LTDC device tree configuration

Stable: 04.05.2021 - 10:30 / Revision: 04.05.2021 - 10:32

A quality version of this page, approved on *4 May 2021*, was based off this revision.

Contents

1 Article purpose

This article explains how to configure the *LTDC* [1] **when the peripheral is assigned to the Linux®OS**.

The configuration is performed using the **device tree mechanism** [2].

The Device tree provides a hardware description of the LTDC [1] used by the STM32 *LTDC Linux driver*.

2 DT bindings documentation

The LTDC is represented by the STM32 LTDC device tree bindings [3].

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 LTDC device tree node is declared in stm32mp151.dtsi [4]. The declaration (shown below) provides the hardware registers base address, the clocks, the interrupts and the reset. Note: The port is also pre-populated to facilitate its use in the many other device tree files.

```
ltdc: display-controller@5a001000 {
        compatible = "st,stm32-ltdc";
        reg = <0x5a001000 0x400>;
        interrupts = <GIC_SPI 88 IRQ_TYPE_LEVEL_HIGH>,
                     <GIC_SPI 89 IRQ_TYPE_LEVEL_HIGH>;
        clocks = <&rcc LTDC_PX>;
        clock-names = "lcd";
        resets = <&rcc LTDC_R>;
        status = "disabled";

        port {
                #address-cells = <1>;
                #size-cells = <0>;
        };
};
```

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

3.2 DT configuration (board level)

The LTDC device tree related to a particular board may have the following nodes, depending on the board hardware:

- **ltdc** node: containing the LTDC pinctrl references and the in/out port descriptions
- **panel** or **i2cx** bridge rgb node (depending of the board hardware): containing the in/out port descriptions related to the LTDC node
- **panel_backlight** node (depending of the board hardware): related to the panel or bridge node

A full example of the LTDC pins [5] is available in the stm32mp15-pinctrl.dtsi [6]

```
ltdc_pins_a: ltdc-0 {
        pins {
                pinmux = <STM32_PINMUX('G',  7, AF14)>, /* LCD_CLK */
                         <STM32_PINMUX('I', 10, AF14)>, /* LCD_HSYNC */
                         <STM32_PINMUX('I',  9, AF14)>, /* LCD_VSYNC */
                         <STM32_PINMUX('F', 10, AF14)>, /* LCD_DE */
                         <STM32_PINMUX('H',  2, AF14)>, /* LCD_R0 */
                         <STM32_PINMUX('H',  3, AF14)>, /* LCD_R1 */
```

```
                                <STM32_PINMUX('H',  8, AF14)>, /* LCD_R2 */
                                <STM32_PINMUX('H',  9, AF14)>, /* LCD_R3 */
                                <STM32_PINMUX('H', 10, AF14)>, /* LCD_R4 */
                                <STM32_PINMUX('C',  0, AF14)>, /* LCD_R5 */
                                <STM32_PINMUX('H', 12, AF14)>, /* LCD_R6 */
                                <STM32_PINMUX('E', 15, AF14)>, /* LCD_R7 */
                                <STM32_PINMUX('E',  5, AF14)>, /* LCD_G0 */
                                <STM32_PINMUX('E',  6, AF14)>, /* LCD_G1 */
                                <STM32_PINMUX('H', 13, AF14)>, /* LCD_G2 */
                                <STM32_PINMUX('H', 14, AF14)>, /* LCD_G3 */
                                <STM32_PINMUX('H', 15, AF14)>, /* LCD_G4 */
                                <STM32_PINMUX('I',  0, AF14)>, /* LCD_G5 */
                                <STM32_PINMUX('I',  1, AF14)>, /* LCD_G6 */
                                <STM32_PINMUX('I',  2, AF14)>, /* LCD_G7 */
                                <STM32_PINMUX('D',  9, AF14)>, /* LCD_B0 */
                                <STM32_PINMUX('G', 12, AF14)>, /* LCD_B1 */
                                <STM32_PINMUX('G', 10, AF14)>, /* LCD_B2 */
                                <STM32_PINMUX('D', 10, AF14)>, /* LCD_B3 */
                                <STM32_PINMUX('I',  4, AF14)>, /* LCD_B4 */
                                <STM32_PINMUX('A',  3, AF14)>, /* LCD_B5 */
                                <STM32_PINMUX('B',  8, AF14)>, /* LCD_B6 */
                                <STM32_PINMUX('D',  8, AF14)>; /* LCD_B7 */
                        bias-disable;
                        drive-push-pull;
                        slew-rate = <1>;
                };
        };

        ltdc_sleep_pins_a: ltdc-sleep-0 {
                pins {
                        pinmux = <STM32_PINMUX('G',  7, ANALOG)>, /* LCD_CLK */
                                 <STM32_PINMUX('I', 10, ANALOG)>, /* LCD_HSYNC */
                                 <STM32_PINMUX('I',  9, ANALOG)>, /* LCD_VSYNC */
                                 <STM32_PINMUX('F', 10, ANALOG)>, /* LCD_DE */
                                 <STM32_PINMUX('H',  2, ANALOG)>, /* LCD_R0 */
                                 <STM32_PINMUX('H',  3, ANALOG)>, /* LCD_R1 */
                                 <STM32_PINMUX('H',  8, ANALOG)>, /* LCD_R2 */
                                 <STM32_PINMUX('H',  9, ANALOG)>, /* LCD_R3 */
                                 <STM32_PINMUX('H', 10, ANALOG)>, /* LCD_R4 */
                                 <STM32_PINMUX('C',  0, ANALOG)>, /* LCD_R5 */
                                 <STM32_PINMUX('H', 12, ANALOG)>, /* LCD_R6 */
                                 <STM32_PINMUX('E', 15, ANALOG)>, /* LCD_R7 */
                                 <STM32_PINMUX('E',  5, ANALOG)>, /* LCD_G0 */
                                 <STM32_PINMUX('E',  6, ANALOG)>, /* LCD_G1 */
                                 <STM32_PINMUX('H', 13, ANALOG)>, /* LCD_G2 */
                                 <STM32_PINMUX('H', 14, ANALOG)>, /* LCD_G3 */
                                 <STM32_PINMUX('H', 15, ANALOG)>, /* LCD_G4 */
                                 <STM32_PINMUX('I',  0, ANALOG)>, /* LCD_G5 */
                                 <STM32_PINMUX('I',  1, ANALOG)>, /* LCD_G6 */
                                 <STM32_PINMUX('I',  2, ANALOG)>, /* LCD_G7 */
                                 <STM32_PINMUX('D',  9, ANALOG)>, /* LCD_B0 */
                                 <STM32_PINMUX('G', 12, ANALOG)>, /* LCD_B1 */
                                 <STM32_PINMUX('G', 10, ANALOG)>, /* LCD_B2 */
                                 <STM32_PINMUX('D', 10, ANALOG)>, /* LCD_B3 */
                                 <STM32_PINMUX('I',  4, ANALOG)>, /* LCD_B4 */
                                 <STM32_PINMUX('A',  3, ANALOG)>, /* LCD_B5 */
                                 <STM32_PINMUX('B',  8, ANALOG)>, /* LCD_B6 */
                                 <STM32_PINMUX('D',  8, ANALOG)>; /* LCD_B7 */
                };
        };
```

A full example of the STM32MP15x Discovery board device tree is available in stm32mp15xx-dkx.dtsi [7].

```
&ltdc {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&ltdc_pins_a>;
        pinctrl-1 = <&ltdc_sleep_pins_a>;
        status = "okay";

        port {
                ltdc_ep0_out: endpoint@0 {
                        reg = <0>;
                        remote-endpoint = <&sii9022_in>;
                };
        };
};

&i2c1 {
...
        hdmi-transmitter@39 {
                compatible = "sil,sii9022";
                reg = <0x39>;
                iovcc-supply = <&v3v3_hdmi>;
                cvcc12-supply = <&v1v2_hdmi>;
                reset-gpios = <&gpioa 10 GPIO_ACTIVE_LOW>;
                interrupts = <1 IRQ_TYPE_EDGE_FALLING>;
                interrupt-parent = <&gpiog>;
                #sound-dai-cells = <0>;
                status = "okay";

                ports {
                        #address-cells = <1>;
                        #size-cells = <0>;

                        port@0 {
                                reg = <0>;
                                sii9022_in: endpoint {
                                        remote-endpoint = <&ltdc_ep0_out>;
                                };
                        };

                        port@3 {
                                reg = <3>;
                                sii9022_tx_endpoint: endpoint {
                                        remote-endpoint = <&i2s2_endpoint>;
                                };
                        };
                };
        };
};
```

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 LTDC internal peripheral
- Device tree
- Linux kernel STM32 LTDC bindings (st,stm32-ltdc.txt)
- Linux kernel STM32MP151 device tree (stm32mp151.dtsi)
- Pinctrl device tree configuration
- Linux kernel STM32MP15x pinctrl device tree (stm32mp15-pinctrl.dtsi)
- Linux kernel STM32MP15x Discovery board device tree (stm32mp15xx-dkx.dtsi)

LCD TFT Display Controller (STM32 specific)

Linux® is a registered trademark of Linus Torvalds.

Operating System

Device Tree

Generic Interrupt Controller

Serial Peripheral Interface

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)