

深圳市慧澜科技有限公司MaCo\_Yao



深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

# Linux PWM 开发指南

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

版本号: 1.0  
发布日期: 2021.9.06

深圳市慧澜科技有限公司MaCo\_Yao

深圳市慧澜科技有限公司MaCo\_Yao

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.9.06	XAA0192	1. 创建该文档

# 目 录

<b>1 概述</b>	<b>1</b>
1.1 编写目的	1
1.2 使用范围	1
1.3 相关人员	1
<b>2 术语及概念</b>	<b>2</b>
2.1 术语定义及缩略语	2
2.2 概念阐述	2
<b>3 模块描述</b>	<b>3</b>
3.1 模块功能	3
3.2 模块位置	3
3.3 模块配置	4
3.3.1 linux-4.9	4
3.3.2 linux-5.4	6
3.4 设备树配置	9
3.4.1 linux-4.9	9
3.4.2 linux-5.4	11
3.5 源码结构	13
3.5.1 linux-4.9	13
3.5.2 linux-5.4	13
3.6 调试接口	13

## 插图

图 3-1	模块功能	3
图 3-2	Device Drivers	4
图 3-3	Pulse-Width Modulation (PWM) Support	5
图 3-4	SUNXI PWM SELECT	5
图 3-5	Sunxi Enhance PWM support	6
图 3-6	Device	7
图 3-7	Pulse-Width Modulation (PWM) Support	7
图 3-8	SUNXI PWM SELECT	8
图 3-9	Sunxi PWM group support	8

# 1 概述

## 1.1 编写目的

介绍 PWM 模块的详细设计方便相关人员进行 PWM 模块的代码设计开发。

## 1.2 使用范围

适用于 Linux-3.10, linux-4.4 和 Linux-4.9 内核, Linux-5.4 内核。

## 1.3 相关人员

PWM 驱动的开发人员/维护人员等

## 2 术语及概念

### 2.1 术语定义及缩略语

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台
频率	PWM 的频率决定了所模拟电平的平滑度（逼真度），人耳感知的频率范围为 20Hz-16Khz，注意 PWM 的频率不要落在这个区间
占空比	决定了一个周期内 PWM 信号高低的比例，进而决定了一个周期内的平均电压，也就是所模拟的电平的电压
极性	决定了是高占空比的信号输出电平高，还是低占空比信号输出电平高。假设一个信号的占空比为 100%，如果为正常极性，则输出电平最大，如果为翻转的极性，则输出电平为 0
开关	控制 PWM 信号是否输出
PWM 对	电机等硬件需要两路脉冲信号来控制其正常运转，一般两路极性相关，频率，占空比参数相同的 PWM 构成一个 PWM 对
PWM 死区控制时间	大功率电机，变频器等由大功率管，IGBT 等元件组成 H 桥或 3 相桥，每个桥的上半桥和下半桥是绝对不能导通的，在 PWM 信号驱动这些元件时，往往会由于没有延迟而造成未关断某路半桥，这样会造成功率元件的损坏，在 PWM 中加入死区时间的控制即是让上半桥关断后，自动插入一个事件，延迟后再打开下半桥

### 2.2 概念阐述

1. 脉冲宽度调制（PWM）是一种对模拟信号电平进行数字编码的方法。通过高分辨率计数器的使用，方波的占空比被调制用来对一个具体模拟信号的电平进行编码。
2. PWM 模块属于 PWM 子系统，会调用 PWM 子系统的相关接口（详情可以查看 PWM 子系统知识）

## 3 模块描述

### 3.1 模块功能

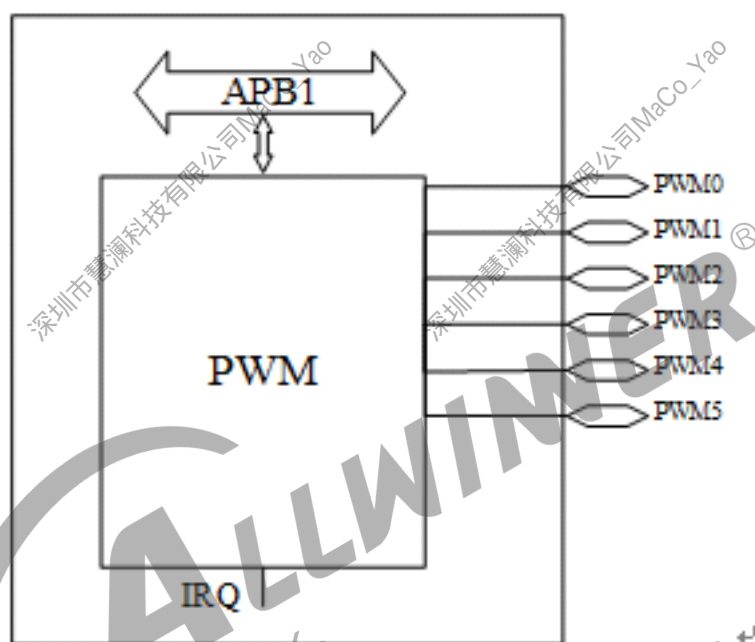


图 3-1: 模块功能

不同平台上拥有不同个数的 PWM 通道，其中两个为一个 PWM 对（平台通道数不相同，PWM 对也就不相同，具体细节可以查看对应方案的 spec）。其中 PWM 具有以下特点：

- 支持脉冲，周期和互补对输出
- 支出捕捉输入
- 带可编程死区发生器，死区时间可控
- 0-24M/100M 输出频率范围。0%-100% 占空比可调，最小分辨率 1/65536
- 支持 PWM 输出和捕捉输入产生中断

### 3.2 模块位置

PWM 模块属于硬件驱动层，直接与硬件通信

## 3.3 模块配置

### 3.3.1 linux-4.9

在 linux-4.9 中, 在命令行中进入内核根目录, 执行 `make ARCH=arm64 menuconfig` 进入配置主界面, 并按以下步骤操作:

1. 首先, 选择 Device Drivers 选项进入下一级配置, 如下图所示:



图 3-2: Device Drivers

2. 选择 Pulse-Width Modulation (PWM) Support 进入下一步配置, 如下图所示



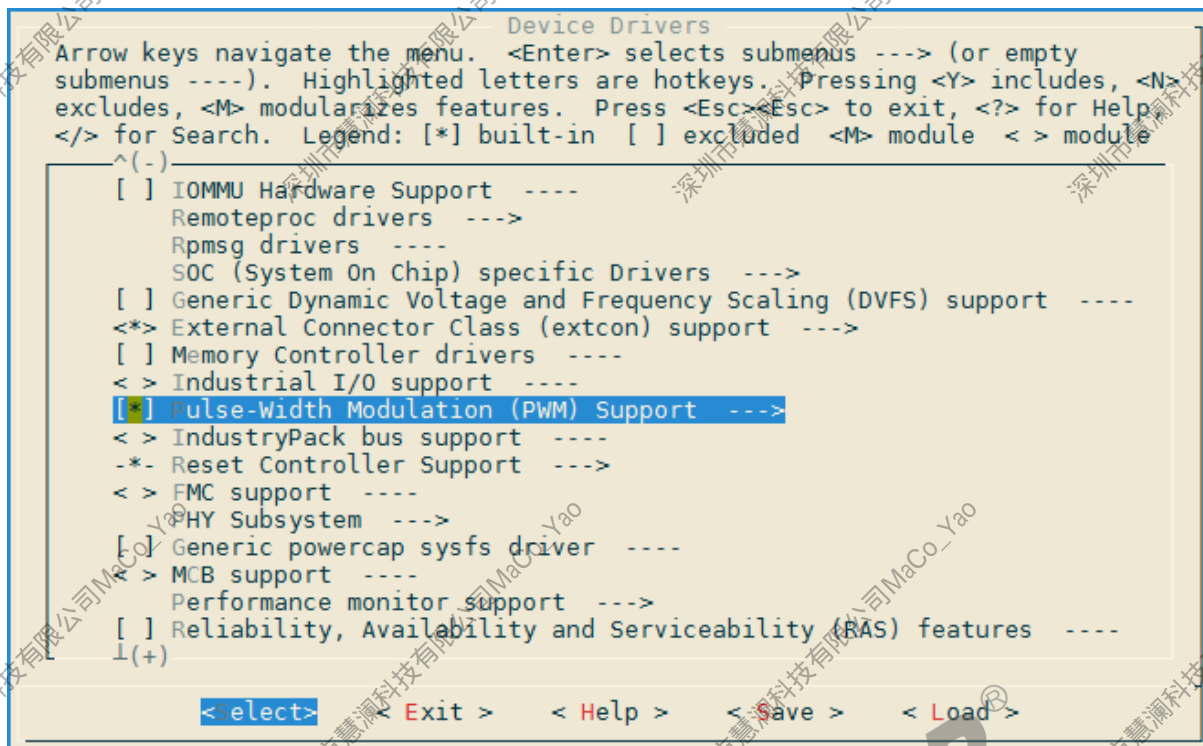


图 3-3: Pulse-Width Modulation (PWM) Support

3. 选择 SUNXI PWM SELECT 进入下一步配置，如下图所示：

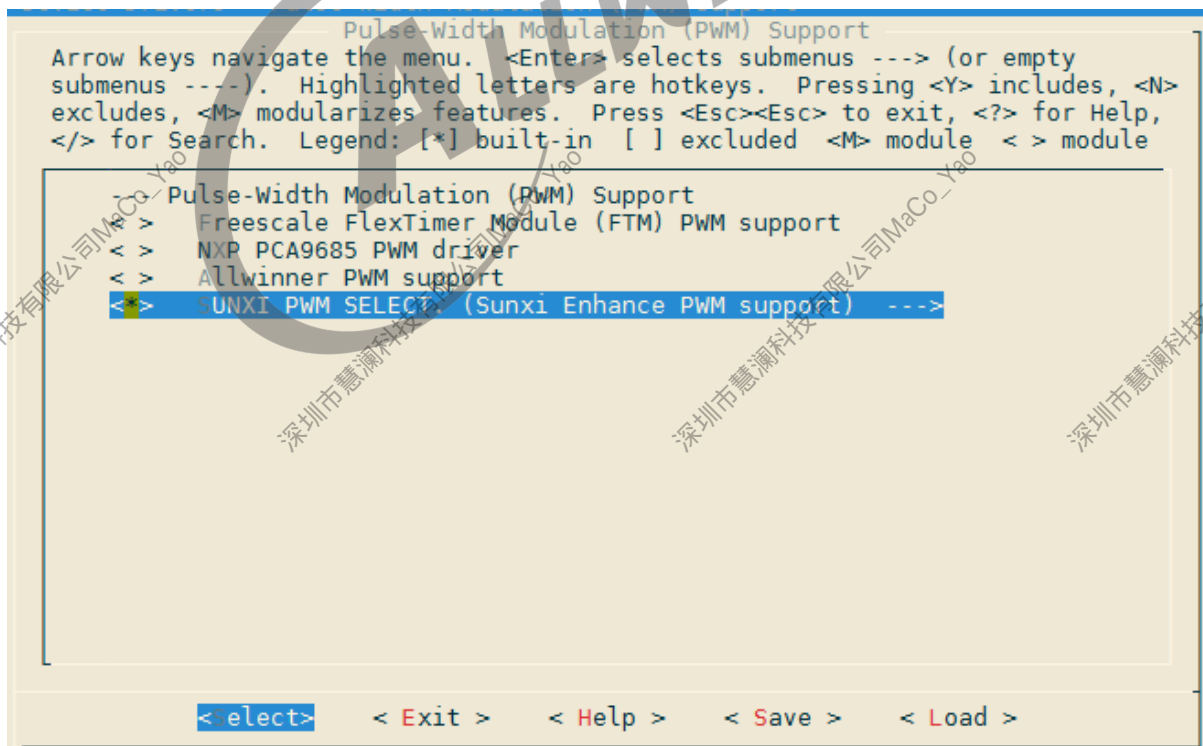


图 3-4: SUNXI PWM SELECT

4. 选择 Sunxi Enhance PWM support 配置

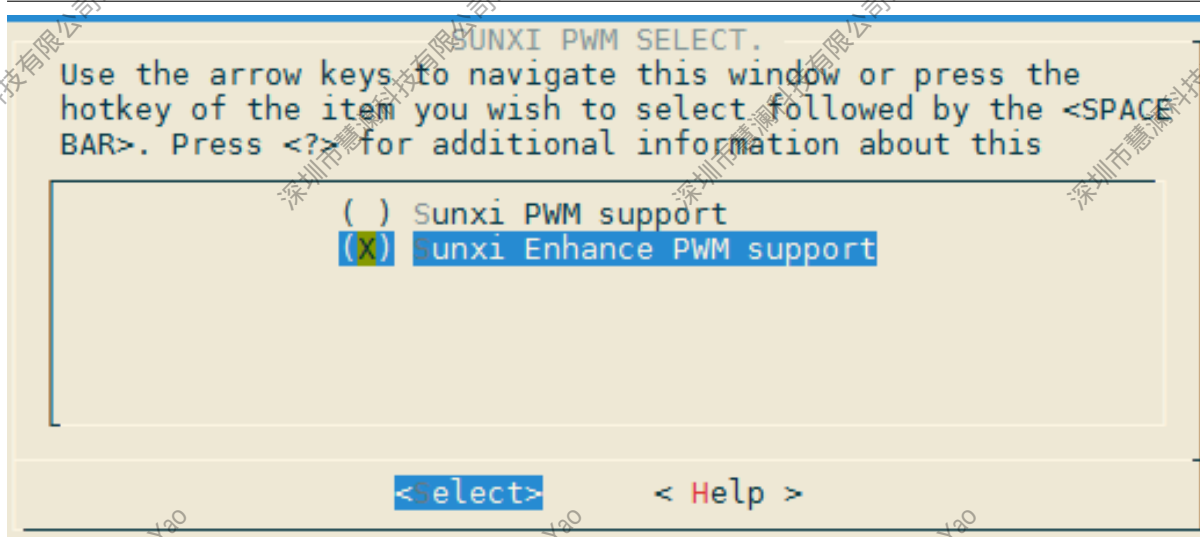


图 3-5: Sunxi Enhance PWM support

在 4.9 内核选择该配置，选择的是对应目录中的 pwm-sunxi-new.c 文件。也可以有以下配置；

在第 3 步中直接选择 Allwinner PWM support 选项，选择的是对应目录中的 pwm-sun4i.c 文件

在第 4 步中选择 Sunxi PWM Support 选项，选择的是对应目录中的 pwm-sunxi.c 文件

### 3.3.2 linux-5.4

linux5.4 平台中，在命令行中进入内核根目录，执行 ./build.sh menuconfig 进入配置主界面，并按以下步骤操作：

1. 首先，选择 Device Drivers 选项进入下一级配置，如下图所示：

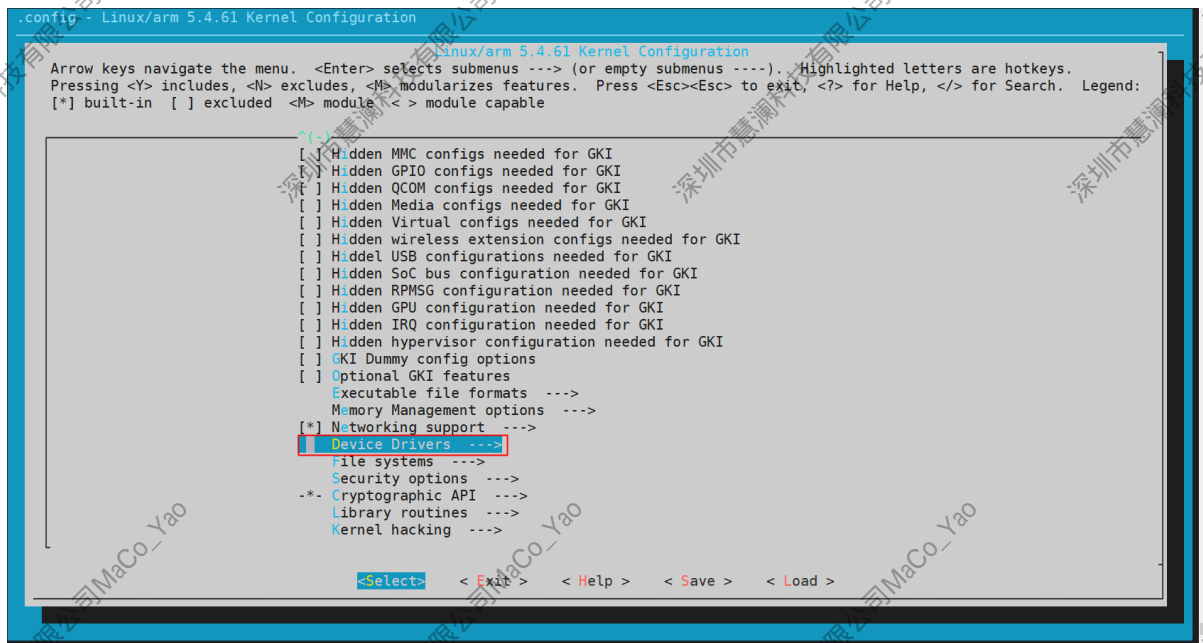


图 3-6: Device

2. 选择 Pulse-Width Modulation (PWM) Support 进入下一步配置，如下图所示

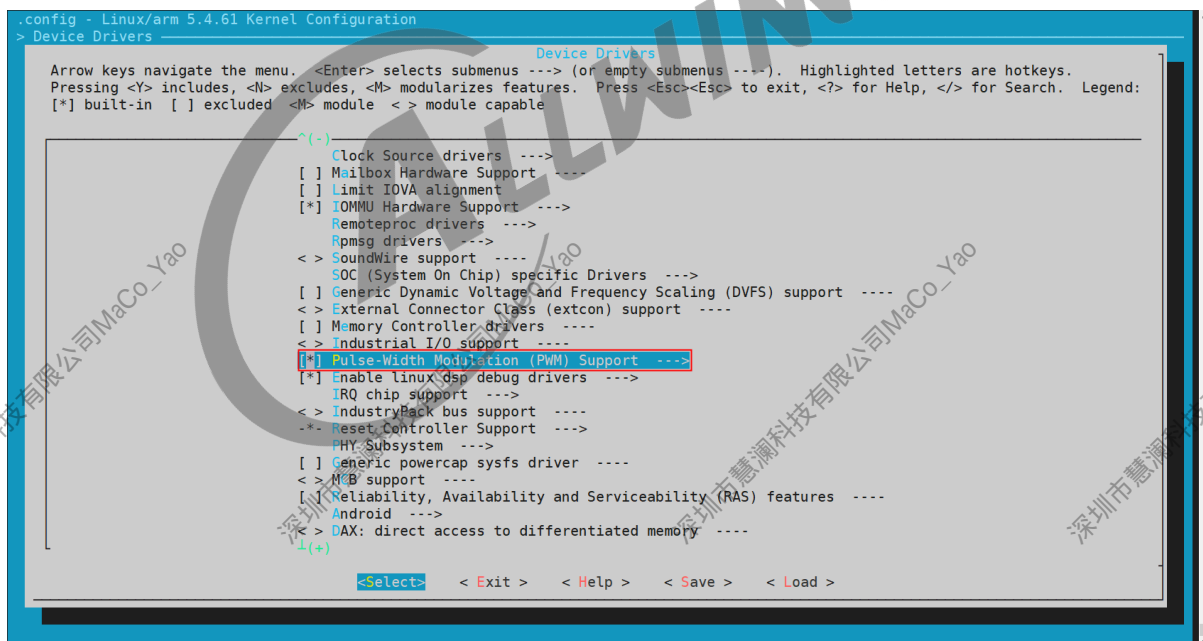


图 3-7: Pulse-Width Modulation (PWM) Support

3. 选择 SUNXI PWM SELECT 进入下一步配置，如下图所示：

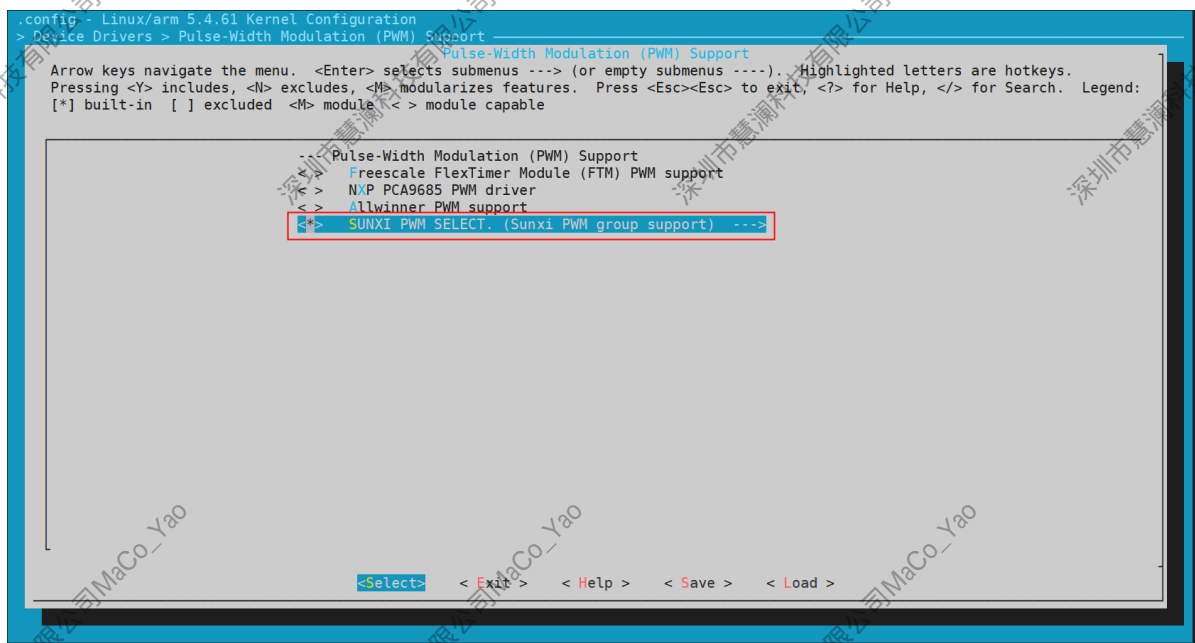


图 3-8: SUNXI PWM SELECT

#### 4. 选择 Sunxi PWM group support 配置

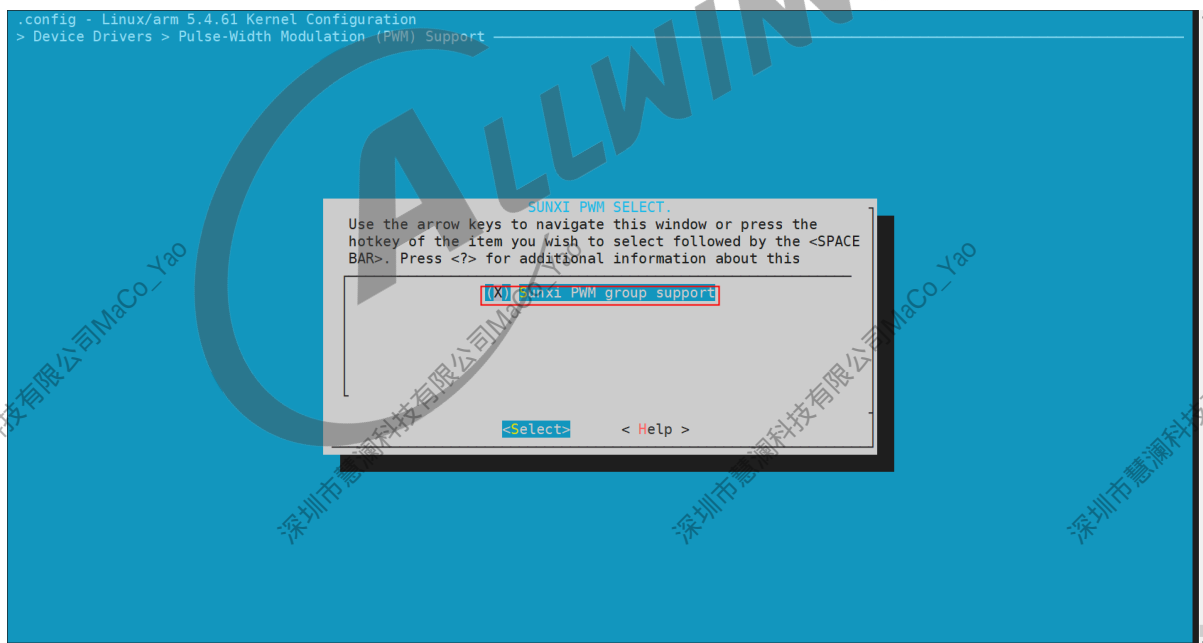


图 3-9: Sunxi PWM group support

## 3.4 设备树配置

### 3.4.1 linux-4.9

PWM 模块在设备树中的配置如下所示：

```
pwm: pwm@0300a000 {
    compatible = "allwinner,sunxi-pwm";
    reg = <0x0 0x0300a000 0x0 0x3c>;           //寄存器地址配置
    pwm-number = <1>;                          //pwm的个数
    pwm-base = <0x0>;                          //pwm的起始序号
    pwms = <&pwm0>, <&pwm1>;
};

s_pwm: s_pwm@0300a000 {
    compatible = "allwinner,sunxi-s_pwm";
    reg = <0x0 0x0300a000 0x0 0x3c>;
    pwm-number = <1>;
    pwm-base = <0x10>;
    pwms = <&spwm0>;
};
```

注意，如果在模块配置中选择了 Sunxi PWM support 选项 (具体参数可以查看相关源文件)，则需要配置以下设备树：

```
pwm0: pwm0@01c23400 {
    compatible = "allwinner,sunxi-pwm0";
    pinctrl-names = "active", "sleep";
    reg_base = <0x01c23400>;
    reg_peci_offset = <0x00>;
    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>;
};
```

PWM 模块在 sys\_config.fex 的配置如下所示：

```
[pwm0]
pwm_used      = 1
pwm_positive  = port:PB2<3><0><default><default>

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

### 3.4.2 linux-5.4

PWM 模块在设备树中的配置如下所示：

```
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>;
};

pwm1: pwm1@2000c11 {
    compatible = "allwinner,sunxi-pwm1";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c11 0x0 0x4>;
    reg_base = <0x02000c00>;
};

pwm2: pwm2@2000c12 {
    compatible = "allwinner,sunxi-pwm2";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c12 0x0 0x4>;
    reg_base = <0x02000c00>;
};

pwm3: pwm3@2000c13 {
    compatible = "allwinner,sunxi-pwm3";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c13 0x0 0x4>;
    reg_base = <0x02000c00>;
};

pwm4: pwm4@2000c14 {
    compatible = "allwinner,sunxi-pwm4";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c14 0x0 0x4>;
    reg_base = <0x02000c00>;
};
```



```

};

pwm5: pwm5@2000c15 {
    compatible = "allwinner,sunxi-pwm5";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c15 0x0 0x4>;
    reg_base = <0x02000c00>;
};

pwm6: pwm6@2000c16 {
    compatible = "allwinner,sunxi-pwm6";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c16 0x0 0x4>;
    reg_base = <0x02000c00>;
};

pwm7: pwm7@2000c17 {
    compatible = "allwinner,sunxi-pwm7";
    pinctrl-names = "active", "sleep";
    reg = <0x0 0x02000c17 0x0 0x4>;
    reg_base = <0x02000c00>;
};

```

在板级目录下的配置:

```

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;
};

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

pwm7_pin_b: pwm7@1 {
    pins = "PD22";
    function = "gpio_out";
};

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

&pwm7 {
    pinctrl-names = "active", "sleep";
};

```



```
pinctrl-0 = <&pwm7_pin_a>,  
pinctrl-1 = <&pwm7_pin_b>;  
status = "okay";  
};
```

具体通道配置按照需求进行配置。

## 3.5 源码结构

PWM 驱动的源代码位于内核的 drivers/pwm 目录下，具体的路径如下所示：

### 3.5.1 linux-4.9

```
drivers/pwm/  
├─ pwm-sunxi-new.c // Sunxi Enhance PWM support对应的PWM驱动  
├─ pwm-sunxi.c // Sunxi PWM support对应的PWM驱动  
├─ pwm-sun4i.c // Allwinner PWM support对应的PWM驱动  
├─ sysfs.c // PWM子系统的文件系统相关文件  
└─ core.c // PWM子系统的核心文件
```

### 3.5.2 linux-5.4

```
drivers/pwm/  
├─ pwm-sunxi-group.c // Sunxi GROUP PWM support对应的PWM驱动  
├─ sysfs.c // PWM子系统的文件系统相关文件  
└─ core.c // PWM子系统的核心文件
```

## 3.6 调试接口

可以直接在 linux 内核中调试 pwm 模块，具体如下：

进入/sys/class/pwm 目录，该目录是 linux 内核为 pwm 子系统提供的类目录，遍历该目录：

```
/sys/class/pwm # ls  
pwmchip0
```

可以看到，上述 pwmchip0 就是我们注册的 pwm 控制器，进入该目录，然后遍历该目录：

```
/sys/class/pwm # cd pwmchip0/  
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # ls  
device      export      npwm        subsystem  uevent      unexport
```

其中 npwm 文件储存了该 pwm 控制器的 pwm 个数，而 export 和 unexport 是导出和删除某个 pwm 设备的文件，下面演示导出 pwm1。

```
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # cat npwm
2
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # echo 1 > export
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # ls
device      export      npwm        pwm1        subsystem  uevent      unexport
```

可以看到目录中多出 pwm1 目录，进入该目录，遍历：

```
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # cd pwm1/
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0/pwm1 # ls
capture      duty_cycle  enable      period      polarity    uevent
```

该目录中，enable 是使能 pwm，duty\_cycle 是占空比，period 是周期，polarity 是极性，可以配置相关的 pwm 并且使能：

```
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0/pwm1 # echo 1000000000 > period
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0/pwm1 # echo 500000000 > duty_cycle
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0/pwm1 # echo normal > polarity
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0/pwm1 # echo 1 > enable
```

如果相关引脚接上了示波器等，可以看到波形。最后返回上层目录，删除该 pwm 设备：

```
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0/pwm1 # cd ..
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # ls
device      export      npwm        pwm1        subsystem  uevent      unexport
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # echo 1 > unexport
/sys/devices/platform/soc/1c23400.pwm/pwm/pwmchip0 # ls
device      export      npwm        subsystem  uevent      unexport
```

## 著作权声明

版权所有 © 2022 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



（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。