



Tina Linux LEDC 开发指南

版本号: 1.2
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 | 2019.03.13 | AWA1526 | 初始版本 |
| 1.1 | 2020.06.10 | AWA1526 | 支持 R329, R818, MR813 |
| 1.2 | 2021.03.15 | AWA1611 | 支持 Linux-5.4 内核, R528 平台 |



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 相关术语说明 | 2 |
| 2.2 源码结构说明 | 2 |
| 2.3 模块配置说明 | 2 |
| 2.3.1 内核配置 | 2 |
| 2.3.2 DTS 配置 | 4 |
| 2.3.2.1 DTS 路径 | 4 |
| 2.3.2.2 DTS 文件 | 4 |
| 2.3.3 sys_config.fex 配置 | 6 |
| 3 接口描述 | 7 |
| 3.1 内部接口 | 7 |
| 3.2 外部接口 | 7 |
| 3.2.1 brightness 调节说明 | 7 |
| 3.2.2 led trigger 使用说明 | 8 |
| 3.2.3 debugfs 使用说明 | 8 |

1 概述

1.1 编写目的

介绍全志 LEDC 驱动的使用方法，方便 LEDC 驱动维护和应用开发。

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 |
|-------|-----------|--------------|
| R328 | Linux-4.9 | leds-sunxi.c |
| R328 | Linux-4.9 | leds-sunxi.c |
| R818 | Linux-4.9 | leds-sunxi.c |
| MR813 | Linux-4.9 | leds-sunxi.c |
| R528 | Linux-5.4 | leds-sunxi.c |

1.3 相关人员

LEDC 驱动和应用开发人员。

2 模块介绍

2.1 相关术语说明

表 2-1: 术语说明表

| 术语 | 说明 |
|------|---------------------------------|
| LED | Light Emitting Diode |
| LEDC | Light Emitting Diode Controller |

2.2 源码结构说明

本模块借助于标准 Linux LED 子系统。其代码路径为：

Linux-4.9内核: tina/lichee/linux4.9/drivers/leds/
Linux-5.4内核: tina/lichee/linux5.4/drivers/leds/

主要包含以下部分代码：

led-core.c: 为led子系统的核心文件。
ledtrigger-xxx.c: 为trigger相关的文件。
leds-sunxi.c: LEDC驱动实现代码。
leds-sunxi.h: 定义全志LEDC驱动数据结构。

2.3 模块配置说明

2.3.1 内核配置

在 tina 根目录下，执行 make kernel_menuconfig，配置路径如下：

Device Drivers
└->LED_Support
 └->LED support for Allwinner platforms

操作图示：

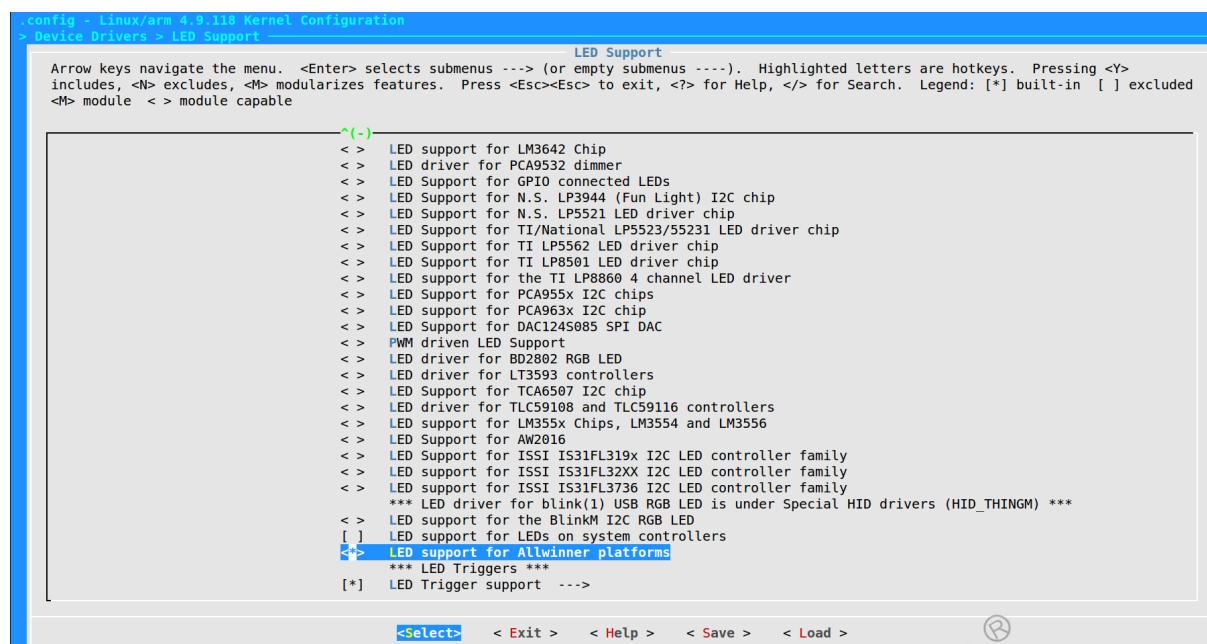


图 2-1: ledc 配置界面

如果需要用到 trigger 的话，需要选择相对应的配置项。配置路径如下：

```
Device Drivers
└->LED_Support
    └->LED Trigger support
```

操作图示：

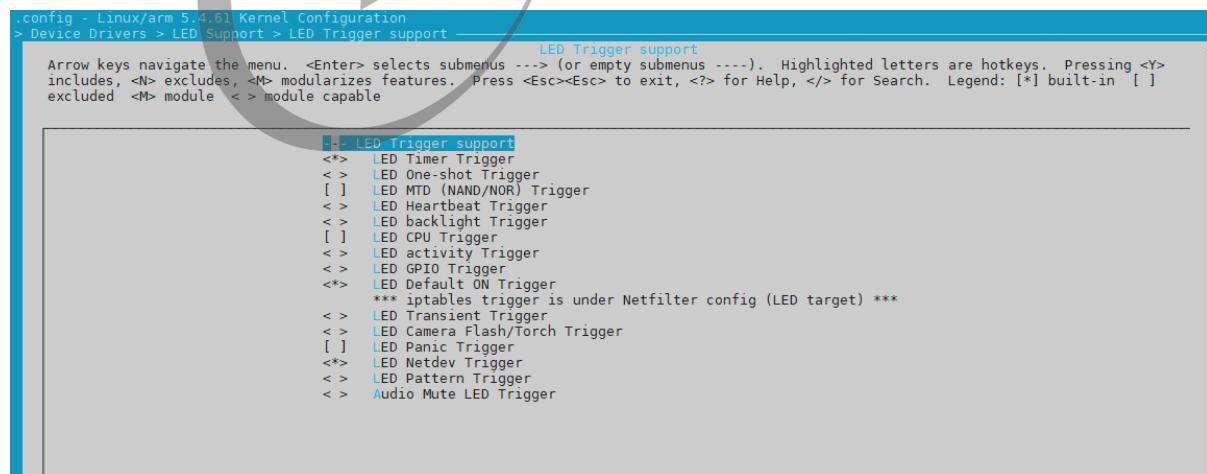


图 2-2: trigger 配置界面

2.3.2 DTS 配置

2.3.2.1 DTS 路径

通过 cdts 命令可跳转到平台 dts 路径，

对于 Linux-4.9 来说：

```
tina/lichee/linux-4.9/arch/arm/boot/dts/<平台代号.dtsi>
tina/lichee/linux-4.9/arch/arm64/boot/dts/sunxi/<平台代号.dtsi>
```

对于 Linux-5.4 来说：

```
tina/lichee/linux-5.4/arch/arm/boot/dts/<平台代号.dtsi>
```

板极相关配置 dts 路径：

通过 cconfigs 命令可跳转到板级 linux 配置路径，

其中，对于 Linux-4.9 来说，board.dts 在板级 linux 配置路径的上一级路径：

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

对于 Linux-5.4 来说，board.dts 就在板级 linux 配置路径下：

```
tina/device/config/chips/<chip>/configs/<board>/linux/board.dts
```

2.3.2.2 DTS 文件

Linux-4.9：

```
ledc@06700000 {
    compatible = "allwinner,sunxi-leds";
    reg = <0x0 0x06700000 0x0 0x50>;
    interrupts = <GIC_SPI 60 IRQ_TYPE_LEVEL_HIGH>;
    interrupt-names = "ledcirq";
    clocks = <&clk_ledc>, <&clk_cpuapb>;
    clock-names = "clk_ledc", "clk_cmuapb";
    pinctrl-names = "default";
    pinctrl-0 = <&ledc_pins_a>;
    led_count      = <32>;           -> LED的数量
    output_mode    = "GRB";           -> 输出模式，默认为GRB
    trans_mode     = "DMA";           -> 数据传输模式，默认为DMA
    reset_ns       = <42>;          -> LED reset时间
    t1h_ns         = <42>;          -> 1码高电平时间
    t1l_ns         = <42>;          -> 1码低电平时间
    t0h_ns         = <42>;          -> 0码高电平时间
    t0l_ns         = <42>;          -> 0码低电平时间
    wait_time0_ns  = <42>;          -> 相邻两个LED数据之间等待的时间
    wait_time1_ns  = <42>;          -> 相邻两帧数据之间等待的时间
```

```
    wait_data_time_ns = <20000000>; -> LEDC内部FIFO等待数据的时间容忍度
};
```

Linux-5.4 中，clk 和 dma 配置有所变化。并且，代码编写文件区分明确，属于 LEDC 的固定配置写在平台 dts 里，而由 LEDC 所控制的 LED 灯的详细参数、PIN 脚等设置则写在板级 dts 里。

平台 dts：

```
ledc: ledc@2008000 {
    #address-cells = <1>;
    #size-cells = <0>;
    compatible = "allwinner,sunxi-leds";
    reg = <0x0 0x02008000 0x0 0x400>;
    interrupts = <GIC_SPI 20 IRQ_TYPE_LEVEL_HIGH>;
    interrupt-names = "ledcirq";
    clocks = <&ccu CLK_LEDCE>, <&ccu CLK_BUS_LEDCE>;
    clock-names = "clk_ledc", "clk_cpuapb";
    dmams = <&dma 42>, <&dma 42>;
    dma-names = "rx", "tx";
    resets = <&ccu RST_BUS_LEDCE>;
    reset-names = "ledc_reset";
    status = "disabled";
};
```

板级 dts：

```
&pio {
    .....//省略
    ledc_pins_a: ledc@0 {
        pins = "PF2";
        function = "ledc";
        drive-strength = <10>;
    };

    ledc_pins_b: ledc@1 {
        pins = "PF2";
        function = "gpio_in";
    };
    .....//省略
};

&ledc {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&ledc_pins_a>;
    pinctrl-1 = <&ledc_pins_b>;
    led_count = <12>;
    output_mode = "GRB";
    reset_ns = <84>;
    t1h_ns = <800>;
    t1l_ns = <320>;
    t0h_ns = <300>;
    t0l_ns = <800>;
    wait_time0_ns = <84>;
    wait_time1_ns = <84>;
    wait_data_time_ns = <600000>;
    status = "disabled";
};
```

{;

Linux-4.9 和 Linux-5.4 的 dts 配置含义基本一致，详细说明如下所示：

- pinctrl-names: 用于表示 0 和 1 的 pinctrl 哪个是默认和休眠状态。
- pinctrl-0: 引脚配置，这里是默认使用的时候配置。
- pinctrl-1: 同上，这里是休眠时的配置。
- led_count: LED 灯的数目，根据硬件配置。
- output_mode: LED 灯输出模式，根据 LED 灯的 datasheet 进行配置。
- reset_ns: LED 灯 reset 时间控制。
- t1h_ns: 1 码高电平时间，根据 LED 灯的 datasheet 进行配置。
- t1l_ns: 1 码低电平时间，根据 LED 灯的 datasheet 进行配置。
- t0h_ns: 0 码高电平时间，根据 LED 灯的 datasheet 进行配置。
- t0l_ns: 0 码低电平时间，根据 LED 灯的 datasheet 进行配置。
- wait_time0_ns: 两个 LED 数据之间的等待时间，根据 LED 灯的 datasheet 进行配置。
- wait_time1_ns: 帧数据之间的等待时间，根据 LED 灯的 datasheet 进行配置。
- wait_data_time_ns: 内部 FIFO 等待数据时间，超过时间触发异常中断。
- status: 设备状态。

通常，如果想要使用一款新的 LEDC 灯，需要确认上述全部配置项都配置正确，比如说引脚配置以及 LED 灯的参数配置（包括 01 码高低电平时间、reset 时间以及 wait 时间），全部配置正确才能成功点亮。

2.3.3 sys_config.fex 配置

sys_config.fex 路径：

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

```
[ledc]
ledc_used      = 1           ->是否使用LEDC
ledc            = port:PE02<2><default><default><default>    ->LEDC对应的引脚
led_count       = 1           ->LED数量
output_mode     = "RGB"       -> 输出模式，默认为GRB
t1h_ns          = 800         -> 1码高电平时间
t1l_ns          = 450         -> 1码低电平时间
t0h_ns          = 400         -> 0码高电平时间
t0l_ns          = 850         -> 0码低电平时间
```

sys_config 的配置含义参考上节 dts 的。

说明

sys_config 的配置优先级高于 **dts** 的配置。

另外，**Linux-5.4** 中不继续使用 **sys_config** 来配置 **LEDC** 了。

3 接口描述

3.1 内部接口

LEDC 驱动主要的内部接口如下表所示：

表 3-1: 内部接口功能列表

| 内部接口功 | 能 |
|--|---|
| <code>sunxi_ledc_set_length</code> | 设置 LED 的数量 |
| <code>sunxi_ledc_set_output_mode</code> | 设置 LEDC 的输出模式 (R、G、B 的排布顺序) |
| <code>sunxi_ledc_set_cpu_mode</code> | 设置 CPU 的传输模式 |
| <code>sunxi_ledc_set_dma_mode</code> | 设置 DMA 的传输模式 |
| <code>sunxi_ledc_enable</code> | 使能 LEDC |
| <code>sunxi_ledc_trans_data</code> | 设置 LEDC 相关寄存器；将 RGB 数据搬到 LEDC FIFO 中，启动 LEDC |
| <code>sunxi_ledc_set_time</code> | 模块初始化时设置 <code>reset_ns</code> 、 <code>t1h_ns</code> 、 <code>t1l_ns</code> 等的时间 |
| <code>sunxi_ledc_reset</code> | 将 <code>transmitted_data</code> 置为 0；释放系统资源；对 LEDC 做 soft reset 操作 |
| <code>sunxi_set_led_brightness</code> | 设置 LED 亮度，范围为 0~255 |
| <code>sunxi_register_led_classdev</code> | 模块初始化时注册 <code>led_classdev</code> 设备 |
| <code>sunxi_unregister_led_classdev</code> | 模块卸载时注销 <code>led_classdev</code> 设备 |

3.2 外部接口

3.2.1 brightness 调节说明

每个 RGB LED 在`/sys/class/leds` 目录下对应有 3 个 `led_classdev` 设备目录，分别如下：

```
/sys/class/leds/sunxi_led[n]r  
/sys/class/leds/sunxi_led[n]g  
/sys/class/leds/sunxi_led[n]b
```

其中 n 表示 LED 的编号，n 最小值为 0。

说明

注意：从 **LEDC PIN** 端开始数，第一个 **LED** 的编号为 **0**，沿着远离 **PIN** 端的方向 **LED** 的编号依次递增。

例如，调节第 0 个 LED 的颜色为白光且最亮，操作如下：

```
echo 255 > /sys/class/leds/sunxi_led0r/brightness  
echo 255 > /sys/class/leds/sunxi_led0g/brightness  
echo 255 > /sys/class/leds/sunxi_led0b/brightness
```

3.2.2 led trigger 使用说明

通过 “/sys/class/leds/[device]/trigger” 来设置 trigger 类型。

Trigger 类型有：backlight、camera、cpu、default-on、disk、gpio、heartbeat、mtd、oneshot、panic、timer、transient。

例如设置 trigger 类型为 timer，操作如下：

```
echo timer > /sys/class/leds/sunxi_led0r/trigger
```

技巧

注意：trigger 类型为 timer 时，默认亮 500ms，灭 500ms。可通过以下节点设置亮和灭持续的时间。

```
/sys/class/leds/[device]/delay_on  
/sys/class/leds/[device]/delay_off
```

3.2.3 debugfs 使用说明

LEDC 相关的 debugfs 文件节点所在目录为 /sys/kernel/debug/sunxi_leds，节点说明如下：

- **reset_ns**：通过该节点可设置和读取 LED 的 reset 时间，范围为 80ns-327us。
- **t1h_ns**：通过该节点可设置和读取 1 码高电平时间，范围为 80ns-2560ns。
- **t1l_ns**：通过该节点可设置和读取 1 码低电平时间，范围为 80ns-1280ns。
- **t0h_ns**：通过该节点可设置和读取 0 码高电平时间，范围为 80ns-1280ns。
- **t0l_ns**：通过该节点可设置和读取 1 码低电平时间，范围为 80ns-2560ns。
- **wait_time0_ns**：通过该节点可设置和读取相邻两个 LED 数据之间等待的时间，范围为 80ns-10us。
- **wait_time1_ns**：通过该节点可设置和读取相邻两帧数据之间等待的时间，范围为 80ns-85s。

- `wait_data_time_ns`: 通过该节点可设置和读取 LEDC 内部 FIFO 等待数据的时间容忍度，范围为 80ns-655us。
- `data`: 通过该节点可读取 data buffer 中的数据，即所有 LED 对应的数据。
- `output_mode`: 通过该节点可设置和读取当前输出的模式，输出模式有 GRB、GBR、RGB、RBG、BGR 和 BRG。
- `trans_mode`: 通过该节点可设置和读取当前的数据传输模式（CPU 或 DMA）。
- `hwversion`: 通过该节点可查看当前 LEDC 的硬件版本。

⚠ 警告

- (1) 设置的时间必须在所说明的时间范围内，否则不会做任何操作。
- (2) 最终设置寄存器之后得到的时间均为 42ns 的整数倍，若通过节点设置的时间不遵循 42ns 的整数倍，则实际所设置的时间为小于该值的最大能够被 42ns 整除的数。例如通过 `reset_ns` 设置 90ns，则设置成功之后的 LED `reset` 时间为 84ns。

debugfs 使用举例如下：

```
echo 84 > /sys/kernel/debug/sunxi_leds/reset_ns
cat /sys/kernel/debug/sunxi_leds/data
echo RGB > /sys/kernel/debug/sunxi_leds/output_mode
cat /sys/kernel/debug/sunxi_leds/hwversion
```

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。

本文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。

本文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、发表或传播本文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。在本文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。本文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，并严格遵循本文档的使用说明。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。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（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。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文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。