings are composed of:

- generic MMC device tree bindings [1].
- SDMMC MMC/SD/SDIO interface bindings [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 SDMMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
   sdmmc1: sdmmc@58005000 {                                  Comments
       compatible = "arm,pl18x", "arm,primecell";
       arm,primecell-periphid = <0x00253180>;
       reg = <0x58005000 0x1000>,                            --> The controller
register location
             <0x58006000 0x1000>;                            --> The delay block
register location
       interrupts = <GIC_SPI 49 IRQ_TYPE_LEVEL_HIGH>;        --> The interrupt number
used
       interrupt-names = "cmd_irq";
       clocks = <&rcc SDMMC1_K>;
       clock-names = apb_pclk
       resets = <&rcc SDMMC1_R>;
       status = "disabled";
   };
```

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SDMMC peripheral may connect to one SD card, one *e*•MMC™ device or one SDIO card.

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";      --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       st,neg-edge;                                          --> Generate data and
command on sdmmc clock falling edge
       st,sig-dir;                                           --> Allow to select
direction polarity of an external transceiver
       st,use-ckin;                                          --> Use sdmmc_ckin pin
from an external transceiver to sample the receive data
       bus-width = <4>;                                      --> Number of data lines,
can be 1, 4 or 8
       vmmc-supply = <&vdd_sd>;                              --> Supply node for
card's power
       vqmmc-supply = <&sd_switch>;                          --> Supply node for IO
line power
       status = "okay";                                      --> Enable the node
   };
```

Below optional properties have to be used when an external transceiver is connected:

- st,sig-dir: This property allows to select external transceiver direction signals polarity. When this property is set, the voltage transceiver IOs are driven as output when the direction signals are high. Without setting this property, the voltage transceiver IOs are driven as output when the direction signals are low.
- st,use-ckin: By setting this property, the sdmmc_ckin pin from an external transceiver is used to sample the receive data.

3.3 DT configuration examples

Below example shows how to configure the SDMMC when an *e*•MMC™ is connected with 8 data lines [4].

```
   &sdmmc2{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc2_b4_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-1 = <&sdmmc2_b4_od_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-2 = <&sdmmc2_b4_sleep_pins_a &sdmmc2_dir_sleep_pins_a>;
       non-removable;                                        --> Non-removable slot,
assume always present
       no-sd;                                                --> Avoid to send SD
command during initialization
       no-sdio;                                              --> Avoid to send SDIO
command during initialization
       st,neg-edge;
       bus-width = <8>;
       vmmc-supply = <&v3v3>;
       vqmmc-supply = <&vdd>;
       mmc-ddr-3_3v;                                         --> Host supports e•MMC™ D
DR 3.3V
       status = "okay";
   };
```

Below example shows how to configure the SDMMC to SD card (4 data lines) with an external transceiver [4].

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       broken-cd;                                            --> use polling mode for
card detection
       st,neg-edge;
       st,sig-dir;
       st,use-ckin;
       bus-width = <4>;
       sd-uhs-sdr12;                                         --> sd modes supported [1]
       sd-uhs-sdr25;
       sd-uhs-sdr50;
       sd-uhs-ddr50;
       sd-uhs-sdr104;
       vmmc-supply = <&vdd_sd>;
       vqmmc-supply = <&sd_switch>;
       status = "okay";
   };
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- [1.01.1] Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/mmc/mmci.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- [4.04.1] arch/arm/boot/dts/stm32mp157c-ed1.dts

Operating System

MultimediaCard

Device Tree

Secure digital

Generic Interrupt Controller

Serial Peripheral Interface

SD memory card (https://www.sdcard.org)

SDIO is an SD-size card with extended input/output functions

input/output

Doubledata rate (memory domain)

SDMMC device tree configuration

Stable: 11.06.2020 - 09:00 / Revision: 10.06.2020 - 15:35

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the SDMMC peripheral, used by the STM32 SDMMC Linux driver and by the MMC framework.

2 DT bindings documentation

The SDMMC device tree bindings are composed of:

- generic MMC device tree bindings [1].
- SDMMC MMC/SD/SDIO interface bindings [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 SDMMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
   sdmmc1: sdmmc@58005000 {                                  Comments
       compatible = "arm,pl18x", "arm,primecell";
       arm,primecell-periphid = <0x00253180>;
       reg = <0x58005000 0x1000>,                            --> The controller
register location
             <0x58006000 0x1000>;                            --> The delay block
register location
       interrupts = <GIC_SPI 49 IRQ_TYPE_LEVEL_HIGH>;        --> The interrupt number
used
       interrupt-names = "cmd_irq";
       clocks = <&rcc SDMMC1_K>;
       clock-names = apb_pclk
       resets = <&rcc SDMMC1_R>;
       status = "disabled";
   };
```

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SDMMC peripheral may connect to one SD card, one *e*•MMC™ device or one SDIO card.

```
   &sdmmc1{                                                   Comments
       pinctrl-names = "default", "opendrain", "sleep";       --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       st,neg-edge;                                           --> Generate data and
command on sdmmc clock falling edge
       st,sig-dir;                                            --> Allow to select
direction polarity of an external transceiver
       st,use-ckin;                                           --> Use sdmmc_ckin pin
from an external transceiver to sample the receive data
       bus-width = <4>;                                       --> Number of data lines,
can be 1, 4 or 8
       vmmc-supply = <&vdd_sd>;                               --> Supply node for
card's power
       vqmmc-supply = <&sd_switch>;                           --> Supply node for IO
line power
       status = "okay";                                       --> Enable the node
   };
```

Below optional properties have to be used when an external transceiver is connected:

- st,sig-dir: This property allows to select external transceiver direction signals polarity. When this property is set, the voltage transceiver IOs are driven as output when the direction signals are high. Without setting this property, the voltage transceiver IOs are driven as output when the direction signals are low.
- st,use-ckin: By setting this property, the sdmmc_ckin pin from an external transceiver is used to sample the receive data.

3.3 DT configuration examples

Below example shows how to configure the SDMMC when an *e*•MMC™ is connected with 8 data lines [4].

```
   &sdmmc2{                                                   Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc2_b4_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-1 = <&sdmmc2_b4_od_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-2 = <&sdmmc2_b4_sleep_pins_a &sdmmc2_dir_sleep_pins_a>;
       non-removable;                                         --> Non-removable slot,
assume always present
       no-sd;                                                  --> Avoid to send SD
command during initialization
       no-sdio;                                                --> Avoid to send SDIO
command during initialization
       st,neg-edge;
       bus-width = <8>;
       vmmc-supply = <&v3v3>;
       vqmmc-supply = <&vdd>;
       mmc-ddr-3_3v;                                           --> Host supports e•MMC™ D
DR 3.3V
       status = "okay";
   };
```

Below example shows how to configure the SDMMC to SD card (4 data lines) with an external transceiver [4].

```
   &sdmmc1{                                                    Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       broken-cd;                                              --> use polling mode for
card detection
       st,neg-edge;
       st,sig-dir;
       st,use-ckin;
       bus-width = <4>;
       sd-uhs-sdr12;                                           --> sd modes supported [1]
       sd-uhs-sdr25;
       sd-uhs-sdr50;
       sd-uhs-ddr50;
       sd-uhs-sdr104;
       vmmc-supply = <&vdd_sd>;
       vqmmc-supply = <&sd_switch>;
       status = "okay";
   };
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- 1.01.1 Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/mmc/mmci.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- 4.04.1 arch/arm/boot/dts/stm32mp157c-ed1.dts

Operating System

MultimediaCard

Device Tree

Secure digital

Generic Interrupt Controller

Serial Peripheral Interface

SD memory card (https://www.sdcard.org)

SDIO is an SD-size card with extended input/output functions

input/output

Doubledata rate (memory domain)

SDMMC device tree configuration

Stable: ***Not stable*** */ Revision: 12.06.2020 - 13:25*

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Article purpose

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

The configuration is performed using the device tree mechanism that provides a hardware description of the SDMMC peripheral, used by the STM32 SDMMC Linux driver and by the MMC framework.

2 DT bindings documentation

The SDMMC device tree bindings are composed of:

- generic MMC device tree bindings [1].
- SDMMC MMC/SD/SDIO interface bindings [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 SDMMC peripheral node is located in *stm32mp157c.dtsi*[3] file.

```
   sdmmc1: sdmmc@58005000 {                                  Comments
       compatible = "arm,pl18x", "arm,primecell";
       arm,primecell-periphid = <0x00253180>;
       reg = <0x58005000 0x1000>,                            --> The controller
register location
             <0x58006000 0x1000>;                            --> The delay block
register location
       interrupts = <GIC_SPI 49 IRQ_TYPE_LEVEL_HIGH>;        --> The interrupt number
used
       interrupt-names = "cmd_irq";
       clocks = <&rcc SDMMC1_K>;
       clock-names = apb_pclk
       resets = <&rcc SDMMC1_R>;
       status = "disabled";
   };
```

This device tree part is related to STM32 microprocessors. It should be kept as is, without being modified by the end-user.

3.2 DT configuration (board level)

The SDMMC peripheral may connect to one SD card, one *e*•MMC™ device or one SDIO card.

```
   &sdmmc1{                                                  Comments
       pinctrl-names = "default", "opendrain", "sleep";      --> For pinctrl
configuration, please refer to Pinctrl device tree configuration
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       st,neg-edge;                                             --> Generate data and
command on sdmmc clock falling edge
       st,sig-dir;                                              --> Allow to select
direction polarity of an external transceiver
       st,use-ckin;                                             --> Use sdmmc_ckin pin
from an external transceiver to sample the receive data
       bus-width = <4>;                                         --> Number of data lines,
can be 1, 4 or 8
       vmmc-supply = <&vdd_sd>;                                 --> Supply node for
card's power
       vqmmc-supply = <&sd_switch>;                             --> Supply node for IO
line power
       status = "okay";                                         --> Enable the node
   };
```

Below optional properties have to be used when an external transceiver is connected:

- st,sig-dir: This property allows to select external transceiver direction signals polarity. When this property is set, the voltage transceiver IOs are driven as output when the direction signals are high. Without setting this property, the voltage transceiver IOs are driven as output when the direction signals are low.
- st,use-ckin: By setting this property, the sdmmc_ckin pin from an external transceiver is used to sample the receive data.

3.3 DT configuration examples

Below example shows how to configure the SDMMC when an *e*•MMC™ is connected with 8 data lines [4].

```
   &sdmmc2{                                                     Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc2_b4_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-1 = <&sdmmc2_b4_od_pins_a &sdmmc2_dir_pins_a>;
       pinctrl-2 = <&sdmmc2_b4_sleep_pins_a &sdmmc2_dir_sleep_pins_a>;
       non-removable;                                           --> Non-removable slot,
assume always present
       no-sd;                                                   --> Avoid to send SD
command during initialization
       no-sdio;                                                 --> Avoid to send SDIO
command during initialization
       st,neg-edge;
       bus-width = <8>;
       vmmc-supply = <&v3v3>;
       vqmmc-supply = <&vdd>;
       mmc-ddr-3_3v;                                            --> Host supports e•MMC™ D
DR 3.3V
       status = "okay";
   };
```

Below example shows how to configure the SDMMC to SD card (4 data lines) with an external transceiver [4].

```
   &sdmmc1{                                                     Comments
       pinctrl-names = "default", "opendrain", "sleep";
       pinctrl-0 = <&sdmmc1_b4_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-1 = <&sdmmc1_b4_od_pins_a &sdmmc1_dir_pins_a>;
       pinctrl-2 = <&sdmmc1_b4_sleep_pins_a &sdmmc1_dir_sleep_pins_a>;
       broken-cd;                                               --> use polling mode for
card detection
       st,neg-edge;
```

```
        st,sig-dir;
        st,use-ckin;
        bus-width = <4>;
        sd-uhs-sdr12;                                          --> sd modes supported [1]
        sd-uhs-sdr25;
        sd-uhs-sdr50;
        sd-uhs-ddr50;
        sd-uhs-sdr104;
        vmmc-supply = <&vdd_sd>;
        vqmmc-supply = <&sd_switch>;
        status = "okay";
    };
```

4 How to configure the DT using STM32CubeMX

The STM32CubeMX tool can be used to configure the STM32MPU device and get the corresponding platform configuration device tree files.
The STM32CubeMX may not support all the properties described in the above DT bindings documentation paragraph. If so, the tool inserts **user sections** in the generated device tree. These sections can then be edited to add some properties and they are preserved from one generation to another. Refer to STM32CubeMX user manual for further information.

5 References

Please refer to the following links for additional information:

- [1.01.1] Documentation/devicetree/bindings/mmc/mmc.txt
- Documentation/devicetree/bindings/mmc/mmci.txt
- arch/arm/boot/dts/stm32mp157c.dtsi
- [4.04.1] arch/arm/boot/dts/stm32mp157c-ed1.dts

Operating System

MultimediaCard

Device Tree

Secure digital

Generic Interrupt Controller

Serial Peripheral Interface

SD memory card (https://www.sdcard.org)

SDIO is an SD-size card with extended input/output functions

input/output

Doubledata rate (memory domain)