



sys_config 配置说明文档

1.0
2019.12.5

文档履历

版本号	日期	制/修订人	内容描述
1.0	2019.12.5		正式版本

目录

1. 前言	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
2. 节点配置说明	2
2.1 系统 (SYSTEM)	2
2.1.1 [product]	2
2.1.2 [platform]	2
2.1.3 [target]	3
2.1.4 [ir_boot_recovery]	4
2.1.5 [box_start_os]	4
2.1.6 [box_standby_led]	5
2.1.7 [card_boot]	5
2.1.8 [key_boot_recovery]	6
2.1.9 [boot_init_gpio]	6
2.1.10 [card0_boot_para]	7
2.1.11 [card2_boot_para]	8
2.1.12 [twi_para]	9
2.1.13 [uart_para]	9
2.1.14 [jtag_para]	10

2.2 DRAM 配置	11
2.2.1 [dram_para]	11
2.3 I2C 总线	12
2.3.1 [twi0]	12
2.3.2 [twi1]	12
2.3.3 [twi2]	13
2.4 UART	13
2.4.1 [uart0]	13
2.5 NAND FLASH	14
2.5.1 [nand0_para]	14
2.6 显示	15
2.6.1 [disp]	15
2.7 HDMI	18
2.7.1 [hdmi]	18
2.8 SD/MMC	19
2.8.1 [sdc0]	19
2.8.2 [sdc1]	20
2.8.3 [sdc2]	21
2.8.4 [gpio_para]	22
2.9 USB 控制器标志	23
2.9.1 [usbc0]	23
2.9.2 [usbc1]	25

2.10 WIFI	25
2.10.1 [wlan]	25
2.11 蓝牙	26
2.11.1 [bt]	26
2.11.2 [bt_lpm]	27
2.12 数字音频总线 (S/PDIF)	28
2.12.1 [spdif]	28
2.12.2 [sndspdif]	28
2.13 数字音频总线 (TDM)	29
2.13.1 [sndahub]	29
2.13.2 [ahub_daudio0/1]	29
2.13.3 [sndhdmi]	29
2.13.4 [snddaudio0]	30
2.13.5 [daudio0]	30
2.14 红外	32
2.14.1 [s_cir0]	32
2.15 动态切换打印	35
2.15.1 [auto_print]	35
2.16 安全	35
2.16.1 [secure]	35
3. FAQ	37
3.1 sys_config.fex 跟 dts 配置同一个节点，会冲突吗？	37

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？	37
3.3 如果看到同一 pin 脚被两个节点复用，是否有问题？	37
4. Declaration	39

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 H616 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（default）：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
advert_enable	0：不反转 logo 1：反转 logo，只能在多核启动下生效

配置举例：

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

2.1.4 [ir_boot_recovery]

配置项	配置项含义
ir_boot_recovery_used	1: 启动时通过 ir 判断是否进入 Android recovery mode 0: close
ir_work_mode	遥控器 0 刷机, 1 一键恢复, 2 安卓 recovery, 3 安卓恢复出厂设置
ir_press_times	ir 遥控器连续按几次才生效, 如果不设置默认为按 1 次生效
ir_detect_time	ir 遥控检测时间, 单位:ms, 如果不设置默认为 3000ms
ir_key_no_duplicate	ir 遥控按键是否可重复, 0: 可重复 (默认), 1: 不可重复;
ir_recovery_key_code0	遥控器检测按键值
ir_addr_code0	遥控器具体按键值

配置举例:

```
ir_boot_recovery_used = 1
ir_work_mode = 1
ir_press_times = 2
ir_detect_time = 1
ir_key_no_duplicate = 0
ir_recovery_key_code0 = 0x11
ir_addr_code0 = 0xfe01
ir_recovery_key_code1 = 0x19
ir_addr_code1 = 0xfe01
ir_recovery_key_code2 = 0x4c
ir_addr_code2 = 0xfe01
ir_recovery_key_code3 = 0x00
ir_addr_code3 = 0xfe01
```

2.1.5 [box_start_os]

配置项	配置项含义
used	是否启用该项功能: 1: 启用 0: 不启用
start_type	是否上电启动系统 1: 直接启动系统; 0: 上电不允许直接启动系统
irkey_used	是否启用 ir 控制启动: 1: 启用 ir 按键启动 0: 禁用 ir 按键启动
pmukey_used	1: 启用 PMU power key 启动 0: 禁用 PMU power key

配置举例：

```
used = 1
start_type = 1
irkey_used = 1
pmukey_used = 0
```

2.1.6 [box_standby_led]

配置项	配置项含义
gpio0	进入假关机，设置 led0 状态为开
gpio1	进入假关机，设置 led1 状态为关
gpio2	进入假关机，设置 led2 状态为关

配置举例：

```
gpio0 = port:PL07<1><default><default><0>
gpio1 = port:PL04<1><default><default><1>
gpio2 = port:PL03<1><default><default><0>
```

注意：用于系统进入假关机设置 led 状态，必须以 gpio0、gpio1、gpio2 递增去命名，才可以申请到对应的 gpio

2.1.7 [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
```

2.1.8 [key_boot_recovery]

配置项	配置项含义
used	是否开启一键 recovery 功能，1：开启 0：禁用
press_mode_enable	长短按模式使能，1：开启，0：关闭
key_work_mode	模式选择，0：刷机，1：一键恢复 (uboot 阶段)，2：安卓 recovery,3：安卓恢复出厂设置.
short_press_mode	短按触发的模式，选项同上
long_press_mode	长按触发的模式，选项同上
key_press_time	定义长按的时间，单位：毫秒
recovery_key	按键配置

配置举例：

```
recovery_key_used = 1
press_mode_enable = 0
key_work_mode = 0
short_press_mode = 0
long_press_mode = 1
key_press_time = 2000
recovery_key = port:PH09<0><default><default><default>
```

2.1.9 [boot_init_gpio]

配置项	配置项含义
boot_init_gpio_used gpio0	Boot 启动阶段初始化 GPIO，1：开启 0：禁用 GPIO 配置

配置项	配置项含义
gpio1	GPIO 配置
gpio2	GPIO 配置

配置举例：

```
boot_init_gpio_used = 1
gpio0 = port:PL07<1><default><default><1>
gpio1 = port:PA03<1><default><default><0>
gpio2 = port:PH02<1><default><default><1>
```

一般用于系统启动 LED GPIO 初始化，必须以 gpio0、gpio1、gpio2 递增去命名，才可以申请到对应的 gpio

2.1.10 [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
sd_c_d1	sd 卡数据 1 线信号的 GPIO 配置
sd_c_d0	sd 卡数据 0 线信号的 GPIO 配置
sd_c_clk	sd 卡时钟信号的 GPIO 配置
sd_c_cmd	sd 命令信号的 GPIO 配置
sd_c_d3	sd 卡数据 3 线信号的 GPIO 配置
sd_c_d2	sd 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>

```

2.1.11 [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 信号的 GPIO 配置
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_ex_dly_used = 2
sdc_io_1v8 = 1
;sdc_dis_host_caps = 0x100
;sdc_type = "tm4"
    
```

2.1.12 [twi_para]

配置项	配置项含义
twi_port	Boot 的 twi 控制器编号
twi_scl	Boot 的 twi 的时钟的 GPIO 配置
twi_sda	Boot 的 twi 的数据的 GPIO 配置

配置举例：

```

twi_port = 0
twi_scl = port:PH14<2><default><default><default>
twi_sda = port:PH15<2><default><default><default>
    
```

2.1.13 [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:PH00<2><1><default><default>
uart_debug_rx = port:PH01<2><1><default><default>
```

2.1.14 [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：2 为 DDR2 3 为 DDR3, 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_para]
dram_clk = 672
dram_type = 3
dram_dx_odt = 0x08080808
dram_dx_dri = 0x0e0e0e0e
dram_ca_dri = 0x1c1c
dram_odt_en = 1
dram_para1 = 0x310b
dram_para2 = 0x0000
dram_mr0 = 0x840
dram_mr1 = 0x4
dram_mr2 = 0x8
dram_mr3 = 0x0
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 = 0x33808080
dram_tpr10 = 0x012f0000
dram_tpr11 = 0xffedddb
dram_tpr12 = 0xeddca998
dram_tpr13 = 0x40
```

2.3 I2C 总线

2.3.1 [twi0]

配置项	配置项含义
twi0_used	TWI 使用控制：1 使用，0 不用
twi0_scl	TWI SCK 的 GPIO 配置
twi0_sda	TWI SDA 的 GPIO 配置

配置举例：

```
twi0_used = 0
twi_scl = port:PA11<2><default><default><default>
twi_sda = port:PA12<2><default><default><default>
```

2.3.2 [twi1]

配置项	配置项含义
twi1_used	TWI 使用控制：1 使用，0 不用
twi1_scl	TWI SCK 的 GPIO 配置
twi1_sda	TWI SDA 的 GPIO 配置

配置举例：

```
twi1_used = 1
twi_scl = port:PA18<3><default><default><default>
twi_sda = port:PA19<3><default><default><default>
```

2.3.3 [twi2]

配置项	配置项含义
twi2_used	TWI 使用控制：1 使用，0 不用
twi2_scl	TWI SCK 的 GPIO 配置
twi2_sda	TWI SDA 的 GPIO 配置

配置举例：

```
twi2_used = 0
twi2_scl = port:PE12<3><default><default><default>
twi2_sda = port:PE13<3><default><default><default>
```

2.4 UART

2.4.1 [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:PH00<2><1><default><default>
uart0_rx = port:PH01<2><1><default><default>
```

2.5 NAND FLASH

2.5.1 [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

配置项	配置项含义
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
```

2.6 显示

2.6.1 [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 8:1080p24 9:1080p50 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:BGR)
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
fb1_format	fb1 的格式 (0:ARGB 1:ABGR 2:RGBA 3:BGR)
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@01000000 {
    disp_init_enable = <1>;
    disp_mode = <0>;

```

```

screen0_output_type = <3>;
screen0_output_mode = <10>;
screen0_output_format = <0>;
screen0_output_bits = <0>;
screen0_output_eotf = <4>;
screen0_output_cs = <257>;
screen0_output_dvi_hdmi = <2>;
screen0_output_range = <2>;
screen0_output_scan = <0>;
screen0_output_aspect_ratio = <8>;
    
```

```

screen1_output_type = <2>;
screen1_output_mode = <11>;
screen1_output_format = <1>;
screen1_output_bits = <0>;
screen1_output_eotf = <4>;
screen1_output_cs = <260>;
screen1_output_dvi_hdmi = <0>;
screen1_output_range = <2>;
screen1_output_scan = <0>;
screen1_output_aspect_ratio = <8>;
dev0_output_type = <4>;
dev0_output_mode = <10>;
dev0_screen_id = <0>;
dev0_do_hpd = <1>;
    
```

```

dev1_output_type = <2>;
dev1_output_mode = <11>;
dev1_screen_id = <1>;
dev1_do_hpd = <1>;
    
```

```

dev2_output_type = <0>;
def_output_dev = <0>;
hdmi_mode_check = <1>;
    
```

```

fb0_format = <0>;
fb0_width = <1280>;
fb0_height = <720>;
    
```

```

fb1_format = <0>;
fb1_width = <0>;
fb1_height = <0>;
    
```

```

chn_cfg_mode = <1>;
    
```

```

disp_para_zone = <1>;
/* VCC-LCD */
/*dc1sw-supply = <&reg_sw>;*/
/* VCC-LVDS and VCC-HDMI */
/*bldo1-supply = <&reg_bldo1>;*/
/* VCC-TV */
/*cldo4-supply = <&reg_cldo4>;*/
    
```

```
};
```

2.7 HDMI

2.7.1 [hdm]i

配置项	配置项含义
hdmi_used	是否使用 hdmi。1: 使用; 0: 不使用
hdmi_power	内核阶段 hdmi 电源配置
hdmi_hdcp_enable	是否使能 hdcp
hdmi_cts_compatibility	cts 兼容性使能设置
hdmi_hpd_mask	hpd 掩码设置
hdmi_cec_support	是否支持 CEC 功能

配置举例:

请看当前目录下的board.dts

```
hdmi: hdmi@06000000 {
    hdmi_used = <1>;
    hdmi_power_cnt = <2>;
    hdmi_power0 = "vcc-hdmi";
    hdmi_power1 = "vdd-hdmi";
    hdmi_hdcp_enable = <1>;
    hdmi_hdcp22_enable = <1>;
    hdmi_cts_compatibility = <0>;
    hdmi_cec_support = <1>;
    hdmi_cec_super_standby = <0>;
    hdmi_skip_bootedid = <1>;
    ddc_en_io_ctrl = <0>;
    power_io_ctrl = <0>;
};
```


2.8 SD/MMC

2.8.1 [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
sdc0_d1	SDC DATA1 的 GPIO 配置
sdc0_d0	SDC DATA0 的 GPIO 配置
sdc0_clk	SDC DATA1 的 GPIO 配置
sdc0_d1	SDC CLK 的 GPIO 配置
sdc0_cmd	SDC CMD 的 GPIO 配置
sdc0_d3	SDC DATA3 的 GPIO 配置
sdc0_d2	SDC DATA2 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
vmmc	SDC 供电电源配置
vqmmc	SDC IO 供电电源配置
vdmcc	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 {
    pinctrl-0 = <&sdc0_pins_a>;
    bus-width = <4>;
    cd-gpios = <&pio PF 6 6 1 3 0xffffffff>;
    cd-used-24M;
    /*non-removable;*/
    /*broken-cd;*/
    /*cd-inverted*/
    /*data3-detect;*/
    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_dldo1>;
vqmmc33sw-supply = <&reg_dldo1>;
vdm33sw-supply = <&reg_dldo1>;
vqmmc18sw-supply = <&reg_aldo1>;
vdm18sw-supply = <&reg_aldo1>;
status = "okay";
};

```

2.8.2 [sdci1]

配置项	配置项含义
sdci1_used	SDC 使用控制：1 使用，0 不用
bus-width	位宽：1-1bit, 4-4bit
sdci1_d1	SDC DATA1 的 GPIO 配置
sdci1_d0	SDC DATA0 的 GPIO 配置
sdci1_clk	SDC CLK 的 GPIO 配置
sdci1_cmd	SDC CMD 的 GPIO 配置
sdci1_d3	SDC DATA3 的 GPIO 配置
sdci1_d2	SDC DATA2 的 GPIO 配置
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 {
    pinctrl-0 = <&sdc1_pins_a>;
    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 = <150000000>;
    ctl-spec-caps = <0x8>;
    status = "okay";
};
```

2.8.3 [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
sdc1_d0	位宽：1-1bit，4-4bit 置
sdc2_ds	SDC eMMC Data Strobe 的 GPIO 配置置
sdc2_d1	SDC DATA1 的 GPIO 配置置
sdc2_d0	SDC DATA0 的 GPIO 配置
sdc2_clk	SDC CLK 的 GPIO 配置置
sdc2_cmd	SDC CMD 的 GPIO 配置
sdc2_d3	SDC DATA3 的 GPIO 配置
sdc2_d2	SDC DATA2 的 GPIO 配置
sdc2_d4	SDC DATA4GPIO 配置
sdc2_d5	SDC DATA5GPIO 配置
sdc2_d6	SDC DATA6GPIO 配置

配置项	配置项含义
sdc2_d7	SDC DATA7GPIO 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 {
    pinctrl-l0 = <&sdc2_pins_a &sdc2_pins_c>;
    non-removable;
    bus-width = <8>;
    mmc-ddr-1_8v;
    mmc-hs200-1_8v;
    mmc-hs400-1_8v;
    no-sdio;
    no-sd;
    cap-mmc-highspeed;
    sunxi-power-save-mode;
    sunxi-dis-signal-vol-sw;
    max-frequency = <100000000>;
    vmmc-supply = <&reg_dldo1>;
    vqmmc-supply = <&reg_aldo1>;
    status = "disabled";
};
```

2.8.4 [gpio_para]

配置项	配置项含义
compatible	该配置的名字
gpio_used	内核 GPIO 初始化使能功能, 1: 开启 0: 禁用
gpio_num	GPIO 引脚数目

配置项	配置项含义
gpio_pin_1	GPIO 引脚配置
gpio_pin_2	GPIO 引脚配置
gpio_pin_3	GPIO 引脚配置
normal_led	正常状态灯使用的 GPIO
standby_led	休眠状态灯使用的 GPIO
network_led	休眠状态灯使用的 GPIO
easy_light_used	是否为顶层接口开启硬件屏蔽，1：开启 0：禁用
normal_led_light	normal_led 灯亮 pin 口状态，1：高电平 0：低电平
standby_led_light	standby_led 灯亮 pin 口状态，1：高电平 0：低电平
network_led_light	network_led 灯亮 pin 口状态，1：高电平 0：低电平

配置举例：

请看当前目录下的board.dts

```
gpio_para {
    device_type = "gpio_para";
    status = "okay";
    compatible = "allwinner,sunxi-init-gpio";
    gpio_num = <2>;
    gpio_pin_1 = <&pio PH 7 1 0xffffffff 0xffffffff 1>;
    gpio_pin_2 = <&pio PH 6 1 0xffffffff 0xffffffff 0>;
    normal_led = "gpio_pin_1";
    standby_led = "gpio_pin_2";
    easy_light_used = <1>;
    normal_led_light = <1>;
    standby_led_light = <1>;
};
```

2.9 USB 控制器标志

2.9.1 [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
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

```
usb0:usb0@0 {
    device_type = "usb0";
    usb_port_type = <0x0>;
    usb_detect_type = <0x1>;
    usb_detect_mode = <0x0>;
    usb_id_gpio;
    usb_det_vbus_gpio;
    usb_drv_vbus_gpio;
    usb_host_init_state = <0x0>;
    usb_regulator_io = "nocare";
    usb_wakeup_suspend = <0x2>;
    usb_luns = <0x3>;
    usb_serial_unique = <0x0>;
    usb_serial_number = "20080411";
    status = "okay";
}
```

```
};
```

2.9.2 [usbc1]

配置项	配置项含义
usbl_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_drv_vbus_gpio = <&pio PH 8 0 1 0xffffffff 0xffffffff>;  
    usb_host_init_state = <0x1>;  
    usb_regulator_io = "nocare";  
    usb_wakeup_suspend = <0x2>;  
    status = "okay";  
};
```

2.10 WIFI

2.10.1 [wlan]

配置项	配置项含义
wlan_used	是否要使用 wifi
wlan_busnum	所使用的 USB 号，如使用的是 USB3，则此值为 3
wlan_io_regulator	wifi 模组 io 使用哪一路 AXP 供电
chip_en	WiFi 模组使能引脚，硬件未使用时不配置；
power_en	power_en 用于 Wi-Fi / 蓝牙模块外部的电源开关
wlan_regon	Wifi 使能脚
wlan_hostwake	wifi 唤醒主控脚

配置举例：

请看当前目录下的 board.dts

```
wlan:wlan {
    compatible = "allwinner,sunxi-wlan";
    clocks = <&clk_losc_out>;
    pinctrl-0 = <&clk_losc_pins_a>;
    pinctrl-names = "default";
    wlan_busnum = <0x1>;
    wlan_power;
    wlan_io_regulator;
    wlan_regon = <&pio PG 18 1 0xffffffff 0xffffffff 0>;
    wlan_hostwake = <&pio PG 15 6 0xffffffff 0xffffffff 0>;
    chip_en;
    power_en;
    status = "okay";
};
```

2.11 蓝牙

2.11.1 [bt]

配置项	配置项含义
bt_used	蓝牙使用控制：1 使用，0 不用
bt_power	bt 模组使用哪一路 AXP 供电（通常情况下和 wifi 相同）
bt_io_regulator	bt 模组 io 使用哪一路 AXP 供电（通常情况下和 wifi 相同）

配置项	配置项含义
bt_rst_n	bt 使能脚

配置举例：

请看当前目录下的board.dts

```
bt:bt {
    compatible = "allwinner,sunxi-bt";
    clocks = <&clk_losc_out>;
    bt_power;
    bt_io_regulator;
    bt_rst_n = <&pio PG 19 1 0xffffffff 0xffffffff 0>;
    status = "okay";
};
```

2.11.2 [btlpm]

配置项	配置项含义
btlpm_used	蓝牙低功耗使用控制：1 使用，0 不用
uart_index	使用的串口序号，如使用 ttyS1，则此值为 1
bt_wake	主控唤醒 bt 引脚
bt_hostwake	bt 唤醒主控引脚

配置举例

```
btlpm:btlpm {
    compatible = "allwinner,sunxi-btlpm";
    uart_index = <0x1>;
    bt_wake = <&pio PG 17 1 0xffffffff 0xffffffff 1>;
    bt_hostwake = <&pio PG 16 6 0xffffffff 0xffffffff 0>;
    status = "okay";
};
```

2.12 数字音频总线（S/PDIF）

2.12.1 [spdif]

配置项	配置项含义
spdif_used	是否开启 spdif，1：开启，0：不开启

配置举例

```
请看当前目录下的board.dts
    spdif:spdif-controller@0x05093000{
        status = "okay";
    };
```

2.12.2 [sndspdif]

配置项	配置项含义
sndspdif_used	是否开启 spdif platform，1：开启，0：不开启

配置举例：

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

注意：要生成并注册 spdif 声卡，就必须要把 spdif_used 和 sndspdif_used 都设置为 1。

2.13 数字音频总线（TDM）

2.13.1 [sndahub]

配置项	配置项含义
sndahub_used	是否开启 sndahub_used, 1: 开启, 0: 不开启

配置举例:

```
请看当前目录下的board.dts
sndahub:sound@7{
    status = "okay";
};
```

2.13.2 [ahub_daudio0/1]

配置项	配置项含义
ahub_daudio1_used	是否开启 ahub_daudio1_used, 1: 开启, 0: 不开启

配置举例:

```
请看当前目录下的board.dts
ahub_daudio0:ahub_daudio0@0x05097000{
    status = "okay";
};
```

2.13.3 [sndhdm1]

配置项	配置项含义
sndhdm_i_used	是否开启 sndhdm_i, 1: 开启, 0: 不开启

配置举例:

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

注意: 要生成并注册 HDMI 声卡, 就必须要把 sndahub_used、ahub_daudio1_used、audiohdm_i_used、sndhdm_i_used 都设置为 1。

2.13.4 [snddaudio0]

配置项	配置项含义
snddaudio0_used	是否使用该接口, 默认配置为 0 1: 使用 0: 不使用

配置举例:

```
请看当前目录下的board.dts
snddaudio0:sound@0 {
    status = "okay";
};
```

2.13.5 [daudio0]

配置项	配置项含义
daudio0_used	是否使用 daudio0 接口, 默认要配置为 1 1: 使用 0: 不使用

配置项	配置项含义
pcm_lrck_period	每声道 bclk 个数/lrck 个数，设置如下：PCM mode: Number of BCLKs within(Left + Right)channel width 注意在 pcm 模式下，pcm_lrck_period 代表左和右声道相加，2 个声道的大小；I2S/Left-Justified/Right-Justified mode: Number of BCLKs within each individual channel width(Left or Right), pcm_lrck_period 代表左或者右声道，一个声道的大小；在 i2s 模式下，一个 lrck 的宽度：232。假如 fs=48k，那么需要的 bclk 是 3.072M=23248k; bclk_div = 24.576M/3.072M=8; 在 pcm 模式下，一个 lrck 的宽度就是 32。假如 fs=8k，那么需要的 bclk 是：328k=256k
pcm_lrckr_period	未使用
slot_width_select	数据 word 的宽度，对 i2s 模式，pcm 模式都有效。16bits/20bits/24bits/32bits
pcm_lsb_first	数据 endian，0: msb first; 1: lsb first
tx_data_mode	数据格式，0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law
rx_data_mode	数据格式，0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law
daudio_master	Master/slave 模式：1:daudio0 slave; 4:daudio0 master
audio_format	1 SND_SOC_DAIFMT_I2S(standard i2s format). use 表示标准 i2s 格式；2 SND_SOC_DAIFMT_RIGHT_J(right justified format). 表示右对齐格式；3 SND_SOC_DAIFMT_LEFT_J(left justified format) 表示左对齐格式；4 SND_SOC_DAIFMT_DSP_A 短帧模式并设置 frame_width 为 0. 短帧；5 SND_SOC_DAIFMT_DSP_B 长帧模式并设置 frame_width 为 1. 长帧；
signal_inversion	信号的翻转，比如标准的 I2S 模式，如果 lrck 翻转是模式，那么用示波器测量，左右声道是跟标准 i2s 模式相反的。如果 bclk 是翻转模式，那么用示波器测量，BCLK 信号是翻转的。1 SND_SOC_DAIFMT_NB_NF(normal bit clock + frame) use 表示 bclk 采用正常模式，lrck 也正常模式 2 SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM) 表示 bclk 采用正常模式，lrck 采用翻转模式 3 SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM) use 表示 bclk 采用翻转模式，lrck 采用正常模式 4 SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM) 表示 bclk 采用翻转模式，lrck 采用翻转模式
frametype	长帧或短帧 0: long frame = 2 clock width; 1: short frame
tdm_config	I2S 或 PCM 选择 0:pcm 1:i2s
mclk_div	时钟分频，默认 0x00

配置项	配置项含义
-----	-------

配置举例：

请看当前目录下的board.dts

```
ahub_daudio0:ahub_daudio0@0x05097000{
    /* for choose the corresponding pins */
    pinctrl-0 = <&ahub_daudio0_pins_c>;
    pinctrl-1 = <&ahub_daudio0_pins_d>;
    pinconfig = <0x01>;
    frametype = <0x00>;
    pcm_lrck_period = <0x20>;
    slot_width_select = <0x20>;
    daudio_master = <0x04>;
    aaudio_format = <0x01>;
    signal_inversion = <0x01>;
    tdm_config = <0x01>;
    mclk_div = <0x00>;
    status = "okay";
};
```

注意：要生成并注册 Daudio0 声卡，就必须要把 sndahub_used、ahub_daudio0_used、snddaudio0_used、daudio0_used 都设置为 1。

2.14 红外

2.14.1 [s_cir0]

配置项	配置项含义
s_cir0_used	是否使用该模块，1：使用 0：不使用
ir_protocol_used	红外协议选择，1：RC5 0：NEC
ir_addr_cnt	红外遥控器数量，最多 64 个
ir_power_key_code0	红外遥控器 powerkey 对应的按键值 0
ir_addr_code0	红外遥控器地址码 0
ir_power_key_code1	红外遥控器 powerkey 对应的按键值 1
ir_addr_code1	红外遥控器地址码 1

配置项	配置项含义
ir_power_key_code2	红外遥控器 powerkey 对应的按键值 2
ir_addr_code2	红外遥控器地址码 2
ir_power_key_code3	红外遥控器 powerkey 对应的按键值 3
ir_addr_code3	红外遥控器地址码 3
ir_power_key_code4	红外遥控器 powerkey 对应的按键值 4
ir_addr_code4	红外遥控器地址码 4
ir_power_key_code5	红外遥控器 powerkey 对应的按键值 5
ir_addr_code5	红外遥控器地址码 5
ir_power_key_code6	红外遥控器 powerkey 对应的按键值 6
ir_addr_code6	红外遥控器地址码 6
ir_power_key_code7	红外遥控器 powerkey 对应的按键值 7
ir_addr_code7	红外遥控器地址码 7
ir_power_key_code8	红外遥控器 powerkey 对应的按键值 8
ir_addr_code8	红外遥控器地址码 8
ir_power_key_code9	红外遥控器 powerkey 对应的按键值 9
ir_addr_code9	红外遥控器地址码 9
ir_power_key_code10	红外遥控器 powerkey 对应的按键值 10
ir_addr_code10	红外遥控器地址码 10
ir_power_key_code11	红外遥控器 powerkey 对应的按键值 11
ir_addr_code11	红外遥控器地址码 11
ir_power_key_code12	红外遥控器 powerkey 对应的按键值 12
ir_addr_code12	红外遥控器地址码 12
ir_power_key_code13	红外遥控器 powerkey 对应的按键值 13
ir_addr_code13	红外遥控器地址码 13
ir_power_key_code14	红外遥控器 powerkey 对应的按键值 14
ir_addr_code14	红外遥控器地址码 14
ir_power_key_code15	红外遥控器 powerkey 对应的按键值 15
ir_addr_code15	红外遥控器地址码 15
ir_power_key_code16	红外遥控器 powerkey 对应的按键值 16
ir_addr_code16	红外遥控器地址码 16
ir_power_key_code17	红外遥控器 powerkey 对应的按键值 17
ir_addr_code17	红外遥控器地址码 17
ir_power_key_code18	红外遥控器 powerkey 对应的按键值 18
ir_addr_code18	红外遥控器地址码 18

配置项	配置项含义
ir_power_key_code19	红外遥控器 powerkey 对应的按键值 19
ir_addr_code19	红外遥控器地址码 19
ir_power_key_code20	红外遥控器 powerkey 对应的按键值 20
ir_addr_code20	红外遥控器地址码 20
rc5_ir_power_key_code0	红外遥控器识别码
rc5_ir_addr_code0	红外遥控器具体的键值码

配置举例：

请看当前目录下的 board.dts

```
&s_cir0 {
    s_cir0_used = <1>;
    ir_power_key_code0 = <0x40>;
    ir_addr_code0 = <0xfe01>;
    ir_power_key_code1 = <0x1a>;
    ir_addr_code1 = <0xfb04>;
    ir_power_key_code2 = <0x57>;
    ir_addr_code2 = <0x00ff>;
    ir_power_key_code3 = <0x57>;
    ir_addr_code3 = <0xff00>;
    ir_power_key_code4 = <0x0b>;
    ir_addr_code4 = <0xf708>;
    ir_power_key_code5 = <0x03>;
    ir_addr_code5 = <0x00ef>;
    ir_power_key_code6 = <0xdc>;
    ir_addr_code6 = <0x4cb3>;
    ir_power_key_code7 = <0x0a>;
    ir_addr_code7 = <0x7748>;
    ir_power_key_code8 = <0x45>;
    ir_addr_code8 = <0xbd02>;
    ir_power_key_code9 = <0x4d>;
    ir_addr_code9 = <0xde21>;
    ir_power_key_code10 = <0x18>;
    ir_addr_code10 = <0xfe02>;
    ir_power_key_code11 = <0x18>;
    ir_addr_code11 = <0xff00>;
    ir_power_key_code12 = <0x4d>;
    ir_addr_code12 = <0xff40>;
    ir_power_key_code13 = <0x88>;
    ir_addr_code13 = <0xdd22>;
    ir_power_key_code14 = <0x0d>;
    ir_addr_code14 = <0xbc00>;
    ir_power_key_code15 = <0x0d>;
```



```

ir_addr_code15 = <0xfc00>;
ir_power_key_code16 = <0x15>;
ir_addr_code16 = <0x7f80>;
ir_power_key_code17 = <0x4d>;
ir_addr_code17 = <0x4040>;
wakeup-source;
};
    
```

2.15 动态切换打印

2.15.1 [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
```

2.16 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。

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 acknowledgment 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 datasheet 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.