gt9xx mtk platform pinctrl 使用方法

Revision

Rev	Change	Date	Author
V0.1	初版	2017-10-29	Lin

1、变量定义

```
struct pinctrl *pinctrl1;//定义 pinctrl 句柄
struct pinctrl_state *pins_default;//定义 pin 状态句柄
struct  pinctrl_state  *eint_as_int,  *eint_output0,  *eint_output1,
*rst_output0,
```

2、dts 配置实例

```
/* TOUCH start */
&touch {
        tpd-resolution = <720 1280>;
        use-tpd-button = <0>;
        tpd-key-num = <3>;
        tpd-key-local= <139 172 158 0>;
        tpd-key-dim-local = <90 883 100 40 230 883 100 40 370 883 100 40 0 0 0 0>;
        tpd-max-touch-num = <5>;
        tpd-filter-enable = <0>;
        tpd-filter-pixel-density = <146>;
        tpd-filter-custom-prameters = <0 0 0 0 0 0 0 0 0 0 0 0>;
        tpd-filter-custom-speed = <0 0 0>;
        pinctrl-names = "default", "state_eint_as_int", "state_eint_output0",
"state_eint_output1","state_rst_output0", "state_rst_output1";
        pinctrl-0 = <&ctp_pins_default>;
        pinctrl-1 = <&ctp_pins_eint_as_int>;
        pinctrl-2 = <&ctp_pins_eint_output0>;
        pinctrl-3 = <&ctp_pins_eint_output1>;
        pinctrl-4 = <&ctp_pins_rst_output0>;
        pinctrl-5 = <&ctp_pins_rst_output1>;
        status = "okay";
};
&pio {
        ctp_pins_default: eint0default {
        };
        ctp_pins_eint_as_int: ctp_eint@0 {
                pins_cmd_dat {
                        pins = <PINMUX_GPIO1__FUNC_GPIO1>;
                        slew-rate = <0>;//0 代表输入，1 代表输出,有些平台这个配置不生效，
                        input-enable;// 可加入 input-enable;
                        bias-disable;//取消上下拉
                };
        };
        ctp_pins_eint_output0: eintoutput0 {
                pins_cmd_dat {
                        pins = <PINMUX_GPIO1__FUNC_GPIO1>;
                        slew-rate = <1>;//输出
                        output-low;
                };
        };
        ctp_pins_eint_output1: eintoutput1 {
                pins_cmd_dat {
                        pins = <PINMUX_GPIO1__FUNC_GPIO1>;
                        slew-rate = <1>;
                        output-high;
                };
        };
        ctp_pins_rst_output0: rstoutput0 {
                pins_cmd_dat {
                        pins = <PINMUX_GPIO79__FUNC_GPIO79>;
                        slew-rate = <1>;
                        output-low;
                };
        };
        ctp_pins_rst_output1: rstoutput1 {
                pins_cmd_dat {
                        pins = <PINMUX_GPIO79__FUNC_GPIO79>;
                        slew-rate = <1>;
                        output-high;
                };
        };
};
/* TOUCH end */
```

3、pinctrl 初始化

3.1 获取 pinctrl 句柄

```
pinctrl1 = devm_pinctrl_get(&pdev->dev);//获取 pinctrl 句柄
if (IS_ERR(pinctrl1)) {
        ret = PTR_ERR(pinctrl1);
        dev_err(&pdev->dev, "xxx Cannot find touch pinctrl1!\n");
        return ret;
}
```

3.2 获取 pin 状态句柄

```
pins_default = pinctrl_lookup_state(pinctrl1, "default");//获取 pin 状态句柄
if (IS_ERR(pins_default)) {
        ret = PTR_ERR(pins_default);
        /* dev_err(&pdev->dev, "fwq Cannot find touch pinctrl default %d!\n", ret);*/
}
eint_as_int = pinctrl_lookup_state(pinctrl1, "state_eint_as_int");
if (IS_ERR(eint_as_int)) {
        ret = PTR_ERR(eint_as_int);
        dev_err(&pdev->dev, "fwq Cannot find touch pinctrl state_eint_as_int!\n");
        return ret;
}
eint_output0 = pinctrl_lookup_state(pinctrl1, "state_eint_output0");
if (IS_ERR(eint_output0)) {
        ret = PTR_ERR(eint_output0);
        dev_err(&pdev->dev, "fwq Cannot find touch pinctrl state_eint_output0!\n");
        return ret;
}
eint_output1 = pinctrl_lookup_state(pinctrl1, "state_eint_output1");
if (IS_ERR(eint_output1)) {
        ret = PTR_ERR(eint_output1);
        dev_err(&pdev->dev, "fwq Cannot find touch pinctrl state_eint_output1!\n");
        return ret;
}
rst_output0 = pinctrl_lookup_state(pinctrl1, "state_rst_output0");
if (IS_ERR(rst_output0)) {
        ret = PTR_ERR(rst_output0);
        dev_err(&pdev->dev, "fwq Cannot find touch pinctrl state_rst_output0!\n");
        return ret;
}
rst_output1 = pinctrl_lookup_state(pinctrl1, "state_rst_output1");
if (IS_ERR(rst_output1)) {
        ret = PTR_ERR(rst_output1);
        dev_err(&pdev->dev, "fwq Cannot find touch pinctrl state_rst_output1!\n");
        return ret;
}
```

4、接口封装使用

4.1 Goodix TP 需要用到的两个接口

```
void tpd_gpio_output(int pin, int level);//控制 INT、RST 输出高低点平
void tpd_gpio_as_int(int pin);//将 INTpin 设置为悬浮输入态(中断功能)
```

4.2 接口封装实现

```
void tpd_gpio_output(int pin, int level)//pin->INT(1) or RST(0)，level->输出电平
{
    mutex_lock(&tpd_set_gpio_mutex);
    TPD_DEBUG("[tpd]tpd_gpio_output pin = %d, level = %d\n", pin, level);
    if (pin == 1) {//INT pin
        if (level)
            pinctrl_select_state(pinctrl1, eint_output1);
        else
            pinctrl_select_state(pinctrl1, eint_output0);
    } else {//RST pin
        if (level)
            pinctrl_select_state(pinctrl1, rst_output1);
        else
            pinctrl_select_state(pinctrl1, rst_output0);
    }
    mutex_unlock(&tpd_set_gpio_mutex);
}
void tpd_gpio_as_int(int pin)
{
    mutex_lock(&tpd_set_gpio_mutex);
    TPD_DEBUG("[tpd]tpd_gpio_as_int\n");
    if (pin == 1)
        pinctrl_select_state(pinctrl1, eint_as_int);
    mutex_unlock(&tpd_set_gpio_mutex);
}
```