Android 10

sys_config 配置说明文档

1.0

2020.2.26

文档履历

版本号	日期	制/修订人	内容描述
1.0	2020.2.26		正式版本

目录

1. 前言

1.1 编写目的

本文档目的是介绍 sys_config.fex 各个节点配置的意义，让用户明确掌握 sys_config.fex 配置和使用方法。

1.2 适用范围

适用于 A133/A100 芯片相关平台。

1.3 相关人员

用户、板级配置维护相关人员。

2. 节点配置说明

2.1 系统 (SYSTEM)

2.1.1 [product]

配置项	配置项含义
version	sdk 版本号
machine	sdk 代号

配置举例：

```
version = "100"
machine = "evb"
```

2.1.2 [platform]

配置项	配置项含义
eraseflag	量产时是否擦除。0：不擦，1：擦除（仅对量产有效，OTA 无效）
next_work	USB 量产完成后状态。1: 不做任何动作 2：重启 3: 关机 4：量产

配置项	配置项含义
debug_mode	uboot 打印等级。 0=LOG_LEVEL_NONE; 1=LOG_LEVEL_ERROR; 2=LOG_LEVEL_WARNING; 3=LOG_LEVEL_NOTICE; 4=LOG_LEVEL_INFO

配置举例：

```
eraseflag   = 1
next_work   = 3
debug_mode  = 1
```

2.1.3 [target]

配置项	配置项含义
boot_clock	启动频率，单位：MHz
storage_type	启动介质选择 0：nand, 1：card0,2：card2,-1（defualt）:auto scan
burn_key	启动时是否需要烧 key 0：不烧 1：烧
dragonboard_test	是否编译支持卡启动的 dragonboard 固件。1：是 0：否

配置举例：

```
boot_clock     = 1008
storage_type   = -1
burn_key       = 1
dragonboard_test= 0
```

2.1.4 [card_boot]

配置项	配置项含义
logical_start	启动卡逻辑起始扇区
sprite_gpio0	卡量产，一键 recovery led 指示灯 GPIO 配置
next_work	卡量产完成后状态：1: 不做任何动作 2：重启 3: 关机 4：量产

配置举例：

```
logical_start      = 40960
sprite_gpio0   = port:PA15<1><default><default><default>
next_work          = 3
```

2.1.5 [card0_boot_para]

配置项	配置项含义
card_ctrl	卡量产相关的控制器选择 0
card_high_speed	速度模式 0 为低速，1 为高速
card_line	4：4 线卡，8：8 线卡
sdc_d1	sdc 卡数据 1 线信号的 GPIO 配置
sdc_d0	sdc 卡数据 0 线信号的 GPIO 配置
sdc_clk	sdc 卡时钟信号的 GPIO 配置
sdc_cmd	sdc 命令信号的 GPIO 配置
sdc_d3	sdc 卡数据 3 线信号的 GPIO 配置
sdc_d2	sdc 卡数据 2 线信号的 GPIO 配置

配置举例：

```
card_ctrl      = 0
card_high_speed = 1
card_line      = 4
sdc_d1         = port:PF0<2><1><3><default>
sdc_d0         = port:PF1<2><1><3><default>
sdc_clk        = port:PF2<2><1><3><default>
sdc_cmd        = port:PF3<2><1><3><default>
sdc_d3         = port:PF4<2><1><3><default>
sdc_d2         = port:PF5<2><1><3><default>
```

2.1.6 [card2_boot_para]

配置项	配置项含义
card_ctrl	卡启动控制器选择 2
card_high_speed	速度模式 0 为低速，1 为高速
card_line	4：4 线卡，8：8 线卡
sdc_ds	ds 信号的 GPIO 配置
sdc_d1	sdc 卡数据 1 线信号的 GPIO 配置
sdc_d0	sdc 卡数据 0 线信号的 GPIO 配置
sdc_clk	sdc 卡时钟信号的 GPIO 配置
sdc_cmd	sdc 命令信号的 GPIO 配置
sdc_d3	sdc 卡数据 3 线信号的 GPIO 配置
sdc_d2	sdc 卡数据 2 线信号的 GPIO 配置
sdc_d4	sdc 卡数据 4 线信号的 GPIO 配置
sdc_d5	sdc 卡数据 5 线信号的 GPIO 配置
sdc_d6	sdc 卡数据 6 线信号的 GPIO 配置
sdc_d7	sdc 卡数据 7 线信号的 GPIO 配置
sdc_emmc_rst	emmc_rst 信号的 GPIO 配置
sdc_ex_dly_used	ex_dly_used 信号
sdc_io_1v8	sdc_io_1v8 高速 emmc 模式配置

配置举例：

```
card_ctrl      = 2
card_high_speed = 1
card_line      = 8
sdc_clk        = port:PC5<3><1><3><default>
sdc_cmd        = port:PC6<3><1><3><default>
sdc_d0         = port:PC10<3><1><3><default>
sdc_d1         = port:PC13<3><1><3><default>
sdc_d2         = port:PC15<3><1><3><default>
sdc_d3         = port:PC8<3><1><3><default>
sdc_d4         = port:PC9<3><1><3><default>
sdc_d5         = port:PC11<3><1><3><default>
sdc_d6         = port:PC14<3><1><3><default>
sdc_d7         = port:PC16<3><1><3><default>
sdc_emmc_rst   = port:PC1<3><1><3><default>
sdc_ds         = port:PC0<3><2><3><default>
sdc_tm4_hs200_max_freq = 150
sdc_tm4_hs400_max_freq = 100
sdc_ex_dly_used = 2
sdc_io_1v8  = 1
sdc_tm4_win_th = 8
;sdc_dis_host_caps = 0x100
;sdc_erase  = 2
;sdc_boot0_sup_1v8 = 1
```

2.1.7 [gpio_bias]

配置项	配置项含义
pc_bias	emmc 电压配置，高速 emmc 才使用

配置举例：

```
pc_bias    = 1800
```

2.1.8 [uart_para]

配置项	配置项含义
uart_debug_port	Boot 串口控制器编号
uart_debug_tx	Boot 串口发送的 GPIO 配置
uart_debug_rx	Boot 串口接收的 GPIO 配置

配置举例：

```
uart_debug_port = 0
uart_debug_tx   = port:PB09<2><1><default><default>
uart_debug_rx   = port:PB10<2><1><default><default>>
```

2.1.9 [jtag_para]

配置项	配置项含义
jtag_enable	JTAG 使能
jtag_ms	测试模式选择输入 (TMS) 的 GPIO 配置
jtag_ck	测试时钟输入 (CLK) 的 GPIO 配置
jtag_do	测试数据输出 (TDO) 的 GPIO 配置
jtag_di	测试数据输出 (TDI) 的 GPIO 配置

配置举例：

```
jtag_enable       = 1
jtag_ms       = port:PH9<3><default><default><default>
jtag_ck       = port:PH10<3><default><default><default>
jtag_do       = port:PH11<3><default><default><default>
jtag_di       = port:PH12<3><default><default><default>
```

2.2 DRAM 配置

2.2.1 [dram_para]

配置项	配置项含义
dram_clk	DRAM 的时钟频率，单位为 MHz
dram_type	DRAM 类型：8 为 LPDDR4 , 由源厂调节，请勿修改
dram_zq	DRAM 控制器内部参数，由源厂调节，请勿修改
dram_odt_en	ODT 是否需要使能，为了省电，一般设置为 0, 由源厂调节，请勿修改
dram_mr0	DRAM CAS 值，可为 6,7,8,9；由源厂调节，请勿修改
dram_xxx	由源厂调节，请勿修改

配置举例：

```
dram_clk      = 672
dram_type     = 7
dram_dx_odt   = 0x06060606
dram_dx_dri   = 0x0c0c0c0c
dram_ca_dri   = 0x1919
dram_para0    = 0x16171411
dram_para1    = 0x30eb
dram_para2    = 0x0000
dram_mr0      = 0x0
dram_mr1      = 0xc3
dram_mr2      = 0x6
dram_mr3      = 0x2
dram_mr4      = 0x0
dram_mr5      = 0x0
dram_mr6      = 0x0
dram_mr11     = 0x0
dram_mr12     = 0x0
dram_mr13     = 0x0
dram_mr14     = 0x0
dram_mr16     = 0x0
dram_mr17     = 0x0
dram_mr22     = 0x0
dram_tpr0     = 0x0
dram_tpr1     = 0x0
dram_tpr2     = 0x0
dram_tpr3     = 0x0
```

```
dram_tpr6     = 0x2fb48080
dram_tpr10    = 0x002f876b
dram_tpr11    = 0x10120c05
dram_tpr12    = 0x12121111
dram_tpr13    = 0x61
dram_tpr14    = 0x211e1e22
```

2.3 UART

2.3.1 [uart0]

配置项	配置项含义
uart0_used	UART 使用控制：1 使用，0 不用
uart0_port	UART 端口号
uart0_type	2：2 线模式;4：4 线模式;8：8 线模式。
uart0_tx	UART TX 的 GPIO 配置
uart0_rx	UART RX 的 GPIO 配置

配置举例：

```
uart0_used      = 1
uart0_port      = 0
uart0_type      = 2
uart0_tx        = port:PB09<2><1><default><default>
uart0_rx        = port:PB10<2><1><default><default>
```

2.4 NAND FLASH

2.4.1 [nand0_para]

配置项	配置项含义
nand_support_2ch	nand0 是否使能双通道
nand0_used	nand0 模块使能标志
nand0_we	nand0 写时钟信号的 GPIO 配置
nand0_ale	nand0 地址使能信号的 GPIO 配置
nand0_cle	nand0 命令使能信号的 GPIO 配置
nand0_ce1	nand0 片选 1 信号的 GPIO 配置
nand0_ce0	nand0 片选 0 信号的 GPIO 配置
nand0_nre	nand0 读时钟信号的 GPIO 配置
nand0_rb0	nand0 Read/Busy 1 信号的 GPIO 配置
nand0_rb1	nand0 Read/Busy 0 信号的 GPIO 配置
nand0_d0	nand0 数据总线信号的 GPIO 配置
nand0_d1	/
nand0_d2	/
nand0_d3	/
nand0_d4	/
nand0_d5	/
nand0_d6	/
nand0_d7	/
nand0_ndqs	nand0 ddr 时钟信号的 GPIO 配置
nand0_ce2	nand0 片选 2 信号的 GPIO 配置
nand0_ce3	nand0 片选 3 信号的 GPIO 配置

配置举例：

```
nand0_support_2ch    = 0

nand0_used        = 0
nand0_we          = port:PC00<2><0><1><default>
nand0_ale         = port:PC01<2><0><1><default>
nand0_cle         = port:PC02<2><0><1><default>
nand0_ce0         = port:PC03<2><1><1><default>
nand0_nre         = port:PC04<2><0><1><default>
nand0_rb0         = port:PC05<2><1><1><default>
nand0_d0          = port:PC06<2><0><1><default>
nand0_d1          = port:PC07<2><0><1><default>
```

```
nand0_d2          = port:PC08<2><0><1><default>
nand0_d3          = port:PC09<2><0><1><default>
nand0_d4          = port:PC10<2><0><1><default>
nand0_d5          = port:PC11<2><0><1><default>
nand0_d6          = port:PC12<2><0><1><default>
nand0_d7          = port:PC13<2><0><1><default>
nand0_ndqs          = port:PC14<2><0><1><default>
nand0_ce1         = port:PC15<2><1><1><default>
nand0_rb1        = port:PC16<2><1><1><default>

nand0_regulator1       = "vcc-nand"
nand0_regulator2       = "none"
nand0_cache_level = 0x55aaaa55
nand0_flush_cache_num = 0x55aaaa55
nand0_capacity_level = 0x55aaaa55
nand0_id_number_ctl = 0x55aaaa55
nand0_print_level = 0x55aaaa55
nand0_p0 = 0x55aaaa55
nand0_p1 = 0x55aaaa55
nand0_p2 = 0x55aaaa55
nand0_p3 = 0x55aaaa55
```

2.5 显示

2.5.1 [disp]

配置项	配置项含义
disp_init_enable	是否进行显示的初始化设置
disp_mode	显示模式：0:screen0 1：screen1
screen0_output_type	屏 0 输出类型 (0:none; 1:lcd; 2:tv; 3:hdmi; 4:vga)
screen0_output_mode	0:480i 1:576i 2:480p 3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p249:1080p50 10:1080p60 11:pal 14:ntsc)
screen1_output_type	屏 1 输出类型 (0:none; 1:lcd; 2:tv; 3:hdmi; 4:vga)
screen1_output_mode	0:480i 1:576i 2:480p 3:576p 4:720p50 5:720p60 6:1080i50 7:1080i60 8:1080p24 9:1080p50 10:1080p60 11:pal 14:ntsc)
fb0_format	fb0 的格式 (0:ARGB 1:ABGR 2:RGBA 3:BGRA)
fb0_width	fb0 的宽度, 为 0 时将按照输出设备的分辨率
fb0_height	fb0 的高度，为 0 时将按照输出设备的分辨率

配置项	配置项含义
fb1_format	fb1 的格式 (0:ARGB 1:ABGR 2:RGBA 3:BGRA)
fb1_width	Fb1 的宽度, 为 0 时将按照输出设备的分辨率
fb1_height	Fb1 的高度，为 0 时将按照输出设备的分辨率
dev0_output_type	boot 阶段显示配置：screen0 对应的输出类型（4：hdmi，2：cvbs）
dev0_output_mode	boot 阶段显示配置：screen0 对应的输出模式
dev0_screen_id	boot 阶段显示配置：screen0 对应的 DE 设备号
dev0_do_hpd	boot 阶段显示配置：是否支持热插拔检测
dev1_output_type	boot 阶段显示配置：screen1 对应的输出类型（4：hdmi，2：cvbs）
dev1_output_mode	boot 阶段显示配置：screen1 对应的输出模式
dev1_screen_id	boot 阶段显示配置：screen1 对应的 DE 设备号
dev1_do_hpd	boot 阶段显示配置：是否支持热插拔检测
dev2_output_type	保留
def_output_dev	保留
hdmi_mode_check	boot 阶段显示配置：是否支持 hdmi 输出模式检查

配置举例：

```
请看当前目录下的board.dts
disp: disp@06000000 {
    disp_init_enable       = <1>;
    disp_mode              = <0>;

    screen0_output_type    = <1>;
    screen0_output_mode    = <4>;

    screen1_output_type    = <1>;
    screen1_output_mode    = <4>;

    screen1_output_format  = <0>;
    screen1_output_bits    = <0>;
    screen1_output_eotf    = <4>;
    screen1_output_cs      = <257>;
    screen1_output_dvi_hdmi = <2>;
    screen1_output_range   = <2>;
    screen1_output_scan    = <0>;
    screen1_output_aspect_ratio = <8>;

    dev0_output_type       = <1>;
    dev0_output_mode       = <4>;
```

```
    dev0_screen_id          = <0>;
    dev0_do_hpd             = <0>;

    dev1_output_type        = <4>;
    dev1_output_mode        = <10>;
    dev1_screen_id          = <1>;
    dev1_do_hpd             = <1>;

    def_output_dev          = <0>;
    hdmi_mode_check         = <1>;

    fb0_format              = <0>;
    fb0_width               = <800>;
    fb0_height              = <1280>;

    fb1_format              = <0>;
    fb1_width               = <0>;
    fb1_height              = <0>;
    chn_cfg_mode            = <1>;

    disp_para_zone          = <1>;
    /*VCC-LCD*/
    dc1sw-supply = <&reg_dc1sw>;
    /*VCC-DSI*/
    eldo3-supply = <&reg_eldo3>;
    /*VCC-PD*/
    dcdc1-supply = <&reg_dcdc1>;
};
```

2.5.2 [lcd0]

配置项	配置项含义
lcd_used	是否使用 lcd_used。1：使用；0：不使用
lcd_driver_name	lcd 驱动名字
lcd_backlight	lcd 默认 backlight
lcd_if	lcd 接口 (A50 平台支持以下两种接口类型 ------0,:RGB，1:I8080，3:LVDS，4:dsi)
lcd_x	lcd 分辨率 x
lcd_y	lcd 分辨率 y
lcd_width	lcd 屏宽度，单位英寸
lcd_height	lcd 屏高度，单位英寸
lcd_dclk_freq	像素时钟频率，lcd_dclk_freq=ht*v*tfps

配置项	配置项含义
lcd_pwm_used	pwm 是否使用（1：enable；0：disable）
lcd_pwm_ch	pwm 通道（通道 0/1）
lcd_pwm_freq	pwm 频率
lcd_pwm_pol	pwm 属性, 0:positive; 1:negative
lcd_pwm_max_limit	lcd backlight pwm max limit(<=255)
lcd_hbp	lcd 行后沿时间
lcd_ht	lcd 行时间
lcd_hspw	lcd 行同步脉宽
lcd_vbp	lcd 场后沿时间
lcd_vt	lcd 场时间
lcd_vspw	lcd 场同步脉宽
lcd_lvds_if	lcd_lvds_if
lcd_lvds_colordepth	lcd_lvds_colordepth
lcd_lvds_mode	内核阶段 hdmi 电源配置
lcd_frm	lcd 格式，0:disable; 1:enable rgb666 dither; 2:enable rgb656 dithe
lcd_hv_clk_phase	lcd hv 时钟
lcd_hv_sync_polarity	lcd io 属性，0:not invert; 1:invert
lcd_gamma_en	lcdgamma 校正使能
lcd_bright_curve_en	lcd 亮度曲线校正使能
lcd_cmap_en	lcd 调色板函数使能
deu_mode	deu_mode
lcdgamma4iep	lcdgamma4iep
smart_color	smart_color
lcdd0	lcd0 的 GPIO 配置
lcdd1	lcd1 的 GPIO 配置
lcdd2	lcd2 的 GPIO 配置
lcdd3	lcd3 的 GPIO 配置
lcdd4	lcd4 的 GPIO 配置
lcdd5	lcd5 的 GPIO 配置
lcdd6	lcd6 的 GPIO 配置
lcdd7	lcd7 的 GPIO 配置
lcdd8	lcd8 的 GPIO 配置
lcdd9	lcd9 的 GPIO 配置

配置举例：

```
请看当前目录下的board.dts
lcd0: lcd0@01c0c000 {
    lcd_used            = <1>;

    lcd_driver_name     = "k101im2qa04";
    lcd_backlight       = <50>;
    lcd_if              = <4>;

    lcd_x               = <800>;
    lcd_y               = <1280>;
    lcd_width           = <135>;
    lcd_height          = <216>;
    lcd_dclk_freq       = <68>;

    lcd_pwm_used        = <1>;
    lcd_pwm_ch          = <0>;
    lcd_pwm_freq        = <50000>;
    lcd_pwm_pol         = <1>;
    lcd_pwm_max_limit   = <255>;


    lcd_hbp             = <36>;
    lcd_ht              = <854>;
    lcd_hspw            = <18>;
    lcd_vbp             = <12>;
    lcd_vt              = <1320>;
    lcd_vspw            = <4>;

    lcd_frm             = <0>;
    lcd_gamma_en        = <0>;
    lcd_bright_curve_en = <0>;
    lcd_cmap_en         = <0>;

    deu_mode            = <0>;
    lcdgamma4iep        = <22>;
    smart_color         = <90>;

    lcd_dsi_if          = <0>;
    lcd_dsi_lane        = <4>;
    lcd_dsi_format      = <0>;
    lcd_dsi_te          = <0>;
    lcd_dsi_eotp        = <0>;

    lcd_pin_power = "dcdc1";
    lcd_pin_power1 = "eldo3";

    lcd_power = "dc1sw";
    lcd_bl_en = <&pio PB 8 1 0 3 1>;
    /*lcd_gpio_1 = <&pio PD 23 1 0 3 1>;*/
```

```
    lcd_gpio_0 = <&pio PD 22 1 0 3 1>;
    pinctrl-0 = <&dsi4lane_pins_a>;
    pinctrl-1 = <&dsi4lane_pins_b>;

};
```

2.6 SD/MMC

2.6.1 [sdc0]

配置项	配置项含义
sdc0_used	SDC 使用控制：1 使用，0 不用
bus-width	位宽：1-1bit，4-4bit
cd-gpios	SDC 卡检测信号的 GPIO 配置
sunxi-power-save-mode	SDC CLK 信号无数据传输时暂停
vmmc	SDC 供电电源配置
vqmmc	SDC IO 供电电源配置
vdmmc	SDC 卡检测信号上拉电阻的电源配置

举例说明：

```
请看当前目录下的board.dts
sdc0: sdmmc@04020000 {
    bus-width = <4>;
    cd-gpios = <&pio PF 6 6 1 3 0xffffffff>;
    /*non-removable;*/
    /*broken-cd;*/
    /*cd-inverted*/
    /*data3-detect;*/
    cd-used-24M;
    cd-gpios = <&pio PF 6 6 1 3 0xffffffff>;
    cap-sd-highspeed;
    sd-uhs-sdr50;
    sd-uhs-ddr50;
    sd-uhs-sdr104;
```

```
    no-sdio;
    no-mmc;
    sunxi-power-save-mode;
    /*sunxi-dis-signal-vol-sw;*/
    max-frequency = <150000000>;
    ctl-spec-caps = <0x8>;
    vmmc-supply = <&reg_dcdc1>;
    vqmmc33sw-supply = <&reg_dcdc1>;
    vdmmc33sw-supply = <&reg_dcdc1>;
    vqmmc18sw-supply = <&reg_eldo1>;
    vdmmc18sw-supply = <&reg_eldo1>;
    status = "okay";
};
```

2.6.2 [sdc1]

配置项	配置项含义
sdc1_used	SDC 使用控制：1 使用，0 不用
bus-width	位宽：1-1bit，4-4bit
sunxi-power-save-mode	SDC CLK 信号无数据传输时暂停
sd-uhs-sdr50	支持 SDR50 速度模式
sd-uhs-ddr50	支持 DDR50 速度模式置
sd-uhs-sdr104	支持 SDR104 速度模式置
cap-sdio-irq	目前只用于 SDIO WIFI 驱动，表示控制器支持 SDIO 中断。
keep-power-in-suspend	目前只用于 SDIO WIFI 驱动，表示休眠时器件供电保持不变
ignore-pm-notify	目前只用于 SDIO WIFI 驱动，表示休眠唤醒时忽略内核的 pm notify
max-frequency	最高接口配置频率配置

举例说明：

```
请看当前目录下的board.dts
sdc1: sdmmc@04021000 {
    bus-width = <4>;
    no-mmc;
    no-sd;
    cap-sd-highspeed;
```

```
    /*sd-uhs-sdr12*/
    /*sd-uhs-sdr25*/
    /*sd-uhs-sdr50;*/
    /*sd-uhs-ddr50;*/
    /*sd-uhs-sdr104*/
    /*sunxi-power-save-mode;*/
    /*sunxi-dis-signal-vol-sw;*/
    cap-sdio-irq;
    keep-power-in-suspend;
    ignore-pm-notify;
    max-frequency = <50000000>;
    ctl-spec-caps = <0x8>;
    status = "okay";
};
```

2.6.3 [sdc2]

配置项	配置项含义
sdc2_used	SDC 使用控制：1 使用，0 不用
non-removable	SDC 连接 Device 具备不可移除属性
bus-width	SDC DATA1 的 GPIO 配置
sdc2_emmc_rst	SDC eMMC Hardware Reset 的 GPIO 配置
cd-gpios	SDC 卡检测信号的 GPIO 配置
sunxi-power-save-mode	SDC CLK 信号无数据传输时暂停
sunxi-dis-signal-vol-sw	MMC 驱动支持开关 IO 电压但不修改 IO 电压值
vmmc	SDC 供电电源配置
vqmmc	SDC IO 供电电源配置
vdmmc	SDC 卡检测信号上拉电阻的电源配置

配置举例：

```
请看当前目录下的board.dts
sdc2: sdmmc@04022000 {
    non-removable;
    bus-width = <8>;
    mmc-ddr-1_8v;
```

```
    mmc-hs200-1_8v;
    mmc-hs400-1_8v;
    no-sdio;
    no-sd;
    ctl-spec-caps = <0x8>;
    cap-mmc-highspeed;
    sunxi-power-save-mode;
    sunxi-dis-signal-vol-sw;
    max-frequency = <100000000>;
    vmmc-supply = <&reg_dcdc1>;
    /*emmc io vol 3.3v*/
    /*vqmmc-supply = <&reg_aldo1>;*/
    /*emmc io vol 1.8v*/
    vqmmc-supply = <&reg_eldo1>;
    status = "disabled";
};
```

2.7 USB 控制器标志

2.7.1 [usbc0]

配置项	配置项含义
usb0_used	USB 使能标志 (xx=1 or 0)。置 1，表示系统中 USB 模块可用，置 0，则表示系统 USB 禁用。此标志只对具体的 USB 控制器模块有效。
usb_port_type	USB 端口的使用情况。(xx=0/1/2) 0：device only 1：host only 2：OTG
usb_detect_type	USB 端口的检查方式。0：无检查方式 1：vbus/id 检查
usb_id_gpio	USB ID pin 脚配置
usb_det_vbus_gpio	USB DET_VBUS pin 脚配置
usb_drv_vbus_gpio	USB DRY_VBUS pin 脚配置
usb_host_init_state	host only 模式下，Host 端口初始化状态。0：初始化后 USB 不工作 1：初始化后 USB 工作
usb_regulator_io	usb 供电的 regulator GPIO
usb_wakeup_suspend	支持 usb 唤醒功能 0：关闭 usb 唤醒功能 1：当进入 normal standby 时候，支持 usb 唤醒（例如鼠标等外设）
USB device	
usb_luns	使用 mass storage 功能时的盘符数量

配置项	配置项含义
usb_serial_unique	usb device 的序列号是否唯一。1：唯一，使用 chip id; 0：相同: 由 usb_serial_number 指定
usb_serial_number	usb device 的序列号量
rndis_wceis	Wireless RNDIS 使能标志。1：使能;0：禁止

配置举例：

```
请看当前目录下的board.dts
usbc0:usbc0@0 {
    device_type = "usbc0";
    usb_port_type = <0x2>;
    usb_detect_type = <0x1>;
    usb_id_gpio = <&pio PH 8 0 0 0xffffffff 0xffffffff>;
    usb_det_vbus_gpio   = "axp_ctrl";
    usb_regulator_io = "nocare";
    usb_wakeup_suspend = <0>;
    usb_serial_unique = <0>;
    usb_serial_number = "20080411";
    rndis_wceis = <1>;
    status = "okay";
};
```

2.7.2 [usbc1]

配置项	配置项含义
usb1_used	USB 使能标志 (xx=1 or 0)。置 1，表示系统中 USB 模块可用，置 0，则表示系统 USB 禁用。此标志只对具体的 USB 控制器模块有效。
usb_drv_vbus_gpio	USB DRY_VBUS pin 脚配置。具体请参考 gpio 配置说明
usb_host_init_state	host only 模式下，Host 端口初始化状态。0：初始化后 USB 不工作 1：初始化后 USB 工作
usb_regulator_io	给 usb 供电的 regulator GPIO
usb_wakeup_suspend	支持 usb 唤醒功能 0：关闭 usb 唤醒功能 1：当进入 normal standby 时候，支持 usb 唤醒（例如鼠标等外设）

配置举例：

```
请看当前目录下的board.dts
usbc1:usbc1@0 {
    device_type = "usbc1";
    usb_regulator_io = "nocare";
    usb_wakeup_suspend = <0>;
    status = "disable";
};
```

2.8 WIFI

2.8.1 [wlan]

配置项	配置项含义
compatible	固定值，请勿修改
clocks	32K/24M DCXO 时钟配置表示，如使用外部时钟 32K/外部 24M，则无需配置
pinctrl-0/pinctrl-names	使用 DCXO，且 DCXO 跟 GPIO 复用时需要配置，否则无需配置
wlan_busnum	表示 Wi-Fi 所使用的 SDIO 控制器号
wlan_power	表示给 Wi-Fi 模组供电的 regulator 名称
wlan_io_regulator	表示给 Wi-Fi 模组的 GPIO 供电的 regulator 名称
wlan_regon	Wi-Fi 模组 power on 控制引脚
wlan_hostwake	表示 Wi-Fi 唤醒主控的 GPIO
chip_en	表示 Wi-Fi 模组使能引脚，硬件未使用时则无需配置
power_en	表示模块外部的电源开关控制引脚，硬件未使用时则无需配置
status	表示是否使用该模块

配置举例：

```
请看当前目录下的board.dts
wlan: wlan@0 {
    compatible    = "allwinner,sunxi-wlan";
```

```
    clocks      = <&clk_losc_out>, <&clk_dcxo_out>;
    pinctrl-0;
    pinctrl-names;
    wlan_busnum  = <0x1>;
    wlan_power   = "axp803-dldo1";
    wlan_io_regulator;
    wlan_regon   = <&r_pio PL 5 1 0xffffffff 0xffffffff 0>;
    wlan_hostwake = <&r_pio PL 6 6 0xffffffff 0xffffffff 0>;
    chip_en;
    power_en;
    status      = "okay";
};
```

2.9 蓝牙

2.9.1 [bt]

配置项	配置项含义
compatible	固定值，请勿修改
clocks	32K/24M DCXO 时钟配置表示，如使用外部时钟 32K/外部 24M，则无需配置
bt_power	表示 BT 模组所用的供电，与 wlan_power 相同
bt_io_regulator	表示 BT 模组所用的 IO 供电，与 wlan_regulator 相同
bt_rst_n	表示 BT 模组 power on 控制引脚
status	表示是否使用该模块

配置举例：

```
请看当前目录下的board.dts
bt: bt@0 {
    compatible   = "allwinner,sunxi-bt";
    clocks       = <&clk_losc_out>;
    bt_power     = "axp803-dldo1";
    bt_io_regulator;
    bt_rst_n     = <&r_pio PL 2 1 0xffffffff 0xffffffff 0>;
    status       = "okay";
```

```
};
```

2.9.2 [btlpm]

配置项	配置项含义
compatible	固定值，请勿修改
uart_index	表示 BT 模组使用的硬件通信端口号
bt_wake	表示 BT 模组休眠后被唤醒时的控制引脚
bt_hostwake	表示 BT 模组中断输出引脚，用于唤醒 AP
status	表示是否使用该模块

```
btlpm: btlpm@0 {
    compatible  = "allwinner,sunxi-btlpm";
    uart_index  = <0x1>;
    bt_wake     = <&r_pio PL 4 1 0xffffffff 0xffffffff 1>;
    bt_hostwake = <&r_pio PL 3 6 0xffffffff 0xffffffff 1>;
    status      = "okay";
};
```

2.10 音频

2.10.1 [spdif]

配置项	配置项含义
status	是否开启 spdif，``okay" 打开，``disabled" 关闭

配置举例

```
请看当前目录下的board.dts
spdif:spdif-controller@0x05094000{
    status = "disabled";
};
```

2.10.2 [sndspdif]

配置项	配置项含义
statuts	是否开启 spdif platform，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
sndspdif:sound@1{
    status = "disabled";
};
```

2.10.3 [dmic]

配置项	配置项含义
status	是否开启 spdif，``okay" 打开，``disabled" 关闭

配置举例

```
请看当前目录下的board.dts
dmic:dmic-controller@0x05095000{
    status = "disabled";
};
```

2.10.4 [snddmic]

配置项	配置项含义
statuts	是否开启 dmic platform，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
snddmic:sound@2{
   status = "disabled";
};
```

注意：要生成并注册 dmic 声卡，就必须要把 dmic 和 snddmic 都设置为 ``okay"。

2.10.5 [codec]

配置项	配置项含义
headphonegain	HPOUT 端接喇叭输出时内核启动默认音量设置取值范围：0x0-0x7, max:0x0, min:0x7
lineout_vol	LINEOUT 端接喇叭输出时内核启动默认音量设置取值范围：0x0-0x1f
digital_vol	内核启动默认数字音量设置取值范围:0x0--0x3f
mic1gain	内核启动默认板载 MIC 增益设置取值范围:0x0--0x7
mic2gain	内核启动默认耳机 MIC 增益设置取值范围 0x0--0x7
adcdrc_cfg	ADC 端 DRC 动态范围控制 0x1：开启，0x0：不开启
adchpf_cfg	ADC 端高通滤波 0x1：开启，0x0：不开启
dacdrc_cfg	播放动态音效调节 0x1：开启，0x0：不开启
dachpf_cfg	播放通路高通滤波开启 0x1：开启，0x0：不开启
pa_level	喇叭开启使能电平选择 1：高电平使能 0: 低电平使能
pa_msleep_time	喇叭 pa 进入稳定状态需要的时间
gpio-spk	外部功放使能脚
avcc-supply	音频模块电源 AVCC 管理相关
cpvin-supply	音频模块电源 CPVIN 管理相关
status	``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
codec:codec@0x05096000 {
    /* MIC and headphone gain setting */
    mic1gain    = <0x1F>;
    mic2gain    = <0x1F>;
    /* ADC/DAC DRC/HPF func enabled */
                        /* 0x1:DAP_HP_EN; 0x2:DAP_SPK_EN; 0x3:DAP_HPSPK_EN */
    adcdrc_cfg  = <0x2>;
    adchpf_cfg  = <0x1>;
    dacdrc_cfg  = <0x2>;
    dachpf_cfg  = <0x0>;
    /* Volume about */
    digital_vol     = <0x00>;
    lineout_vol     = <0x1a>;
    headphonegain   = <0x00>;
    dac_digital_vol = <0x1A0A0>;
    /* Pa enabled about */
    pa_level    = <0x01>;
    pa_msleep_time  = <0x78>;
    gpio-spk = <&pio PH 6 1 1 1>;
    /* regulator about */
    avcc-supply = <&reg_aldo1>;
    cpvin-supply = <&reg_eldo1>;
    status = "okay";
};
```

2.10.6 [sndcodec]

配置项	配置项含义
status	是否使用该接口，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
sndcodec:sound@0 {
    status = "okay";
};
```

注意：要生成并注册 codec 声卡，就必须要把 codec 和 sndcodec 都设置为 ``okay"。

2.10.7 [daudio0]

配置项	配置项含义
daudio_master	i2s 主从模式选择，1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) use，2: SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master) not use，3: SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave) not use，4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) use
tdm_config	i2s 模式选择，0：pcm，1：i2s
audio_format	i2s 传输格式选择，1：标准 i2s，2：右对齐，3：左对齐，4：长帧，5：短帧
signal_inversion	i2s 信号翻转选择，1：SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)，2：SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)，3：SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)，4：SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)
word_select_size	数据 word 宽度选择，选值有：16bits/20bits/24bits/32bits
pcm_lrck_period	PCM 模式中一个 LRCK 有多少个 BCLK 选择，选值有：16/32/64/128/256
msb_lsb_first	数据有效位选择，0: msb first，1: lsb first
sign_extend	数据拓展位填补选择，0: zero pending 0 填充，1: sign extend 最后一位填充
slot_width_select	slot 宽度选择，选值有：8 bit width / 16 bit width / 32 bit width
frametype	长短帧选择，0: short frame = 1 clock width，1: long frame = 2 clock width
mclk_div	MCLK 分频系数选择，0: not output(normal setting this) 1/2/4/6/8/12/16/24/32/48/64/96/128/176/192，setting mclk as input clock to external codec, freq is pll_audio/mclk_div
tx_data_mode	PCM 传输模式中 TX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
rx_data_mode	PCM 传输模式中 RX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
status	模块使能开关，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
daudio0:daudio@0x05090000 {
    mclk_div    = <0x01>;
    frametype   = <0x00>;
    tdm_config  = <0x01>;
    sign_extend     = <0x00>;
    tx_data_mode    = <0x00>;
    rx_data_mode    = <0x00>;
    msb_lsb_first   = <0x00>;
    pcm_lrck_period = <0x80>;
    audio_format    = <0x01>;
    daudio_master   = <0x04>;
    signal_inversion  = <0x01>;
    slot_width_select = <0x20>;
    status = "disabled";
};
```

2.10.8 [snddaudio0]

配置项	配置项含义
status	是否使用该接口，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
snddaudio0:sound@3 {
    status = "disabled";
};
```

2.10.9 [daudio1]

配置项	配置项含义
daudio_master	i2s 主从模式选择，1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) use，2: SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master) not use，3: SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave) not use，4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) use
tdm_config	i2s 模式选择，0：pcm，1：i2s
audio_format	i2s 传输格式选择，1：标准 i2s，2：右对齐，3：左对齐，4：长帧，5：短帧
signal_inversion	i2s 信号翻转选择，1：SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)，2：SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)，3：SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)，4：SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)
word_select_size	数据 word 宽度选择，选值有：16bits/20bits/24bits/32bits
pcm_lrck_period	PCM 模式中一个 LRCK 有多少个 BCLK 选择，选值有：16/32/64/128/256
msb_lsb_first	数据有效位选择，0: msb first，1: lsb first
sign_extend	数据拓展位填补选择，0: zero pending 0 填充，1: sign extend 最后一位填充
slot_width_select	slot 宽度选择，选值有：8 bit width / 16 bit width / 32 bit width
frametype	长短帧选择，0: short frame = 1 clock width，1: long frame = 2 clock width
mclk_div	MCLK 分频系数选择，0: not output(normal setting this) 1/2/4/6/8/12/16/24/32/48/64/96/128/176/192，setting mclk as input clock to external codec, freq is pll_audio/mclk_div
tx_data_mode	PCM 传输模式中 TX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
rx_data_mode	PCM 传输模式中 RX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
status	模块使能开关，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
daudio1:daudio@0x05091000 {
    mclk_div    = <0x01>;
    frametype   = <0x00>;
    tdm_config  = <0x01>;
```

```
    sign_extend    = <0x00>;
    tx_data_mode   = <0x00>;
    rx_data_mode   = <0x00>;
    msb_lsb_first  = <0x00>;
    pcm_lrck_period = <0x80>;
    audio_format   = <0x01>;
    daudio_master  = <0x04>;
    signal_inversion = <0x01>;
    slot_width_select = <0x20>;
    status = "disabled";
};
```

2.10.10 [snddaudio1]

配置项	配置项含义
status	是否使用该接口，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
snddaudio1:sound@4 {
    status = "disabled";
};
```

2.10.11 [daudio2]

配置项	配置项含义
daudio_master	i2s 主从模式选择，1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) use，2: SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master) not use，3: SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave) not use，4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) use
tdm_config	i2s 模式选择，0：pcm，1：i2s

配置项	配置项含义
audio_format	i2s 传输格式选择，1：标准 i2s，2：右对齐，3：左对齐，4：长帧，5：短帧
signal_inversion	i2s 信号翻转选择，1：SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)，2：SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)，3：SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)，4：SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)
word_select_size	数据 word 宽度选择，选值有：16bits/20bits/24bits/32bits
pcm_lrck_period	PCM 模式中一个 LRCK 有多少个 BCLK 选择，选值有：16/32/64/128/256
msb_lsb_first	数据有效位选择，0: msb first，1: lsb first
sign_extend	数据拓展位填补选择，0: zero pending 0 填充，1: sign extend 最后一位填充
slot_width_select	slot 宽度选择，选值有：8 bit width / 16 bit width / 32 bit width
frametype	长短帧选择，0: short frame = 1 clock width，1: long frame = 2 clock width
mclk_div	MCLK 分频系数选择，0: not output(normal setting this) 1/2/4/6/8/12/16/24/32/48/64/96/128/176/192，setting mclk as input clock to external codec, freq is pll_audio/mclk_div
tx_data_mode	PCM 传输模式中 TX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
rx_data_mode	PCM 传输模式中 RX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
status	模块使能开关，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
daudio2:daudio@0x05092000 {
    mclk_div    = <0x01>;
    frametype   = <0x00>;
    tdm_config  = <0x01>;
    sign_extend     = <0x00>;
    tx_data_mode    = <0x00>;
    rx_data_mode    = <0x00>;
    msb_lsb_first   = <0x00>;
    pcm_lrck_period = <0x80>;
    audio_format    = <0x01>;
    daudio_master   = <0x04>;
    signal_inversion  = <0x01>;
    slot_width_select = <0x20>;
```

```
    status = "disabled";
};
```

2.10.12 [snddaudio2]

配置项	配置项含义
status	是否使用该接口，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
snddaudio2:sound@5 {
    status = "disabled";
};
```

2.10.13 [daudio3]

配置项	配置项含义
daudio_master	i2s 主从模式选择，1: SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) use，2: SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master) not use，3: SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave) not use，4: SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) use
tdm_config	i2s 模式选择，0：pcm，1：i2s
audio_format	i2s 传输格式选择，1：标准 i2s，2：右对齐，3：左对齐，4：长帧，5：短帧
signal_inversion	i2s 信号翻转选择，1：SND_SOC_DAIFMT_NB_NF(normal bit clock + frame)，2：SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM)，3：SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM)，4：SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM)
word_select_size	数据 word 宽度选择，选值有：16bits/20bits/24bits/32bits

配置项	配置项含义
pcm_lrck_period	PCM 模式中一个 LRCK 有多少个 BCLK 选择，选值有：16/32/64/128/256
msb_lsb_first	数据有效位选择，0: msb first，1: lsb first
sign_extend	数据拓展位填补选择，0: zero pending 0 填充，1: sign extend 最后一位填充
slot_width_select	slot 宽度选择，选值有：8 bit width / 16 bit width / 32 bit width
frametype	长短帧选择，0: short frame = 1 clock width，1: long frame = 2 clock width
mclk_div	MCLK 分频系数选择，0: not output(normal setting this) 1/2/4/6/8/12/16/24/32/48/64/96/128/176/192，setting mclk as input clock to external codec, freq is pll_audio/mclk_div
tx_data_mode	PCM 传输模式中 TX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
rx_data_mode	PCM 传输模式中 RX 选择数据格式，0:16bit linear PCM，1：8bit linear PCM
status	模块使能开关，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
daudio3:daudio@0x05093000 {
    mclk_div    = <0x01>;
    frametype   = <0x00>;
    tdm_config  = <0x01>;
    sign_extend     = <0x00>;
    tx_data_mode    = <0x00>;
    rx_data_mode    = <0x00>;
    msb_lsb_first   = <0x00>;
    pcm_lrck_period = <0x80>;
    audio_format    = <0x01>;
    daudio_master   = <0x04>;
    signal_inversion  = <0x01>;
    slot_width_select = <0x20>;
    status = "disabled";
};
```

2.10.14 [snddaudio3]

配置项	配置项含义
status	是否使用该接口，``okay" 打开，``disabled" 关闭

配置举例：

```
请看当前目录下的board.dts
snddaudio0:sound@6 {
   status = "disabled";
};
```

注意：要生成并注册 Daudio 声卡，就必须要把相应的 daudio、snddaudio 都设置为 ``okay"。

2.11 动态切换打印

2.11.1 [auto_print]

配置项	配置项含义
auto_print_used	TF 卡与 UART 动态切换使能

配置举例：

```
auto_print_used = 1
```

2.12 gsensor

配置项	配置项含义
device_type	设备类型，这里填 gsensor
gsensor_twi_id	gsensor 所在的 i2c id

配置项	配置项含义
gsensor_twi_addr	gsensor 在 i2c 中的地址
gsensor_int1	gsensor 的中断管脚配置
gsensor-supply	gsensor 所使用的电源
gsensor_vcc_io_val	gsensor 所需要的电压。

配置举例：

```
请看当前目录下的board.dts
twi1: twi@0x05002400{
    clock-frequency = <200000>;
    pinctrl-0 = <&twi1_pins_a>;
    pinctrl-1 = <&twi1_pins_b>;
    status = "okay";
    gsensor {
        compatible = "allwinner,sc7a20";
        reg = <0x19>;
        device_type = "gsensor";
        status = "okay";
        gsensor_twi_id = <0x1>;
        gsensor_twi_addr = <0x19>;
        gsensor_int1 = <&pio PH 11 6 1 0xffffffff 0xffffffff>;
        gsensor-supply = <&reg_dcdc1>;
        gsensor_vcc_io_val = <3300>;
    };
};
```

2.13 lightsensor

配置项	配置项含义
device_type	设备类型，这里填 lightsensor
ls_twi_id	lightsensor 所在的 i2c id
ls_twi_addr	lightsensor 在 i2c 中的地址
ls_int	lightsensor 的中断管脚配置
lightsensor-supply	lightsensor 所使用的电源

配置举例：

```
twi1: twi@0x05002400{
    clock-frequency = <200000>;
    pinctrl-0 = <&twi1_pins_a>;
    pinctrl-1 = <&twi1_pins_b>;
    status = "okay";
    lightsensor {
        compatible = "allwinner,stk3x1x";
        reg = <0x48>;
        device_type = "lightsensor";
        status = "okay";
        ls_twi_id = <0x1>;
        ls_twi_addr = <0x48>;
        ls_int = <&pio PH 4 6 1 0xffffffff 0xffffffff>;
        lightsensor-supply = <&reg_dcdc1>;
    };
};
```

2.14 CTP

配置项	配置项含义
device_type	设备类型，这里填 ctp
ctp_name	标示 CTP 的 fw 信息，目前 gslX680，gslX680new 和 gt9xx 的驱动需要用到此项
ctp_twi_id	ctp 所在的 i2c id
ctp_twi_addr	ctp 在 i2c 中的地址
ctp_screen_max_x	ctp 的 x 轴范围
ctp_screen_max_y	ctp 的 y 轴范围
ctp_revert_x_flag	ctp 的 x 轴是否翻转
ctp_revert_y_flag	ctp 的 y 轴是否翻转
ctp_exchange_x_y_flag	ctp 的 x 轴、y 轴是否互换
ctp_int_port	ctp 的中断管脚配置
ctp_wakeup	ctp 的唤醒管脚配置
ctp-supply	ctp 所使用的电源
ctp_power_ldo_vol	ctp 所需要的电压

配置举例：

```
请看当前目录下的board.dts
twi0: twi@0x05002000{
    clock-frequency = <400000>;
    pinctrl-0 = <&twi0_pins_a>;
    pinctrl-1 = <&twi0_pins_b>;
    status = "okay";
    ctp {
        compatible = "allwinner,gslX680";
        reg = <0x40>;
        device_type = "ctp";
        status = "okay";
        ctp_name = "gslX680_3676_1280x800";
        ctp_twi_id = <0x0>;
        ctp_twi_addr = <0x40>;
        ctp_screen_max_x = <0x320>;
        ctp_screen_max_y = <0x500>;
        ctp_revert_x_flag = <1>;
        ctp_revert_y_flag = <1>;
        ctp_exchange_x_y_flag = <0x1>;
        ctp_int_port = <&pio PH 9 6 0xffffffff 0xffffffff 0>;
        ctp_wakeup = <&pio PH 10 1 0xffffffff 0xffffffff 1>;
        ctp-supply = <&reg_ldoio0>;
        ctp_power_ldo_vol = <3300>;
    };
};
```

2.15 摄像头

2.15.1 [vind0]

配置项	配置项含义
vind0_clk	CSI 的时钟频率
vind0_clk	ISP 的时钟频率
status	模块使能
device_type	子设备设备的类型
actuatorX_name	对焦马达名称，要和驱动中定义一致
actuatorX_slave	对焦马达设备 I2C 地址
actuatorX_af_pwdn	对焦马达启动或关闭控制 IO 口
actuatorX_afvdd	对焦马达供电

配置项	配置项含义
actuatorX_afvdd_vol	对焦马达供电电压
flashX_type	闪光灯类型
flashX_en	闪光灯控制引脚
flashX_mode	闪光灯工作模式
flashX_flvdd	闪光灯供电口
flashX_flvdd_vol	闪光灯供电电压
device_id	闪光灯 ID
sensorX_mname	摄像头名称，要和驱动中定义一致
sensorX_twi_cci_id	使用的 TWI（I2C）编号
sensorX_twi_addr	设备 I2C 地址
sensorX_mclk_id	使用的时钟编号，0 或 1
sensorX_pos	相机位置，``front" 表示前摄，``rear" 表示后摄
sensorX_isp_used	ISP 使能
sensorX_fmt	图像格式选择，0 是 YUV，1 是 RAW
sensorX_stby_mode	休眠模式
sensorX_vflip	垂直镜像使能
sensorX_hflip	水平镜像使能
sensorX_iovdd-supply	摄像头控制 IO 有效电压输入端
sensorX_iovdd_vol	摄像头控制 IO 有效电压
sensorX_power_en	摄像头控制 IO
sensorX_reset	摄像头控制 IO
vincX_csi_sel	CSI 选择，0 或 1
vincX_mipi_sel	MIPI 选择，0 或 1，和 CSI 选择一致
vincX_isp_sel	ISP 选择，0 或 1
vincX_isp_tx_ch	ISP 通道选择，0 到 3
vincX_tdm_rx_sel	TDM 分时复用模块通道选择，0 或者 1，0xff 为不启用
vincX_rear_sensor_sel	前摄的摄像头选择
vincX_front_sensor_sel	后摄的摄像头选择
vincX_sensor_list	sensor list 功能使能

配置举例：

```
请看当前目录下的board.dts
vind0:vind@0 {
    vind0_clk = <336000000>;
    vind0_isp = <300000000>;
    status = "okay";

    actuator0:actuator@0 {
        device_type = "actuator0";
        actuator0_name = "ad5820_act";
        actuator0_slave = <0x18>;
        actuator0_af_pwdn = <>;
        actuator0_afvdd = "afvcc-csi";
        actuator0_afvdd_vol = <2800000>;
        status = "disabled";
    };
    flash0:flash@0 {
        device_type = "flash0";
        flash0_type = <2>;
        flash0_en = <&r_pio PL 11 1 0 1 0>;
        flash0_mode = <>;
        flash0_flvdd = "";
        flash0_flvdd_vol = <>;
        device_id = <0>;
        status = "disabled";
    };
    sensor0:sensor@0 {
        device_type = "sensor0";
        sensor0_mname = "gc2385_mipi";
        sensor0_twi_cci_id = <2>;
        sensor0_twi_addr = <0x6e>;
        sensor0_mclk_id = <0>;
        sensor0_pos = "rear";
        sensor0_isp_used = <1>;
        sensor0_fmt = <1>;
        sensor0_stby_mode = <0>;
        sensor0_vflip = <0>;
        sensor0_hflip = <0>;
        sensor0_iovdd-supply = <&reg_dldo2>;
        sensor0_iovdd_vol = <1800000>;
        sensor0_avdd-supply = <&reg_dldo3>;
        sensor0_avdd_vol = <2800000>;
        sensor0_dvdd-supply = <&reg_eldo2>;
        sensor0_dvdd_vol = <1800000>;
        sensor0_power_en = <>;
        sensor0_reset = <&pio PE 9 1 0 1 0>;
        sensor0_pwdn = <&pio PE 8 1 0 1 0>;
        status  = "okay";
    };
    sensor1:sensor@1 {
        device_type = "sensor1";
```

```
    sensor1_mname = "gc030a_mipi";
    sensor1_twi_cci_id = <2>;
    sensor1_twi_addr = <0x42>;
    sensor1_mclk_id = <0>;
    sensor1_pos = "front";
    sensor1_isp_used = <1>;
    sensor1_fmt = <1>;
    sensor1_stby_mode = <0>;
    sensor1_vflip = <0>;
    sensor1_hflip = <0>;
    sensor1_iovdd-supply = <&reg_dldo2>;
    sensor1_iovdd_vol = <1800000>;
    sensor1_avdd-supply = <&reg_dldo3>;
    sensor1_avdd_vol = <2800000>;
    sensor1_dvdd-supply = <&reg_eldo2>;
    sensor1_dvdd_vol = <1800000>;
    sensor0_power_en = <>;
    sensor1_reset = <&pio PE 7 1 0 1 0>;
    sensor1_pwdn = <&pio PE 6 1 0 1 0>;
    status  = "okay";
};
vinc0:vinc@0 {
    vinc0_csi_sel = <0>;
    vinc0_mipi_sel = <0>;
    vinc0_isp_sel = <0>;
    vinc0_isp_tx_ch = <0>;
    vinc0_tdm_rx_sel = <0xff>;
    vinc0_rear_sensor_sel = <0>;
    vinc0_front_sensor_sel = <1>;
    vinc0_sensor_list = <0>;
    status = "okay";
};
vinc1:vinc@1 {
    vinc1_csi_sel = <0>;
    vinc1_mipi_sel = <0>;
    vinc1_isp_sel = <0>;
    vinc1_isp_tx_ch = <0>;
    vinc1_tdm_rx_sel = <0xff>;
    vinc1_rear_sensor_sel = <0>;
    vinc1_front_sensor_sel = <1>;
    vinc1_sensor_list = <0>;
    status = "okay";
};
vinc2:vinc@2 {
    vinc2_csi_sel = <1>;
    vinc2_mipi_sel = <1>;
    vinc2_isp_sel = <0>;
    vinc2_isp_tx_ch = <0>;
    vinc2_tdm_rx_sel = <0xff>;
    vinc2_rear_sensor_sel = <0>;
    vinc2_front_sensor_sel = <1>;
    vinc2_sensor_list = <0>;
```

```
    status = "okay";
};
vinc3:vinc@3 {
    vinc3_csi_sel = <1>;
    vinc3_mipi_sel = <1>;
    vinc3_isp_sel = <0>;
    vinc3_isp_tx_ch = <0>;
    vinc3_tdm_rx_sel = <0xff>;
    vinc3_rear_sensor_sel = <0>;
    vinc3_front_sensor_sel = <1>;
    vinc3_sensor_list = <0>;
    status = "okay";
};
```

2.16 安全

2.16.1 [secure]

配置项	配置项含义
dram_region_mbytes	预留 80MB 内存给与 Widevine L1 使用
drm_region_mbytes	暂未使用，无需关注
drm_region_start_mbytes	暂未使用，无需关注

配置举例：

```
;dram_region_mbytes      = 80
;drm_region_mbytes       = 0
;drm_region_start_mbytes = 0
```

3. FAQ

3.1 sys_config.fex 跟 dts 配置同一个节点，会冲突吗？

答：sys_config.fex 的配置会覆盖 dts 的配置。

3.2 为什么创建跟 dts 同名的节点，但是驱动一直加载不成功？

```
example:
[test_first]
test_used = 1
test_para = "first_test"
```

答：这是因为该节点的 used 节点命名问题导致该节点可能没打开，used 节点的命名必须为" 节点主键"+"_used"，这样才能有效的覆盖 dts 里面 status 状态，避免 dts 里面 test_first 节点的 status 的状态为 disabled，导致该节点不可用，正确配置如下：

```
example:
[test_first]
test_first_used = 1
test_para = "first_test"
```

3.3 如果看到同一 pin 脚被两个节点复用，是否有问题？

答：这个问题有以下两种可能。（1）首先判断两个节点的有没有同时开启，如果没有同时开启，就不会有问题。（2）如果两个节点同时开启，需要判断以下该节点被调用的阶段是不是相同。如果两个节点被调用的阶段不同，则没问题。例如以下例子 twi 在 boot 阶段调用，uart0 在 Linux kernel 调用，则此复用不会出现问题。

```
example:
[twi]
twi_port       = 0
twi_scl        = port:PH0<2><default><default><default>
twi_sda        = port:PH1<2><default><default><default>

[uart0]
uart0_used     = 1
uart0_port     = 0
uart0_type     = 2
uart0_tx       = port:PH0<3><1><default><default>
uart0_rx       = port:PH1<3><1><default><default>
```

注意：如果两个节点被调用的阶段相同，则不允许复用。

3.4 为何 sys_config.fex 相比以往的 Android 版本配置少了这么多？

答：自 Android 10 开始，使用 longan 设计以后，大部分 BSP 的配置将不再体现在 sys_config.fex，而是写到 board.dts 文件，但是 sys_config.fex 依然保留给客户配置，方法不变。

4. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.