全志科技
Allwinner Technology

A33

pinctrl 接口使用说明书

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2013-12-25		初始版本

目录

1. 引言

1.1. 编写目的

本文档主要介绍 pinctrl 接口使用方法。

1.2. 适应范围

硬件平台：A33

软件版本：Linux-3.4 及后续内核版本

1.3. 相关人员

本文档可供系统维护人员、驱动开发人员和测试人员参考。

2. pinctrl 子系统

2.1. 概述

许多 Soc 内部都包含 pin 控制器，通过 pin 控制器的寄存器，我们可以配置一个或者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，目的是为了统一各 SoC 厂商的 pin 脚管理，避免各 SoC 厂商各自实现相同的 pin 脚管理子系统，减少 SoC 厂商系统移植工作量。

通过 pinctrl 驱动可以操作 pin 控制器，完成如下工作：

1. 枚举并且命名 pin 控制器可控制的所有引脚；
2. 提供引脚的复用能力；
3. 提供配置引脚的能力，如驱动能力、上拉下拉和数据属性等；

基于 sunxi 平台实现的 pinctrl 驱动，是在 linux pinctrl 驱动通用框架上进行实现与扩张的，平台实现的 pinctrl 驱动除了拥有以上功能之外，还具有如下功能：

1. 与 gpio 子系统的交互；
2. 实现 pin 中断；

2.2. 内核配置

A33 平台默认选上 pinctrl 相关配置。Pinctrl 子系统调试信息配置如下：

Device Drivers --->

Pin controllers --->

[*] Debug PINCTRL calls

2.3. sys_config.fex

在 sys_config 中，gpio 描述形式为：

port:端口+组内序号<功能分配><内部电阻状态><驱动能力><输出电平>

1. 端口，例如：PA, PB, PC, ……
2. 组内序号，例如：0, 1, 2, ……
3. 功能分配，指定 pin 脚功能，参考 IC datasheet
4. 内部电阻状态，包括三种状态，0：上拉下拉禁用（默认），1：上拉，2：下拉
5. 驱动能力，可配驱动能力四级，分别是 0（默认），1，2，3
6. 输出电平，0 或者 1，只有当 pin 脚配成输出时才生效

示例：

```
[uart0]
uart_used           = 1
uart_port           = 0
uart_type           = 2
uart_tx             = port:PF2<4><1><default><default>
uart_rx             = port:PF4<4><1><default><default>
```

2.4. 模块加载

编进内核，无需加载。

2.5. 依赖资源

sys_config 模块。

3. pinctrl 接口

常用接口：

- devm_pinctrl_get_select_default

3.1. devm_pinctrl_get

函数原型	struct pinctrl * evm_pinctrl_get(struct device *dev);
函数功能	根据设备获取 pin 操作句柄，所以的 pin 操作必须基于此 pinctrl 句柄。与 pinctrl_get 接口功能完全一样，只是 devm_pinctrl_get 会将申请的 pinctrl 句柄做记账，绑定到设备句柄信息中。 设备驱动申请 pin 资源，推荐优先使用 devm_pinctrl_get 接口。
返回值	pinctrl 句柄
参数	dev：使用 pin 的设备，pinctrl 子系统会通过设备名与 pin 配置信息匹配。

3.2. devm_pinctrl_put

函数原型	void devm_pinctrl_put(struct pinctrl *p);
函数功能	释放 pinctrl 句柄，必须与 devm_pinctrl_get 配对使用。
返回值	无
参数	p：pinctrl 句柄

3.3. pinctrl_lookup_state

函数原型	struct pinctrl_state * pinctrl_lookup_state(struct pinctrl *p, const char *name);
函数功能	查找 pin 句柄指定状态下的状态句柄。
返回值	状态句柄
参数	p：pinctrl 句柄
	name：状态名称，A33 平台上只有 default 一种状态

3.4. pinctrl_select_state

函数原型	int pinctrl_select_state(struct pinctrl *p, struct pinctrl_state *s);
函数功能	设置 pin 句柄的状态到硬件。

返回值	0 表现成功，其他表示失败
参数	p：pinctrl 句柄
	s：状态句柄

3.5. devm_pinctrl_get_select

函数原型	struct pinctrl * devm_pinctrl_get_select(struct device *dev, const char *name);
函数功能	获取设备的 pin 操作句柄，并将 pinctrl 句柄设置为指定状态。
返回值	pinctrl 句柄
参数	dev：使用 pin 的设备，pinctrl 子系统会通过设备名与 pin 配置信息匹配。
	name：状态名称，A33 平台上只有 default 一种状态

3.6. devm_pinctrl_get_select_default

函数原型	struct pinctrl * devm_pinctrl_get_select_default(struct device *dev);
函数功能	获取设备的 pin 操作句柄，并将 pinctrl 句柄设置为 default 状态。
返回值	pinctrl 句柄
参数	dev：使用 pin 的设备，pinctrl 子系统会通过设备名与 pin 配置信息匹配。

3.7. pin_config_get

函数原型	int pin_config_get(const char *dev_name, const char *name, unsigned long *config);
函数功能	获取指定 pin 的属性。
返回值	0 表现成功，其他表示失败
参数	dev_name：使用 pin 的设备
	name：pin 的名称
	config：配置属性

3.8. pin_config_set

函数原型	int pin_config_set(const char *dev_name, const char *name, unsigned long config);
函数功能	设置指定 pin 的属性。
返回值	0 表现成功，其他表示失败
参数	dev_name：使用 pin 的设备
	name：pin 的名称
	config：配置属性

3.9. pin_config_group_get

函数原型	int pin_config_group_get(const char *dev_name, const char *pin_group, unsigned long *config);
函数功能	获取指定 group 的属性。
返回值	0 表现成功，其他表示失败

参数	dev_name：使用 pin 的设备
	pin_group：group 的名称
	config：配置属性

3.10. pin_config_group_set

函数原型	int pin_config_group_set(const char *dev_name, const char *pin_group, unsigned long config);
函数功能	设置指定 group 的属性。
返回值	0 表现成功，其他表示失败
参数	dev_name：使用 pin 的设备
	pin_group：group 的名称
	config：配置属性

4. 使用示例

4.1. 通过 sys_config 申请

设备需要的 pin 资源可以在 sys_confix 文件中配置，在 sunxi-pinctrl 初始化时会解析 sys_config 中的设备 pin 配置信息，并将设备的 pin 配置信息添加到 sunxi-pinctrl 模块中，设备驱动申请设备 pin 资源时只需告诉 pinctrl 设备名称即可，pinctrl 会自动将设备的 pin 资源信息绑定到该设备的 pinctrl 句柄中。

一般设备驱动只需要使用一个接口 devm_pinctrl_get_select_default 就可以申请到设备所有 pin 资源。

示例 1：

```
static int gmac_system_init(struct platform_device *pdev)
{
    int ret = 0;
    struct pinctrl *gmac_pin;
    gmac_pin = devm_pinctrl_get_select_default(&pdev->dev);
    if (IS_ERR_OR_NULL(gmac_pin))
        ret = -EINVAL;
    return ret;
}
```

示例 2：

```
static int twi_request_gpio(struct sunxi_i2c *i2c)
{
    int ret = 0;

    if (!twi_chan_is_enable(i2c->bus_num))
        return -1;
```

```
    i2c->pctrl = devm_pinctrl_get(i2c->adap.dev.parent);
    if (IS_ERR(i2c->pctrl)) {
        return -1;
    }

    i2c->pctrl_state = pinctrl_lookup_state(i2c->pctrl, PINCTRL_STATE_DEFAULT);
    if (IS_ERR(i2c->pctrl_state)) {
        return -1;
    }

    ret = pinctrl_select_state(i2c->pctrl, i2c->pctrl_state);

    return ret;
}

static void twi_release_gpio(struct sunxi_i2c *i2c)
{
    devm_pinctrl_put(i2c->pctrl);
}
```

4.2. 设备驱动直接申请

sunxi-pinctrl 可通过以下接口单独控制指定 pin 或 group 的相关属性。

- pin_config_set
- pin_config_get
- pin_config_group_set
- pin_config_group_get

目前 sunxi-pinctrl 平台支持配置 pin 以下四种属性。

- Mux
- Pull
- Driver-strength
- Output data

示例 1：

```
static int sunxi_pin_resource_req(struct gpio_config *pin_cfg)
{
    char pin_name[SUNXI_PIN_NAME_MAX_LEN];
    unsigned config;

    if (!IS_AXP_PIN(pin_cfg->gpio)) {
        sunxi_gpio_to_name(pin_cfg->gpio, pin_name);
        config = SUNXI_PINCFG_PACK(SUNXI_PINCFG_TYPE_FUNC, pin_cfg->mul_sel);
        pin_config_set(SUNXI_PINCTRL, pin_name, config);
        if (pin_cfg->pull != GPIO_PULL_DEFAULT) {
            config = SUNXI_PINCFG_PACK(SUNXI_PINCFG_TYPE_PUD, pin_cfg->pull);
```

```
                pin_config_set(SUNXI_PINCTRL, pin_name, config);
        }
        if (pin_cfg->drv_level != GPIO_DRVLVL_DEFAULT) {
                config = SUNXI_PINCFG_PACK(SUNXI_PINCFG_TYPE_DRV, pin_cfg->drv_level);
                pin_config_set(SUNXI_PINCTRL, pin_name, config);
        }
        if (pin_cfg->data != GPIO_DATA_DEFAULT) {
                config = SUNXI_PINCFG_PACK(SUNXI_PINCFG_TYPE_DAT, pin_cfg->data);
                pin_config_set(SUNXI_PINCTRL, pin_name, config);
        }
    } else {
        sunxi_gpio_to_name(pin_cfg->gpio, pin_name);
        config = SUNXI_PINCFG_PACK(SUNXI_PINCFG_TYPE_FUNC, pin_cfg->mul_sel);
        pin_config_set(AXP_PINCTRL, pin_name, config);
        if (pin_cfg->data != GPIO_DATA_DEFAULT) {
                config = SUNXI_PINCFG_PACK(SUNXI_PINCFG_TYPE_DAT, pin_cfg->data);
                pin_config_set(AXP_PINCTRL, pin_name, config);
        }
    }
}
```

4.3. 设备驱动中使用 pin 中断

目前 sunxi-pinctrl 使用 irq-domain 为 gpio 中断实现虚拟 irq 的功能，使用 gpio 中断功能时，设备驱动只需要通过 gpio_to_irq 获取虚拟中断号后，其他均可按照标准 irq 接口操作。

示例 1：

```
static int sunxi_gpio_eint_test(struct platform_device *pdev)
{
    struct device    *dev = &pdev->dev;
    int              virq;
    int              ret;

    /* map the virq of gpio */
    virq = gpio_to_irq(GPIOA(0));
    if (IS_ERR_VALUE(virq)) {
        pr_warn("map gpio [%d] to virq failed, errno = %d\n",
                GPIOA(0), virq);
        return -EINVAL;
    }
    pr_warn("gpio [%d] map to virq [%d] ok\n", GPIOA(0), virq);

    /* request virq, set virq type to high level trigger */
    ret = devm_request_irq(dev, virq, sunxi_gpio_irq_test_handler,
                           IRQF_TRIGGER_HIGH, "PA0_EINT", NULL);
    if (IS_ERR_VALUE(ret)) {
```

```
        pr_warn("request virq %d failed, errno = %d\n",
                virq, ret);
        return -EINVAL;
    }

    /* enbale virq */
    //enable_irq(virq);

    return 0;
}
```

4.4.注意事项

pinctrl 模块兼容 GPIO 的功能，如果 pin 是作为 GPIO input/ouput，仍然可以使用 gpiolib 中的标准接口，但是如果要使用 GPIO 的复用功能，则需要使用 pinctrl 接口。

5. FAQs

1. 调用 devm_pinctrl_get_select_default 返回错误

错误分析：pinctrl 初始化的时候，会去读取 sys_config 中 pin 的配置信息，建立 pin 的复用映射表和属性映射表。对映射表中的设备名，驱动初始化时会去读取有没有存在子健 device_name，如果不存在，则以主键名为设备名。

解决方法：当采用主键名作为设备名时，检查 sys_config 中主键名是否与需要申请 pin 资源的设备名一致。例如：

主键名：[uart_para2]，设备名：[uart2]，（错误）

主键名：[uart2]，设备名：[uart2]，（正确）

6. Declaration