Pinctrl device tree configuration

Pinctrl device tree configuration

Stable: 06.02.2020 - 14:42 / Revision: 06.02.2020 - 14:41

Template:ArticleMainWriter Template:ArticleApprovedVersion

Contents

1 Purpose

The purpose of this article is to explain how to configure the GPIO internal peripheral through **the pin controller (pinctrl) framework**, **when this peripheral is assigned to Linux® OS (Cortex-A)**. The configuration is performed using the Device tree.

To better understand I/O management, it is recommended to read the Overview of GPIO pins article.

This article also provides an example explaining how to add a new pin in the device tree.

2 DT bindings documentation

The Pinctrl device tree bindings are composed of:

- generic DT bindings[1] used by the pinctrl framework.
- vendor pinctrl DT bindings[2] used by the stm32-pinctrl driver: this binding document explains how to write device tree files for pinctrl.

3 DT configuration

3.1 DT configuration (STM32 level)

The pin controller node is located in the pinctrl dtsi file *stm32mp157-pinctrl.dtsi*[3]. See Device tree for more explanations about device tree file split. The pin controller node is composed of three parts:

3.1.1 STM32 pin controller information

```
pinctrl: pin-controller {                Comments
         #address-cells = <1>;
         #size-cells = <1>;
         ranges = <0 0x50002000 0xa400>;    -->Provides IP start address and memory map de
         interrupt-parent = <&exti>;        -->Provides interrupt parent controller (used
         st,syscfg = <&exti 0x60 0xff>;     -->Provides phandle for IRQ mux selection
         pins-are-numbered;

        ...
};
```

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

3.1.2 GPIO bank information

```
pinctrl: pin-controller {                Comments
         ...

        gpioa: gpio@50002000 {
                gpio-controller;
                #gpio-cells = <2>;
                interrupt-controller;       -->Indicates that this GPIO bank can be used as
                #interrupt-cells = <2>;
                reg = <0x0 0x400>;          -->Provides offset in pinctrl addres map for th
                clocks = <&rcc GPIOA>;      -->phandle on GPIO bank clock
                st,bank-name = "GPIOA";
                status = "disabled";
        };

        gpiob: gpio@50003000 {
                gpio-controller;
                #gpio-cells = <2>;
                interrupt-controller;
                #interrupt-cells = <2>;
                reg = <0x1000 0x400>;
                clocks = <&rcc GPIOB>;
```

```
                st,bank-name = "GPIOB";
                status = "disabled";
        };

        ...
};
```

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

3.1.3 Pin state definition

- Extract of *stm32mp157c-pinctrl.dtsi*[3] file:

```
pinctrl: pin-controller {
        ...
         usart3_pins_a: usart3@0 {                                                  Comme
                pins1 {
                        pinmux = <STM32_PINMUX('B', 10, AF7)>, /* USART3_TX */      -->Pin
                                 <STM32_PINMUX('G', 8, AF8)>; /* USART3_RTS */      -->Pin
                        bias-disable;                                               -->Gen
                        drive-push-pull;                                            -->Gen
                        slew-rate = <0>;                                            -->Gen
                };
                pins2 {
                        pinmux = <STM32_PINMUX('B', 12, AF8)>, /* USART3_RX */
                                 <STM32_PINMUX('I', 10, AF8)>; /* USART3_CTS_NSS */
                        bias-disable;
                };
       };
       ...
};
```

- Refer to GPIO internal peripheral for more details on hardware pin configuration.

3.2 DT configuration (board level)

As seen in Pin controller configuration (pin state definition part), all pin states are defined inside the pin controller node.

Each device that requires pins has to select the desired pin state phandle inside the board device tree file (see Device tree for more explanations about device tree file split).

The STM32MP1 devices feature a lot of possible pin combinations for a given internal peripheral. From one board to another, different sets of pins can consequently be used for an internal peripheral. Note that "_a", "_b" suffixes are used to identify pin muxing combinations in the device tree pinctrl file. The right suffixed combination must then be used in the device tree board file.

- Example:

```
&usart3 {
        ...
        pinctrl-names = "default","sleep";
        pinctrl-0 = <&usart3_pins_a>;
        pinctrl-1 = <&usart3_sleep_pins_a>;
        ...
};
```

3.3 DT configuration examples

3.3.1 How to add new pin states

To add new pin states and affect them to a foo_device, proceed as follows:

1. Find the pins you need:

   In the example below, the foo_device needs to configure PC13, PG8 and PI2.

   AF2 is selected as alternate function on PC13, and AF5 on PG8 and PI2.

   Each pin requires an internal pull-up.

2. Write your pin state phandle in *stm32mp157c-pinctrl.dtsi*.

```
pinctrl: pin-controller {
        ...
         foo_pins_a: foo@0 {
                pins {
                        pinmux = <STM32_PINMUX('C', 13, AF2)>,
                                 <STM32_PINMUX('G', 8, AF5)>,
                                  <STM32_PINMUX('I', 2, AF5)>;
                        bias-pull-up;
                };
        };
        ...
};
```

   All the possible settings are described in GPIO internal peripheral.

3. Select the pin state phandle required for your device in the board file.

```
&foo {
        ...
        pinctrl-names = "default";
        pinctrl-0 = <&foo_pins_a>;
        ...
};
```

4 How to configure GPIOs 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:

- â Documentation/devicetree/bindings/pinctrl/pinctrl-bindings.txt , Generic pinctrl device tree bindings
- â Documentation/devicetree/bindings/pinctrl/st,stm32-pinctrl.txt , STM32 pinctrl device tree bindings
- â 3.03.1 stm32mp157-pinctrl.dtsi STM32MP157C Pinctrl device tree file