



D1 Tina Linux PWM 开发指南

版本号: 1.0
发布日期: 2021.04.09

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.09	AWA1611	新建初始版本



目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
2 模块介绍	2
2.1 源码结构说明	2
2.2 模块配置说明	2
2.2.1 内核配置	2
2.2.1.1 pwm-sunxi-group.c	2
2.2.2 dts 配置	2
3 接口描述	6
3.1 驱动层使用说明	6
3.2 应用层使用说明	7



表 格

1-1 适用产品列表	1
3-1 pwm_request 接口说明表	6
3-2 pwm_free 接口说明表	6
3-3 pwm_config 接口说明表	7
3-4 pwm_set_polarity 接口说明表	7
3-5 pwm_enable 接口说明表	7
3-6 pwm 节点列表	8



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
D1	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：

```
tina/lichee/linux-5.4/drivers/pwm/pwm-sunxi-group.c
```

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。

2.2.1.1 pwm-sunxi-group.c

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

2.2.2 dts 配置

通过命令 `cdts` 可以跳转到方案 dts 的路径。

方案 dts 路径：

```
tina/lichee/linux-5.4/arch/riscv/boot/dts/sunxi/sun20iw1p1.dtsi
```

pwm 配置如下：

```
pwm: pwm@2000c00 {
    #pwm-cells = <0x3>;
    compatible = "allwinner,sunxi-pwm";
    reg = <0x0 0x02000c00 0x0 0x3ff>;
    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";
    reg = <0x0 0x02000c10 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm1: pwm1@2000c11 {
    compatible = "allwinner,sunxi-pwm1";
    reg = <0x0 0x02000c11 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm2: pwm2@2000c12 {
    compatible = "allwinner,sunxi-pwm2";
    reg = <0x0 0x02000c12 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm3: pwm3@2000c13 {
    compatible = "allwinner,sunxi-pwm3";
    reg = <0x0 0x02000c13 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm4: pwm4@2000c14 {
    compatible = "allwinner,sunxi-pwm4";
    reg = <0x0 0x02000c14 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm5: pwm5@2000c15 {
    compatible = "allwinner,sunxi-pwm5";
    reg = <0x0 0x02000c15 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm6: pwm6@2000c16 {
    compatible = "allwinner,sunxi-pwm6";
    reg = <0x0 0x02000c16 0x0 0x4>;
    reg_base = <0x02000c00>;
};
pwm7: pwm7@2000c17 {
    compatible = "allwinner,sunxi-pwm7";
    reg = <0x0 0x02000c17 0x0 0x4>;
    reg_base = <0x02000c00>;
};
};
```

板级 dts 主要是配置 pwm 的引脚设置，通过 cconfigs 可以跳转到板级 dts 的路径下：

```
tina/device/config/chips/d1/configs/nezha/linux/board.dts
```

board.dts 配置如下所示：

```
&pio {
    .....//省略其他模块的引脚设置
    pwm0_pin_a: pwm0@0 {
        pins = "PD16";
        function = "pwm0";
        drive-strength = <10>;
        bias-pull-up;
    };

    pwm0_pin_b: pwm0@1 {
        pins = "PD16";
        function = "gpio_in";
        bias-disable;
    };

    pwm2_pin_a: pwm2@0 {
        pins = "PD18";
        function = "pwm2";
        drive-strength = <10>;
        bias-pull-up;
    };

    pwm2_pin_b: pwm2@1 {
        pins = "PD18";
        function = "gpio_in";
    };

    pwm7_pin_a: pwm7@0 {
        pins = "PD22";
        function = "pwm7";
        drive-strength = <10>;
        bias-pull-up;
    };

    pwm7_pin_b: pwm7@1 {
        pins = "PD22";
        function = "gpio_in";
    };
    .....//省略其他模块的引脚设置
};

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

&pwm2 {
    pinctrl-names = "active", "sleep";
    pinctrl-0 = <&pwm2_pin_a>;
    pinctrl-1 = <&pwm2_pin_b>;
    status = "okay";
};

&pwm7 {
    pinctrl-names = "active", "sleep";
    pinctrl-0 = <&pwm7_pin_a>;
    pinctrl-1 = <&pwm7_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, 它创建了一个 pwmchip, 对应 CPUX 上面的 pwm 功能:

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

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”；
- 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：

```
1 int pwm_setup()
2 {
3     int ret, fd;
4     fd = open("/sys/class/pwm/pwmchip0/export", O_WRONLY);
5     if (fd < 0) {
6         dbmsg("open export failed\n");
7         return -1;
8     }
9
10    ret = write(fd, "0", strlen("0"));
11    if(ret < 0) {
12        dbmsg ("creat pwm0 error\n");
13        return -1;
14    }
15
16    return 0;
17 }
```






著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本档内容的部分或全部，且不得以任何形式传播。

商标声明

、 **全志科技** （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本档作为使用指导仅供参考。由于产品版本升级或其他原因，本档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本档中提供准确的信息，但并不确保内容完全没有错误，因使用本档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。