Rockchip
瑞芯微电子

Rockchip
PINCTRL 开发指南

发布版本:1.0

日期:2017.02

前言

概述

产品版本

芯片名称	内核版本
RK3228H	Linux3.10
RK3328	Linux3.10

读者对象

本文档（本指南）主要适用于以下工程师：
技术支持工程师
软件开发工程师

修订记录

日期	版本	作者	修改说明
2017-02-16	V1.0	WDC	初始版本

目录

1 PINCTRL 系统概述

RK3328/RK3228H Soc 内部包含了 pin 的控制器，通过对 pin 寄存器配置，我们可以配置一个或一组引脚的功能和特性。

在软件方面，Linux 内核中提供了 pinctrl 子系统，通过 pinctrl 的驱动可以操作 pin 的寄存器，用法为 Linux 通用接口，RK3328/RK3228H 的 pinctrl 的驱动完成如下工作：

A. 枚举并且命名所有引脚
B. 切换引脚的复用功能
C. 配置引脚的驱动强度
D. 配置引脚的上下拉功能
E. 配置引脚的 schmitter 触发能力

2 PINCTRL 配置

2.1 驱动文件与 DTS 配置

驱动文件所在位置：

drivers/pinctrl/pinctrl-rk3368.c; 注意，因为这套驱动是从 rk3368 上沿用下来，所以 pinctrl 驱动就是这个名字。

驱动 DTS 节点配置 pinctrl，驱动会将“default”，“idle”，“sleep”对应的这 3 组 pinctrl 配置解析储存起来，在特定的场合使用。Pinctrl-state.h 中给出了常用的 3 种状态：

default 状态表示设备处于 active 时的状态，一般在设备驱动的.resume 中配置，另外在启动时也会配置 pin 脚为 default 状态。

idle 状态表示系统处于 idle 时需要配置的 pin 脚状态，此时系统并没有进入深度休眠。

sleep 状态表示系统处于深度休眠时的 pin 脚状态，一般在设备驱动的.suspend 中配置。

例如，“default” 这组 pin 的配置会在驱动 probe 的时候，配置到寄存器里面，而其他组的配置需要在代码里面解析出来，再选择切换使用。例如 HDMI pinctrl 的 DTS 配置与代码中使用：

```
hdmi: hdmi@ff3c0000 {
    compatible = "rockchip,rk322xh-hdmi";
    reg = <0x0 0xff3c0000 0x0 0x20000>,
          <0x0 0xff430000 0x0 0x10000>;
    pinctrl-names = "default", "gpio";
    pinctrl-0 = <&hdmi_cec &hdmii2c_xfer &hdmi_hpd>;
    pinctrl-1 = <&i2c3_gpio &hdmi_cec>;
    rockchip,grf = <&grf>;
    rockchip,hdmi_audio_source = <0>;
    rockchip,hdcp_enable = <0>;
    rockchip,cec_enable = <0>;
    status = "disabled";
 };

hdmi_i2c{
    hdmi_i2c_xfer: hdmi-i2c-xfer {
        rockchip,pins =
            <0 A5 RK_FUNC_1&pcfg_pull_none_smt>,
            <0 A6 RK_FUNC_1&pcfg_pull_none_smt>;
```

```
        };

        hdmi_cec: hdmi-cec {
            rockchip,pins =
                <0 A3 RK_FUNC_1&pcfg_pull_none>;
            };
    };

    i2c3{
        i2c3_gpio: i2c3_gpio {
            rockchip,pins =
                <0 A5 RK_FUNC_GPIO &pcfg_pull_none>,
                <0 A6 RK_FUNC_GPIO &pcfg_pull_none>;
        };
    };
```

代码中使用：

驱动解析得到 gpio 状态，并切换成 gpio 模式：

```
gpio_state = pinctrl_lookup_state(hdmi_dev->dev->pins->p, "gpio");
pinctrl_select_state(hdmi_dev->dev->pins->p, gpio_state);
```

default 状态不需要驱动解析，直接切换成 default 模式：

```
pinctrl_select_state(hdmi_dev->dev->pins->p,
hdmi_dev->dev->pins->default_state);
```

2.2 IOMUX 配置

iomux 配置即切换 pin 所对应的 mux 值，其中有如下的定义，分别对应相应的寄存器值：

```
#define RK_FUNC_GPIO    0
#define RK_FUNC_1       1
#define RK_FUNC_2       2
#define RK_FUNC_3       3
#define RK_FUNC_4       4
#define RK_FUNC_5       5
#define RK_FUNC_6       6
#define RK_FUNC_7       7
```

下面仍然举例 hdmii2c_xfer；首先，我们在 3228H 的 trm 的 GRF 章节里面找到了 i2c3hdmi_scl 和 i2c3hdmi_sda 两个 pin 脚对应的是 gpio0a5 和 gpio0a6，两个功能脚都是二进制‘01’作为功能值，即使用 RK_FUNC_1；

如果硬件原理图上与所给参考代码定义 pin 引脚不同或者 mux 值不同时候，如何修改。假设现在有某个具体的产品是使用 i2c2 来连接 HDMI 作通讯功能，通过在该产品的 rk3228h-xxx.dts 引用覆盖来实现，下面为示例：

```
&hdmi_rk_fb {
    status = "okay";
    pinctrl-names = "default", "gpio";
    pinctrl-0 = <&i2c2_xfer &hdmi_cec>;
    pinctrl-1 = <&i2c2_gpio>;
};
```

2.3 驱动强度配置

驱动强度配置，即配置所对应的驱动强度电流值，分别对应相应的寄存器值，与 mux 用法类似，以下为示例：

```
pcfg_pull_down_12ma: pcfg-pull-down-12ma {
    bias-pull-down;
    drive-strength = <12>;
};

gmac {
    rgmii_pins: rgmii-pins {
        rockchip,pins =
             /* mac_txd1 */
            <3 5 RK_FUNC_1 &pcfg_pull_none_12ma>,
            /* mac_txd0 */
            <3 4 RK_FUNC_1 &pcfg_pull_none_12ma>,
            /* mac_rxd3 */
            <3 3 RK_FUNC_1 &pcfg_pull_none>,
            /* mac_rxd2 */
            <3 2 RK_FUNC_1 &pcfg_pull_none>,
            /* mac_txd3 */
            <3 1 RK_FUNC_1 &pcfg_pull_none_12ma>,
            /* mac_txd2 */
            <3 0 RK_FUNC_1 &pcfg_pull_none_12ma>;
    };
};
```

如果想增加或减少驱动强度，但是与 dtsi 定义的驱动强度不同时候，如何修改。同样类似 mux 的修改，在产品的 DTS 文件里面引用之后，修改覆盖。

每一个 pin 具有自己所在对应的驱动电流强度范围，所以配置的时候要选择其有效可配电流值，如果不是该 pin 对应的有效电流值，配置将会出错，无法生效。

2.4 上下拉配置

上下拉配置，即配置芯片 pad 内部上拉，下拉或者都不配置，分别对应相应的寄存器值，与 mux 用法类似，以下为示例：

```
pcfg_pull_up: pcfg-pull-up {
    bias-pull-up;
};

pcfg_pull_down: pcfg-pull-down {
    bias-pull-down;
};

pcfg_pull_none: pcfg-pull-none {
    bias-disable;
};

pcfg_pull_up_8ma: pcfg-pull-up-8ma {
    bias-pull-up;
    drive-strength = <8>;
};

uart1 {
    uart1_xfer: uart1-xfer {
        rockchip,pins =
            <3 12 RK_FUNC_2 &pcfg_pull_up>,
            <3 13 RK_FUNC_2 &pcfg_pull_none>;
    };
};
```

如果想配置成上拉，下拉或者都不配置，但是与所给参考代码定义的配置不同时候，如何修改。同样类似 mux 的修改，在产品的 DTS 文件里面引用之后，修改覆盖。

2.5 Schmitter 触发配置

某些 pin 脚需要配置 schmitt 功能，例如 i2c，32k 输入等，软件上提供这个功能的配置，同样是配置 dts，以下为示例:

```
pcfg_pull_none_smt: pcfg-pull-none-smt {
    bias-disable;
    input-schmitt-enable;
};

i2c1 {
    i2c1_xfer: i2c1-xfer {
        rockchip,pins =
            <2 GPIO_A4 RK_FUNC_2 &pcfg_pull_none_smt>,
            <2 GPIO_A5 RK_FUNC_2 &pcfg_pull_none_smt>;
```



```
        };
    };
```

2.6 常见问题

A. 如果 DTS 配置正确了，但是读寄存器配置不对，请确认该驱动是否有调用到 probe 函数；

B. 如果 DTS 配置正确了，但是读寄存器配置不对，且 a 已正确，请确认是否有 pinctrl 的错误 log 出现，例如：

C. 如果 DTS 配置正确了，但是读寄存器配置不对，且 a，b 都已正确，有可能被其他模块使用，一般都是切成 gpio，请搜索 DTS 文件，确认是否有其他模块使用该 pin 脚；如果 DTS 没有检索出来，可以在 drivers/pinctrl/pinctrl-rk3368.c 的 rockchip_set_mux()函数中加入打印，例如现在要找 GPIO1_B7 被哪个驱动所用，可以加入下面补丁，通过 dump_stack()看是否为正确调用：

```
    diff --git a/drivers/pinctrl/pinctrl-rk3368.c
b/drivers/pinctrl/pinctrl-rk3368.c
    index c6c04ac..c1dd0bd 100644
    --- a/drivers/pinctrl/pinctrl-rk3368.c
    +++ b/drivers/pinctrl/pinctrl-rk3368.c
    @@ -661,6 +661,11 @@ static
introckchip_set_mux(structrockchip_pin_bank *bank, int pin, int mux)
    dev_dbg(info->dev, "setting mux of GPIO%d-%d to %d\n",
    bank->bank_num, pin, mux);

    +       if ((bank->bank_num == 1) && (pin == 15)) {
    +               dev_err(info->dev,"setting mux of GPIO%d-%d
to %d\n", bank->bank_num, pin, mux);
    +               dump_stack();
    +       }
    +
    regmap = (bank->iomux[iomux_num].type & IOMUX_SOURCE_PMU)
                                    ? info->regmap_pmu :
info->regmap_base
```

RK3328/RK3228H 中的一些 pin 脚，除了正常的 iomux 设置外，还需要在里面做特别设置，详细部分请参阅 trm GRF 章节中 GRF_COM_IOMUX 相关寄存器描述。

3 GPIO 使用

pinctrl 的功能配置里面，最常使用的就是 gpio，下面介绍 gpio 使用方式；

3.1 DTS 配置与代码使用

以下面的声卡 es8316 为例，在 DTS 定义了两个 gpio，分别是 GPIO0_B3 和 GPIO4_D4:

```
es8316: es8316@10 {
    #sound-dai-cells = <0>;
    compatible = "everest,es8316";
    reg = <0x10>;
    clocks = <&cru SCLK_I2S_8CH_OUT>;
    clock-names = "mclk";
    spk-con-gpio = <&gpio0 11 GPIO_ACTIVE_HIGH>;
    hp-det-gpio = <&gpio4 28 GPIO_ACTIVE_LOW>;
};
```

代码中使用，spk-con-gpio 申请为 gpio 输出，另外一个 hp-det-gpio 申请为输入：

```
es8316->spk_ctl_gpio = of_get_named_gpio_flags(np,"spk-con-gpio",0,
&flags);
ret = devm_gpio_request_one(&i2c->dev, es8316->spk_ctl_gpio,
GPIOF_DIR_OUT, NULL);

es8316->hp_det_gpio = of_get_named_gpio_flags(np, "hp-det-gpio",0,
&flags);
ret = devm_gpio_request_one(&i2c->dev, es8316->hp_det_gpio,
GPIOF_IN, "hpdet");
```

下面就可以对输出的 gpio 做输出高或输出低操作，读输入 gpio 的高低电平。

3.2 GPIO 常见问题

A. 当使用 gpio request 时候，会将该 pin 的 mux 值强制切换为 gpio，所以使用该 pin 脚为 gpio 功能的时候确保该 pin 脚没有被其他模块所使用；

B. 如果用 io 命令读某个 gpio 的寄存器，读出来的值是异常，0x00000000 或 0xffffffff 等，请确认该 gpio 的 clk 是不是被关了，打开 clk，应该可以读到正确的寄存器；

C. 测量该 pin 脚的电压不对时，如果排除了外部因素，可以确认下该 pin 所在的 io 电压源是否正确，以及 io-domain 配置是否正确。

D. 如果使用该 gpio 时，不会动态的切换输入输出，建议在开始时就设置好 gpio

输出方向，后面拉高拉低时使用 gpio_set_value()接口，而不建议使用 gpio_direction_output(), 因为 gpio_direction_output 接口里面有 mutex 锁，对中断上下文调用会有错误异常，且相比 gpio_set_value，gpio_direction_output 所做事情更多，消耗更多时间。