Tina Linux PWM
开发指南

版本号: 1.1
发布日期: 2021.03.15

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.05.19	AWA1526	新建初始版本
1.1	2021.03.15	AWA1611	支持 Linux-5.4，R528

目　录

表　格

1 概述

1.1 编写目的

介绍全志 PWM 的使用方法。

1.2 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
R6	Linux-3.10	pwm-sunxi.c
R11	Linux-3.4	pwm-sunxi.c
R16	Linux-3.4	pwm-sunxi.c
R18	Linux-4.4	pwm-sunxi.c
R30	Linux-4.4	pwm-sunxi.c
R40	Linux-3.10	pwm-sunxi-new.c
R328	Linux-4.9	pwm-sunxi-new.c
R329	Linux-4.9	pwm-sunxi-group.c
R818	Linux-4.9	pwm-sunxi-group.c
MR813	Linux-4.9	pwm-sunxi-group.c
R528	Linux-5.4	pwm-sunxi-group.c

1.3 相关人员

PWM 驱动和应用开发人员。

2 模块介绍

2.1 源码结构说明

本模块借助于标准 Linux PWM 子系统。其代码路径为：

```
lichee/<linux-version>/drivers/pwm/pwm-sunxi.c       :  旧版本驱动
lichee/<linux-version>/drivers/pwm/pwm-sunxi-new.c   :  旧版本驱动
lichee/<linux-version>/drivers/pwm/pwm-sunxi-group.c :  新版本驱动
```

系统一共有三种 pwm 驱动，不同平台可能使用着不一样的驱动，配置也不一致。下面会分开三种 pwm 驱动来描述 pwm 驱动的配置方法。

2.2 模块配置说明

2.2.1 内核配置

在 tina 根目录下，执行 make kernel_menuconfig，进行内核驱动的配置。根据不同方案配置不同版本的驱动，配置路径分别如下三小节所示。

2.2.1.1 pwm-sunxi.c

```
Device Drivers
└─>Pulse-Width Modulation (PWM) Support
    └─>SUNXI PWM SELECT.
        └─>Sunxi PWM support
```

2.2.1.2 pwm-sunxi-new.c

```
Device Drivers
└─>Pulse-Width Modulation (PWM) Support
    └─>SUNXI PWM SELECT.
        └─>Sunxi Enhance PWM Support
```

2.2.1.3 pwm-sunxi-group.c

```
Device Drivers
└->Pulse-Width Modulation (PWM) Support
    └->SUNXI PWM SELECT.
        └->Sunxi PWM group support
```

2.2.2 sysconfig 配置

sys_config.fex 路径：

```
tina/device/config/chips/<chip>/configs/<board>/sys_config.fex
```

2.2.2.1 pwm-sunxi.c

pwm-sunxi.c 驱动 pwm 相关的配置如下：

```
[pwm0_para]
pwm_used            = 0
pwm_positive        = port:PH00<2><0><default><default>

[pwm1_para]
pwm_used            = 1
pwm_positive        = port:PH01<2><0><default><default>
```

2.2.2.2 pwm-sunxi-new.c

pwm-sunxi-new.c 驱动 pwm 相关的配置如下：

```
[pwm0]
pwm_used            = 0
pwm_positive        = port:PB2<3><0><default><default>

[pwm0_suspend]
pwm_used            = 1
pwm_positive        = port:PB2<7><0><default><default>
```

2.2.3 dts 配置

linux3.10 以上内核，才有 dts 相关的配置，主要是配置了寄存器地址。一般情况下，默认 SDK 中已经配置好，不需要做改动，只需要确认对应的引脚配置正确，按照第三节的描述就可以正常使用。

方案 dts 路径：

```
32位平台:tina/lichee/<linux-version>/arch/arm/boot/dts/<平台代号.dtsi>
64位平台:tina/lichee/<linux-version>/arch/arm64/boot/dts/sunxi/<平台代号.dtsi>
```

2.2.3.1 pwm-sunxi.c

pwm-sunxi.c 驱动对应的 dts 配置如下：

```
pwm: pwm@01c21400 {
        compatible = "allwinner,sunxi-pwm";
        reg = <0x0 0x01c21400 0x0 0x3c>;
        pwm-number = <1>;
        pwm-base = <0x0>;
        pwms = <&pwm0>;
};

pwm0: pwm0@01c21400 {
        compatible = "allwinner,sunxi-pwm0";
        pinctrl-names = "active", "sleep";
        pinctrl-0 = <&pwm0_pins_a>;
        pinctrl-1 = <&pwm0_pins_b>;
        reg_base = <0x01c21400>;
        reg_busy_offset = <0x00>; //以下的是寄存器配置
        reg_busy_shift = <28>;
        reg_enable_offset = <0x00>;
        reg_enable_shift = <4>;
        reg_clk_gating_offset = <0x00>;
        reg_clk_gating_shift = <6>;
        reg_bypass_offset = <0x00>;
        reg_bypass_shift = <9>;
        reg_pulse_start_offset = <0x00>;
        reg_pulse_start_shift = <8>;
        reg_mode_offset = <0x00>;
        reg_mode_shift = <7>;
        reg_polarity_offset = <0x00>;
        reg_polarity_shift = <5>;
        reg_period_offset = <0x04>;
        reg_period_shift = <16>;
        reg_period_width = <16>;
        reg_active_offset = <0x04>;
        reg_active_shift = <0>;
        reg_active_width = <16>;
        reg_prescal_offset = <0x00>;
        reg_prescal_shift = <0>;
        reg_prescal_width = <4>;
};
............../*省略，其他pwm配置*/
```

2.2.3.2 pwm-sunxi-new.c

pwm-sunxi-new.c 驱动对应的 dts 配置如下：

```
pwm: pwm@0300a000 {
        compatible = "allwinner,sunxi-pwm";
        reg = <0x0 0x0300a000 0x0 0x3ff>;
        clocks = <&clk_pwm>;
        pwm-number = <8>;
        pwm-base = <0x0>;
        pwms = <&pwm0>, <&pwm1>, <&pwm2>, <&pwm3>, <&pwm4>,
               <&pwm5>, <&pwm6>, <&pwm7>;
};

pwm0: pwm0@0300a000 {
        compatible = "allwinner,sunxi-pwm0";
        pinctrl-names = "active", "sleep";
        pinctrl-0 = <&pwm0_pins_a>;
        pinctrl-1 = <&pwm0_pins_b>;
        reg_base = <0x0300a000>;
        reg_peci_offset = <0x00>; //以下的是寄存器配置
        reg_peci_shift = <0x00>;
        reg_peci_width = <0x01>;

        reg_pis_offset = <0x04>;
        reg_pis_shift = <0x00>;
        reg_pis_width = <0x01>;

        reg_crie_offset = <0x10>;
        reg_crie_shift = <0x00>;
        reg_crie_width = <0x01>;

        reg_cfie_offset = <0x10>;
        reg_cfie_shift = <0x01>;
        reg_cfie_width = <0x01>;

        reg_cris_offset = <0x14>;
        reg_cris_shift = <0x00>;
        reg_cris_width = <0x01>;

        reg_cfis_offset = <0x14>;
        reg_cfis_shift = <0x01>;
        reg_cfis_width = <0x01>;
        reg_clk_src_offset = <0x20>;
        reg_clk_src_shift = <0x07>;
        reg_clk_src_width = <0x02>;

        reg_bypass_offset = <0x20>;
        reg_bypass_shift = <0x05>;
        reg_bypass_width = <0x01>;

        reg_clk_gating_offset = <0x20>;
        reg_clk_gating_shift = <0x04>;
        reg_clk_gating_width = <0x01>;

        reg_clk_div_m_offset = <0x20>;
        reg_clk_div_m_shift = <0x00>;
        reg_clk_div_m_width = <0x04>;

        reg_pdzintv_offset = <0x30>;
        reg_pdzintv_shift = <0x08>;
        reg_pdzintv_width = <0x08>;

        reg_dz_en_offset = <0x30>;
```

```
        reg_dz_en_shift = <0x00>;
        reg_dz_en_width = <0x01>;

        reg_enable_offset = <0x40>;
        reg_enable_shift = <0x00>;
        reg_enable_width = <0x01>;

        reg_cap_en_offset = <0x44>;
        reg_cap_en_shift = <0x00>;
        reg_cap_en_width = <0x01>;

        reg_period_rdy_offset = <0x60>;
        reg_period_rdy_shift = <0x0b>;
        reg_period_rdy_width = <0x01>;

        reg_pul_start_offset = <0x60>;
        reg_pul_start_shift = <0x0a>;
        reg_pul_start_width = <0x01>;
        reg_mode_offset = <0x60>;
        reg_mode_shift = <0x09>;
        reg_mode_width = <0x01>;

        reg_act_sta_offset = <0x60>;
        reg_act_sta_shift = <0x08>;
        reg_act_sta_width = <0x01>;

        reg_prescal_offset = <0x60>;
        reg_prescal_shift = <0x00>;
        reg_prescal_width = <0x08>;

        reg_entire_offset = <0x64>;
        reg_entire_shift = <0x10>;
        reg_entire_width = <0x10>;

        reg_active_offset = <0x64>;
        reg_active_shift = <0x00>;
        reg_active_width = <0x10>;

};
............../*其他pwm配置*/
```

2.2.3.3 pwm-sunxi-group.c in Linux-4.9

Linux-4.9 的 pwm-sunxi-group.c 驱动对应的方案 dts 配置如下：

```
pwm: pwm@0300a000 {
        compatible = "allwinner,sunxi-pwm";
        reg = <0x0 0x0300a000 0x0 0x3ff>;
        clocks = <&clk_pwm>;
        pwm-number = <16>;
        pwm-base = <0x0>;
        pwms = <&pwm0>, <&pwm1>, <&pwm2>, <&pwm3>, <&pwm4>,
                <&pwm5>, <&pwm6>, <&pwm7>, <&pwm8>, <&pwm9>,
        <&pwm10>, <&pwm11>, <&pwm12>, <&pwm13>,
        <&pwm14>, <&pwm15>;
};
```

```
s_pwm: pwm@07020c00 {
        compatible = "allwinner,sunxi-pwm";
        reg = <0x0 0x07020c00 0x0 0x3ff>;
        clocks = <&clk_spwm>;
        pwm-number = <1>;
        pwm-base = <16>;
        pwms = <&s_pwm0>;
};

pwm0: pwm0@0300a000 {
        compatible = "allwinner,sunxi-pwm0";
        pinctrl-names = "active", "sleep";
        reg_base = <0x0300a000>;
};
............../*其他pwm配置*/
```

其中，s_pwm 是带有 cpus 的平台才有的配置，不带 cpus 的平台不需要配置 s_pwm。

pwm 的引脚配置放置于板级 dts，路径为：

```
tina/device/config/chips/<平台名称>/configs/<board>/board.dts
```

boardd.dts 配置如下，这里以 pwm0 作为示例，其他 pwm 可参考 pwm0：

```
pio: pinctrl@0300b000 {
    ....
        pwm0_pin_a: pwm0@0 {
                allwinner,pins = "PD23";
                allwinner,function = "pwm0";
                allwinner,muxsel = <0x02>;
                allwinner,drive = <0x2>;
                allwinner,pull = <0>;
                allwinner,data = <0xffffffff>;
        };

        pwm0_pin_b: pwm0@1 {
                allwinner,pins = "PD23";
                allwinner,function = "io_disabled";
                allwinner,muxsel = <0x07>;
                allwinner,drive = <0x2>;
                allwinner,pull = <0>;
                allwinner,data = <0xffffffff>;
        };
    ....
};

pwm0: pwm0@0300a000 {
        pinctrl-names = "active", "sleep";
        pinctrl-0 = <&pwm0_pin_a>;
        pinctrl-1 = <&pwm0_pin_b>;
    status = "okay";
};
```

2.2.3.4 pwm-sunxi-group.c in Linux-5.4

另外，由于 Linux-5.4 中 clk 的配置改变了，因此 dts 的配置也有所修改。

Linux-5.4 的 pwm-sunxi-group.c 驱动对应的 dts 配置如下：

```
pwm: pwm@2000c00 {
        #pwm-cells = <0x3>;
        compatible = "allwinner,sunxi-pwm";
        reg = <0x0 0x02000c00 0x0 0x400>;
        clocks = <&ccu CLK_BUS_PWM>;
        resets = <&ccu RST_BUS_PWM>;
        pwm-number = <8>;
        pwm-base = <0x0>;
        sunxi-pwms = <&pwm0>, <&pwm1>, <&pwm2>, <&pwm3>, <&pwm4>,
                <&pwm5>, <&pwm6>, <&pwm7>;
};

pwm0: pwm0@2000c10 {
        compatible = "allwinner,sunxi-pwm0";
        pinctrl-names = "active", "sleep";
        reg = <0x0 0x02000c10 0x0 0x4>;
        reg_base = <0x02000c00>;
};
............../*其他pwm配置*/
```

board.dts 配置如下，这里以 pwm3 作为示例：

```
&pio {
    ....
        pwm3_pin_a: pwm3@0 {
                pins = "PB0";
                function = "pwm3";
                drive-strength = <10>;
                bias-pull-up;
        };

        pwm3_pin_b: pwm3@1 {
                pins = "PB0";
                function = "gpio_in";
                bias-disable;
        };
    ....
};

&pwm3 {
        pinctrl-names = "active", "sleep";
        pinctrl-0 = <&pwm3_pin_a>;
        pinctrl-1 = <&pwm3_pin_b>;
        status = "okay";
};
```

一般方案的 dts 已经是配置完成的，想要使用 pwm 的时候只需在 board.dts 配置好 pwm 通路以及对应的引脚，即可使用。

3 接口描述

3.1 驱动层使用说明

1、按照以下接口使用：

```
1. pwm_request:  申请pwm句柄
2. pwm_config:  配置pwm period & duty，注意单位是ns
3. pwm_set_polarity:  设置pwm的极性
4. pwm_enable:  使能pwm
```

2、不使用时：

```
1. pwm_disable:  关闭pwm
2. pwm_free:  释放pwm句柄
```

3、接口具体说明如下：

(1)pwm_request

表 3-1: pwm_request 接口说明表

类别	介绍
函数原型	struct pwm_device *pwm_request(int pwm_id, const char *label);
参数	pwm_id：pwm 的索引号，从 0 开始；label: 标签名
返回	成功返回 pwm 句柄，如果失败，则返回 NULL
功能描述	申请 pwm

(2)pwm_free

表 3-2: pwm_free 接口说明表

类别	介绍
函数原型	void pwm_free(struct pwm_device *pwm);
参数	pwm：pwm 句柄
返回	无返回值
功能描述	释放 pwm

(3)pwm_config

表 3-3: pwm_config 接口说明表

类别	介绍
函数原型	int pwm_config(struct pwm_device *pwm, int duty_ns, int period_ns)
参数	pwm: pwm 句柄。duty_ns: 有效区域时间, duty_ns / period_ns = 占空比。period_ns: pwm 的周期时间，单位为 ns
返回	成功则返回 0，失败则返回错误码
功能描述	配置 pwm 的周期以及占空比

(4)pwm_set_polarity

表 3-4: pwm_set_polarity 接口说明表

类别	介绍
函数原型	int pwm_set_polarity(struct pwm_device *pwm, enum pwm_polarity polarity);
参数	pwm: pwm 句柄。polarity: pwm 极性，PWM_POLARITY_NORMAL 为正常，高电平有效，PWM_POLARITY_INVERSED 为反转，即低电平有效
返回	成功则返回 0，失败则返回错误码
功能描述	配置 pwm 的周期以及占空比

(5)pwm_enable

表 3-5: pwm_enable 接口说明表

类别	介绍
函数原型	void pwm_enable(struct pwm_device *pwm);
参数	pwm：pwm 句柄
返回	成功则返回 0，失败则返回错误码
功能描述	使能 pwm

3.2 应用层使用说明

相关调试节点一般在/sys/class/pwm 目录下，以 R18 为例，它分别创建了两个 pwmchip, 对应 CPUX,CPUs 上面的 pwm 功能：

```
root@TinaLinux:/sys/class/pwm# ls
pwmchip0   pwmchip16
```

1、要使用 pwm，例如使用 CPUX 的 pwm0, 则按如下操作，生成 pwm0 目录：

```
root@TinaLinux:/# echo 0 > /sys/class/pwm/pwmchip0/export
root@TinaLinux:/# ls /sys/class/pwm/pwmchip0/pwm0/
capture     enable      polarity    uevent
duty_cycle  period      power
```

如果要使用 CPUX 的 pwm1，则写 1 进去节点。

说明

如果在驱动 *(例如 lcd 背光驱动)* 中已经申请过该 *pwm*，则这里再次申请 *(export)* 会提示*"Resource busy"*。

2、通过新增的 pwm0 目录下的节点来设置 pwm：

表 3-6: pwm 节点列表

节点	介绍
period	表示 pwm 的周期，单位 ns
duty_cycle	表示占空比，单位 ns
enable	表示是否使能 pwm
polarity	表示 pwm 极性 (normal/inversed)

使能 pwm 操作节点顺序可如下所示：

- period 可通过 "echo N > period" 写入数据，修改频率。
- duty_cycle 可以通过 "echo N > duty_cycle" 写入数据，修改占空比。占空比需要设置在频率之后。
- 最后，"echo 1 > enable" 来使能该通道的 pwm。

3、通过 cat 以下节点，可查看 pwm 使用情况：

```
root@TinaLinux:/# cat sys/kernel/debug/pwm
platform/7020c00.s_pwm, 1 PWM device
 pwm-0   ((null)              ): period: 0 ns duty: 0 ns polarity: normal

platform/300a000.pwm, 2 PWM devices
 pwm-0   (sysfs               ): requested period: 0 ns duty: 0 ns polarity: normal
 pwm-1   ((null)              ): period: 0 ns duty: 0 ns polarity: normal
```

说明

括号里的名称有以下几种方式：

- **在驱动层通过 *API* 接口 *pwm_request* 申请时传入参数标签名 *label* 来确定的，比如说 *lcd* 背光驱动的 *pwm* 节点 *"lcd"*；**
- **在应用层通过 *export* 节点使能的，显示为 *"sysfs"*；**
- **没有使能的 *pwm* 通道，显示为 *"(null)"*。**

4、通过编写代码来操作 pwm：操作 pwm 的节点与上述三小节的节点一样，不过操作的方式变成了：编写代码 open/fopen 打开 pwm 节点，write/fwrite 来向 pwm 节点写入数据等等。

简单的示例如下所示：

```
int pwm_setup()
{
    int ret, fd;
    fd = open("/sys/class/pwm/pwmchip0/export", O_WRONLY);
    if (fd < 0) {
        dbmsg("open export failed\n");
        return -1;
    }

    ret = write(fd, "0", strlen("0"));
    if(ret < 0) {
        dbmsg   ("creat pwm0 error\n");
        return -1;
    }

    return 0;
}
```

著作权声明

商标声明

免责声明