全志科技
Allwinner Technology

A83T

System Configuration 说明书

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2014-10-20		建立初始版本

目录

1. 系统

1.1. [product]

配置项	配置项含义
Version = "100"	配置的版本号
machine = "f1"	方案名字

配置举例：

[product]

version = "100"

machine = "f1"

1.2. [platform]

配置项	配置项含义
eraseflag=1	量产时是否擦除。0：不擦，1：擦除（仅仅对量产工具，升级工具无效）
next_work=3	PhoenixUSBPro 烧写完后样机的行为. 0: 不做任何处理, 2: 重启, 3: 关机, 4: 重新烧写

配置举例：

;--

; eraseflag - 1 erase data partition, 0 - do not erase data partition

; next_work - action after burn, 0x0 by config, 0x1 normal, 0x2 reboot, 0x3 shutdown,0x4 reupdate

;--

[platform]

eraseflag = 0

next_work=3

1.3. [target]

配置项	配置项含义
boot_clock=xx	启动频率; xx 表示多少 MHZ
storage_type = -1	启动介质选择 0 ： nand, 1：card0,2：card2,-1（defualt）自动扫描启动介质:

配置举例：

[target]

boot_clock = 1008

storage_type = -1

1.4. [charging_type]

配置项	配置项含义

charging_type = 1	为 1，才能进入安卓关机充电模式；为 0 是进入 boot standby 进行充电。

注：如果想进入安卓关机充电功能，应该务必加上以上内容。

1.5. [key_detect_en]

配置项	配置项含义
keyen_flag = 1	当 keyen_flag = 1 时，支持按键检测；当 keyen_flag = 0 时，不支持按键检测

配置举例：

[key_detect_en]

keyen_flag = 1

1.6. [power_sply]

配置项	配置项含义
dcdc1_vol = 1003000	dcdc1 的输出电压，3000mV
dcdc2_vol = 1000900	dcdc2 的输出电压，900mV
dcdc3_vol = 1000900	dcdc3 的输出电压，900mV
dcdc4_vol = 1000900	dcdc4 的输出电压，900mV
dcdc5_vol = 1001200	dcdc5)的输出电压，1200mV
dcdc6_vol = 1000900	dcdc5)的输出电压，900mV
aldo1_vol = 1800	aldo2 的输出电压，1200mV
aldo2_vol = 1001800	aldo2 的输出电压，1800mV
aldo3_vol = 1003000	aldo3 的输出电压，3000mV
eldo2_vol = 1800	aldo3 的输出电压，1800mV
dldo2_vol = 4200	aldo3 的输出电压，4200mV
dldo4_vol = 2800	aldo3 的输出电压，2800mV
fldo2_vol = 1000900	aldo3 的输出电压，900mV
dldo1_vol = 2800	aldo3 的输出电压，2800mV

配置说明:

电压名称 = 100XXXX ：表示把该路电压设置为 XXXX 指定的电压值，同时打开输出开关

电压名称 = 000XXXX ：表示把该路电压设置为 XXXX 指定的电压值，同时关闭输出开关，当有需要时由内核驱动打开

电压名称 = 0 ：表示关闭该路电压输出开关，不修改原有的值

配置举例：

```
;--------------------------------------------------------------------------------
;
;  各路电压输出语法说明：
;
;  电压名称  = 100XXXX  ：表示把该路电压设置为 XXXX 指定的电压值，同时打开输出开关
;  电压名称  = 000XXXX  ：表示把该路电压设置为 XXXX 指定的电压值，同时关闭输出开关，当有需要时由内核驱动打开
```

```
;   电压名称    = 0            ：表示关闭该路电压输出开关，不修改原有的值
;
;--------------------------------------------------------------------------------
[power_sply]
dcdc1_vol                       = 1003000
dcdc2_vol                       = 1000900
dcdc3_vol                       = 1000900
dcdc4_vol                       = 1000900
dcdc5_vol                       = 1001200
dcdc6_vol                       = 1000900
aldo1_vol                       = 1800
aldo2_vol                       = 1001800
aldo3_vol                       = 1003000
eldo2_vol                       = 1800
dldo2_vol                       = 4200
dldo4_vol                       = 2800
fldo2_vol                       = 1000900
dldo1_vol                       = 2800
```

1.7. [gpio_bias]

配置项	配置项含义
pb_bias ="axp81x:dcdc1:3000"	PB 的 bias 电压为 axp81x 的 dcdc1，输出为 3000mV
pc_bias="axp81x:dcdc1:3000"	PC 的 bias 电压为 axp81x 的 dcdc1，输出为 3000mV
pd_bias="axp81x:aldo1:1800"	PD 的 bias 电压为 axp81x 的 aldo1，输出为 1800mV
pe_bias="axp81x:dcdc1:3000"	PE 的 bias 电压为 axp81x 的 dcdc1，输出为 3000mV
pf_bias="axp81x:dcdc1:3000"	PF 的 bias 电压为 axp81x 的 dcdc1，输出为 3000mV
pg_bias="axp81x:dcdc1:3000"	PG 的 bias 电压为 axp81x 的 dcdc1，输出为 3000mV
ph_bias="axp81x:dcdc1:3000"	PH 的 bias 电压为 axp81x 的 dcdc1，输出为 3000mV
pl_bias="axp81x:aldo3:3000"	PL 的 bias 电压为 axp81x 的 aldo3，输出为 3000mV

注意：调整 GPIO 电压时，必须保证 bias 电压和调整后的 GPIO 电压匹配，不然重则烧坏物理器件，轻则 GPIO 驱动能力不足导致设备无法使用或不稳定等问题.

配置举例：

```
;-------------------------------------------
; normal config: eg. px_bias = "pmu_name:supply_name:voltage"
;                       pmu_name = axp809, axp806
;                       supply_name = dcdc1, dcdc2, aldo1, gpio1ldo, etc...
;
; special config: eg. px_bias = "floating"
;                       when the gpio is float, use it
;                       and the register will ignore it(use default value)
;
```

```
; special config: eg. px_bias = "constant:voltage"
;                       when the gpio connect a constant voltage, use it
;                       and the register will set a matching value
;-------------------------------------------
[gpio_bias]
;pa_bias          = "axp81x:dcdc1:3000"
pb_bias           = "axp81x:dcdc1:3000"
pc_bias           = "axp81x:dcdc1:3000"
pd_bias           = "axp81x:aldo1:1800"
pe_bias           = "axp81x:dcdc1:3000"
pf_bias           = "axp81x:dcdc1:3000"
pg_bias           = "axp81x:dcdc1:3000"
ph_bias           = "axp81x:dcdc1:3000"
pl_bias           = "axp81x:aldo3:3000"
```

1.8. [card_boot]

配置项	配置项含义
logical_start = 40960	启动卡逻辑起始扇区
sprite_gpio0 =	卡量产 gpio led 灯配置
next_work = 2	1-不做任何动作，2-重启，3-关机，4-量产，5-正常启动

举例配置：

```
[card_boot]
logical_start     = 40960
sprite_gpio0      =
next_work         = 2
```

1.9. [card_boot0_para]

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

配置举例：

```
[card0_boot_para]
```

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

1.10. [card_boot2_para]

配置项	配置项含义
card_ctrl=2	卡启动控制器选择 2
card_high_speed=xx	速度模式 0 为低速，1 为高速
card_line=8	8 线卡
sdc_ cmd =xx	sdc 命令信号的 GPIO 配置
sdc_ clk =xx	sdc 卡时钟信号的 GPIO 配置
sdc_ d0 =xx	sdc 卡数据 0 线信号的 GPIO 配置
sdc_ d1 =xx	sdc 卡数据 1 线信号的 GPIO 配置
sdc_d2=xx	sdc 卡数据 2 线信号的 GPIO 配置
sdc_d3=xx	sdc 卡数据 3 线信号的 GPIO 配置
sdc_d4=xx	sdc 卡数据 4 线信号的 GPIO 配置
sdc_d5=xx	sdc 卡数据 5 线信号的 GPIO 配置
sdc_d6=xx	sdc 卡数据 6 线信号的 GPIO 配置
sdc_d7=xx	sdc 卡数据 7 线信号的 GPIO 配置
sdc_2xmode=1	设置内部控制器的采样模式，默认配置为 1
sdc_ddrmode=1	是否使用 ddr 模式，默认配置为 1

配置举例：
[card2_boot_para]
card_ctrl = 2
card_high_speed = 1
card_line = 8
sdc_cmd = port:PC06<3><1><2><default>
sdc_clk = port:PC05<3><1><2><default>
sdc_d0 = port:PC08<3><1><2><default>
sdc_d1 = port:PC09<3><1><2><default>
sdc_d2 = port:PC10<3><1><2><default>
sdc_d3 = port:PC11<3><1><2><default>
sdc_d4 = port:PC12<3><1><2><default>
sdc_d5 = port:PC13<3><1><2><default>
sdc_d6 = port:PC14<3><1><2><default>
sdc_d7 = port:PC15<3><1><2><default>

sdc_2xmode = 1
sdc_ddrmode = 1

1.11. [twi_para]

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

配置举例：

```
[twi_para]
twi_port = 0
twi_scl = port:PH02<2><default><default><default>
twi_sda = port:PH03<2><default><default><default>
```

1.12. [uart_para]

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

配置举例：

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

1.13. [force_uart_para]

配置项	配置项含义
force_uart_port = 0	kernel 打印使用的 uart 控制器号. 0~4
force_uart_tx=xx	串口发送的 GPIO 配置
force_uart_rx=xx	串口接收的 GPIO 配置

此项用于配置 kernel 使用哪路 uart 控制器来打印.
发布的固件都是不带卡打印的，打印口被 SD 卡占用，可以通过本项来配置 kernel 用其他的 uart 口打印.

配置举例：

```
[force_uart_para]
force_uart_port = 0
force_uart_tx = port:PF02<3><1><default><default>
force_uart_rx = port:PF04<3><1><default><default>
```

1.14. [jtag_para]

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

配置举例：

[jtag_para]

jtag_enable = 1

jtag_ms = port:PF00<3><default><default><default>

jtag_ck = port:PF05<3><default><default><default>

jtag_do = port:PF03<3><default><default><default>

jtag_di = port:PF01<3><default><default><default>

1.15. [clock]

配置项	配置项含义
pll_ve = 432	VE 时钟频率
;pll_periph = 600	PERIPH 时钟默认 600M，不要修改.
pll_gpu = 432	GPU 时钟频率
pll_hsic = 480	HSIC 时钟频率
pll_de = 504	DE 时钟频率
apb2_parent = pll_periph	设置 apb2 的时钟源为 pll_periph
apb2 = 40	apb2 总线时钟频率

配置举例：

[clock]

pll_ve = 432

;pll_periph = 600

pll_gpu = 432

pll_hsic = 480

pll_de = 504

apb2_parent = pll_periph

apb2 = 40

1.16. [pm_para]

配置项	配置项含义
standby_mode = x	if 1 == standby_mode, then support super standby; else, support normal standby.

配置举例：

```
;-------------------------------------------------------------------------------------------------------
; if 1 == standby_mode, then support super standby;
; else, support normal standby.
;-------------------------------------------------------------------------------------------------------
[pm_para]
standby_mode        = 1
```

2. SDRAM

2.1. [dram_para]

配置项	配置项含义
dram_clk =xx	DRAM 的时钟频率，单位为 MHz;它为 24 的整数倍，最低不得低于 120，
dram_type =xx	DRAM 类型： 2 为 DDR2 3 为 DDR3
dram_zq=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_odt_en=xx	ODT 是否需要使能 0： 不使能 1： 使能 一般情况下，为了省电，此项为 0
dram_para1=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_para2 =xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_mr0 =xx	DRAM CAS 值，可为 6,7,8,9；具体需根据 DRAM 的规格书和速度来确定
dram_mr1 =xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_mr2 =xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_mr3 =xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr0=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr1=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr2=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr3=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr4=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr5=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr6=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr7=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr8=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr9=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr10=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr11=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改

dram_tpr12=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改
dram_tpr13=xx	DRAM 控制器内部参数，由原厂来进行调节，请勿修改

配置举例：

```
;*****************************************************************************
;sdram configuration
;dram_para2 = 0x00001200   ;代表启用 dram 双通道
;dram_para2 = 0x00001100   ;代表启用 dram 单通道
;*****************************************************************************
[dram_para]
dram_clk        = 732
dram_type       = 7
dram_zq          = 0x3bfb
dram_odt_en     = 1
dram_para1      = 0x10E40000
dram_para2      = 0x0000
dram_mr0         = 0x1840
dram_mr1         = 0x40
dram_mr2         = 0x8
dram_mr3         = 0x2
dram_tpr0       = 0x0048A192
dram_tpr1       = 0x01B1B18d
dram_tpr2       = 0x00076052
dram_tpr3       = 0
dram_tpr4       = 0
dram_tpr5       = 0
dram_tpr6       = 0
dram_tpr7       = 0
dram_tpr8       = 1
dram_tpr9       = 0
dram_tpr10      = 0
dram_tpr11      = 0x5005
dram_tpr12      = 0
dram_tpr13      = 0xC00
```

3. GMAC

3.1. [gmac0]

平板方案未使用 gmac，暂不处理.

配置项	配置项含义

gmac_used = 0	
gmac_txd0 =	
gmac_txd0 =	
gmac_txd0 =	
gmac_txd0 =	
gmac_txd0 =	
gmac_txd0 =	
gmac_txd0 =	
gmac_txclk =	
gmac_txen =	
gmac_gtxclk =	
gmac_rxd0 =	
gmac_rxd0 =	
gmac_rxd0 =	
gmac_rxd0 =	
gmac_rxd0 =	
gmac_rxd0 =	
gmac_rxd0 =	
gmac_rxdv =	
gmac_rxclk =	
gmac_txerr =	
gmac_rxerr =	
gmac_col =	
gmac_crs =	
gmac_clkin =	
gmac_mdc =	
gmac_mdio =	

配置举例：

[gmac0]

gmac_used = 0

gmac_txd0 =

gmac_txd1 =

gmac_txd2 =

gmac_txd3 =

gmac_txd4 =

gmac_txd5 =

gmac_txd6 =

gmac_txd7 =

gmac_txclk =

gmac_txen =

gmac_gtxclk =

gmac_rxd0 =

gmac_rxd1 =
gmac_rxd2 =
gmac_rxd3 =
gmac_rxd4 =
gmac_rxd5 =
gmac_rxd6 =
gmac_rxd7 =
gmac_rxdv =
gmac_rxclk =
gmac_txerr =
gmac_rxerr =
gmac_col =
gmac_crs =
gmac_clkin =
gmac_mdc =
gmac_mdio =

4. STANDBY

4.1. [wakeup_src_para]

配置项	配置项含义
wakeup_src0	
wakeup_src_wl	wifi 唤醒源的 pin 设置.
wakeup_src_bt	bluetooth 唤醒源的 pin 设置.
bb_wake_ap	3g 唤醒源的 pin 设置.

此项用于配置 GPIO 唤醒源.

配置举例：

```
;-------------------------------------------------------------------------------
; wakeup_src_para:
;     sometimes, u would like to add more wakeup src in standby mode, these para will be help;
;     u need to make sure the standby mode support the wakeup src. Also, some hw condition must be guaranteed.
;     see: scene_lock for more help info;
;     wakeup_src: to make the scenario work, the wakeup src is needed.
;-------------------------------------------------------------------------------
[wakeup_src_para]
wakeup_src0 =
wakeup_src_wl = port:PL04<6><default><default><0>
wakeup_src_bt = port:PL05<6><default><default><0>
bb_wake_ap =
```

关于 GPIO 唤醒源配置的说明:

1. 默认情况下，对于唤醒源为 PL, PM, AXP_PORT 的情况，以上配置是可直接使用。
2. 对于其他 io 口，若要求能唤醒系统，需要通过 scene_manager 启用 extended_standby, 配置供电依赖关系, 从而支持唤醒源的检测；
3. 本配置中，对于 wakeup_src0 这一子键，命名上没有限制，可根据需要配置为脚本解析系统支持的任意名字。
4. 本配置中，关心这一唤醒源的模块，需自行处理该唤醒源对应的 io 口的配置。Pm 管理模块只是在进入 standby 时，enable 中断，无多余操作。
5. 本配置在 pm 模块初始化后生效，支持使用 extended_standby 的接口，进行动态 disable/enable；

4.2. [sys_pwr_dm_para]

根据 soc 的特性，开关后会影响系统稳定性的 power_domain 交由系统管理，属于 sys_pwr_dm. 在 soc 整合完毕后，sys_pwr_dm 即固定了。归为 sys_pwr_dm 的供电，会由系统进行管理，进行掉电或开电操作；

若出于方案的差异化需求，不希望系统对特定的 power_domain 进行操作，需将特定的 power_domain 从 sys_pwr_dm 拿掉；

以下范例：将 vcc-io 的供电管理，从 arisc 拿掉，从而，进入 suspend 状态时，系统不会关闭 vcc-io 对应的 regulator: dcdc1;

支持哪些 pwr_domain 属性的更改，需参考 pmu_regu 的描述，或咨询系统研发人员；

```
;-------------------------------------------------------------------------------
;sys_pwr_dm_para
;this para is used to change default sys_pwr_dm config when necessary.
;     allowed sys_pwr_dm is such as follow:
;                     vdd-cpua
;                     vdd-cpub
;                     vdd-gpu
;                     vcc-dram
;                     vdd-sys
;                     which is compatible with pmu regu config. see: [pmu1_regu] for more info.
;     value: 0: del the pwr_dm from sys_pwr_dm_mask;
;               1: add the pwr_dm into sys_pwr_dm_mask;
;-------------------------------------------------------------------------------
[sys_pwr_dm_para]
vcc-io = 0
```

4.3. [dynamic_standby_para]

配置项	配置项含义
enable=x	0: 下面的所有配置项不起作用; 1: 下面的所有配置项起作用
dram_selfresh_flag=x	0: 待机时 dram 进自刷新

	1: 待机时 dram 不进自刷新

配置举例:

```
;-------------------------------------------------------------------------------
;dynamic_standby_para
;     enable:
;      value: 0: all config is ignored.
;                  1: all config is effective.
;     dram_selfresh_flag:
;      value: 0: dram will not enter selfresh,
;                    this config is used for stop dram entering selfresh, in case of dram memory have bug.
;                  1: dram will enter slefresh.
;
;-------------------------------------------------------------------------------
[dynamic_standby_para]
enable = 0
dram_selfresh_flag = 1
```

5. I2C 总线

主控有 3 个 I2C（twi）控制器

5.1. [twi0]

配置项	配置项含义
twi_used =xx	TWI 使用控制：1 使用，0 不用
twi_scl =xx	TWI SCK 的 GPIO 配置
twi_sda=xx	TWI SDA 的 GPIO 配置

配置举例：

```
[twi0]
twi_used            = 1
twi_scl             = port:PH0<2><default><default><default>
twi_sda             = port:PH1<2><default><default><default>
```

5.2. [twi1]

配置项	配置项含义
twi_used =xx	TWI 使用控制：1 使用，0 不用
twi_scl =xx	TWI SCK 的 GPIO 配置
twi_sda=xx	TWI SDA 的 GPIO 配置

配置举例：

[twi1]
twi_used = 1
twi_scl = port:PH2<2><default><default><default>
twi_sda = port:PH3<2><default><default><default>

5.3. [twi2]

配置项	配置项含义
twi_used =xx	TWI 使用控制：1 使用，0 不用
twi_scl =xx	TWI SCK 的 GPIO 配置
twi_sda=xx	TWI SDA 的 GPIO 配置

配置举例：
[twi2]
twi_used = 1
twi_scl = port:PH4<2><default><default><default>
twi_sda = port:PH5<2><default><default><default>

6. 串口(UART)

主控有 5 路 uart 接口，5 路支持 4 线或者 2 线通讯（但十分不建议用 uart0 作为控制台以外的用途），实例中，有些路仅仅写出 2 路的配置形式，但实际使用时只要将其按照 4 路的格式补全，也能支持 4 线通讯

6.1. [uart0]

配置项	配置项含义
uart_used =xx	UART 使用控制：1 使用，0 不用
uart_port =xx	UART 端口号
uart_type = xx	UART 类型
uart_tx =xx	UART TX 的 GPIO 配置
uart_rx=xx	UART RX 的 GPIO 配置
uart_regulator = xx	UART 的供电选择

配置举例：
[uart0]
uart_used = 1
uart_port = 0
uart_type = 2
uart_tx = port:PB09<2><1><default><default>
uart_rx = port:PB10<2><1><default><default>
uart_regulator = "vcc-io"

6.2. [uart1]

配置项	配置项含义
uart_used	UART 使用控制：1 使用，0 不用
uart_port	UART 端口号
uart_type	UART 类型
uart_tx	UART TX 的 GPIO 配置
uart_rx	UART RX 的 GPIO 配置
uart_rts	UART RTS 的 GPIO 配置
uart_cts	UART CTS 的 GPIO 配置
uart_regulator = xx	UART 的供电选择

配置举例：

```
[uart1]
uart_used           = 1
uart_port           = 1
uart_type           = 4
uart_tx             = port:PG06<2><1><default><default>
uart_rx             = port:PG07<2><1><default><default>
uart_rts            = port:PG08<2><1><default><default>
uart_cts            = port:PG09<2><1><default><default>
uart_regulator  = "vcc-io"
```

6.3. [uart2]

配置项	配置项含义
uart_used =xx	UART 使用控制：1 使用，0 不用
uart_port =xx	UART 端口号
uart_type =xx	UART 类型
uart_tx =xx	UART TX 的 GPIO 配置
uart_rx＝xx	UART RX 的 GPIO 配置
uart_rts=xx	UART RTS 的 GPIO 配置
uart_cts=xx	UART CTS 的 GPIO 配置
uart_regulator = xx	UART 的供电选择

配置举例：

```
[uart2]
uart_used           = 1
uart_port           = 2
uart_type           = 4
uart_tx             = port:PB00<2><1><default><default>
uart_rx             = port:PB01<2><1><default><default>
uart_rts            = port:PB02<2><1><default><default>
uart_cts            = port:PB03<2><1><default><default>
```

uart_regulator = "vcc-io"

6.4. [uart3]

配置项	配置项含义
uart_used =xx	UART 使用控制：1 使用，0 不用
uart_port =xx	UART 端口号
uart_type =xx	UART 类型
uart_tx =xx	UART TX 的 GPIO 配置
uart_rx＝xx	UART RX 的 GPIO 配置
uart_rts=xx	UART RTS 的 GPIO 配置
uart_cts=xx	UART CTS 的 GPIO 配置
uart_regulator = xx	UART 的供电选择

配置举例：

[uart3]
uart_used = 0
uart_port = 3
uart_type = 4
uart_tx = port:PG10<3><1><default><default>
uart_rx = port:PG11<3><1><default><default>
uart_rts = port:PG12<3><1><default><default>
uart_cts = port:PG13<3><1><default><default>
uart_regulator = "vcc-io"

6.5. [uart4]

配置项	配置项含义
uart_used =xx	UART 使用控制：1 使用，0 不用
uart_port =xx	UART 端口号
uart_type =xx	UART 类型
uart4_tx =xx	UART TX 的 GPIO 配置
uart4_rx＝xx	UART RX 的 GPIO 配置
uart_cts=xx	UART CTS 的 GPIO 配置
uart_regulator = xx	UART 的供电选择

配置举例：

[uart4]
uart_used = 0
uart_type = 4
uart_tx = port:PE10<3><1><default><default>
uart_rx = port:PE11<3><1><default><default>
uart_rts = port:PE12<3><1><default><default>
uart_cts = port:PE13<3><1><default><default>

uart_regulator = "vcc-io"

7. SPI 总线

7.1. [spi0]

配置项	配置项含义
spi_used =xx	SPI 使用控制：1 使用，0 不用
spi_cs_bitmap =xx	由于 SPI 控制器支持多个 CS，这一个参数表示 CS 的掩码；
spi_cs0 =xx	SPI CS0 的 GPIO 配置
spi_sclk =xx	SPI CLK 的 GPIO 配置
spi_mosi＝xx	SPI MOSI 的 GPIO 配置
spi_miso=xx	SPI MISO 的 GPIO 配置
spi_regulator = xx	SPI 的供电选择

配置举例：

[spi0]
spi_used = 0
spi_cs_bitmap = 1
spi_mosi = port:PC00<3><default><default><default>
spi_miso = port:PC01<3><default><default><default>
spi_sclk = port:PC02<3><default><default><default>
spi_cs0 = port:PC03<3><1><default><default>
spi_regulator = "vcc-io"

7.2. [spi1]

配置项	配置项含义
spi_used =xx	SPI 使用控制：1 使用，0 不用
spi_cs_bitmap =xx	由于 SPI 控制器支持多个 CS，这一个参数表示 CS 的掩码；
spi_cs0 =xx	SPI CS0 的 GPIO 配置
spi_sclk =xx	SPI CLK 的 GPIO 配置
spi_mosi＝xx	SPI MOSI 的 GPIO 配置
spi_miso=xx	SPI MISO 的 GPIO 配置
spi_regulator = xx	SPI 的供电选择

配置举例：

[spi1]
spi_used = 0

spi_cs_bitmap = 1
spi_cs0 = port:PG06<3><1><default><default>
spi_sclk = port:PG07<3><default><default><default>
spi_mosi = port:PG08<3><default><default><default>
spi_miso = port:PG09<3><default><default><default>
spi_regulator = "vcc-io"

7.3. [spi_devices]

配置项	配置项含义
spi_dev_num=xx	该项目直接和下面的[spi_board0]相关，它指定主板连接 spi 设备的数目，假如有 N 个 SPI 设备那么[spi_devices]中就要有 N 个（[spi_board0]到[spi_board（N-1）]）配置

配置举例：
[spi_devices]
spi_dev_num = 1

7.4. [spi_board0]

配置项	配置项含义
modalias=xx	Spi 设备名字，
max_speed_hz =xx	最大传输速度（HZ）
bus_num =xx	Spi 设备控制器序号
chip_select=xx	理论上可以选 0，1，2，3，目前只支持 1，2（芯片没引出接口）
mode＝xx	SPI MOSI 的 GPIO 配置可选值 0-3

配置举例：
[spi_board0]
modalias = "m25p32"
max_speed_hz = 33000000
bus_num = 0
chip_select = 0
mode = 0

8. 触摸屏

8.1. [rtp_para]

电阻屏的配置参数，现在一般不用.

配置举例:

```
[rtp_para]
rtp_used          = 0
rtp_screen_size = 5
rtp_regidity_level = 5
rtp_press_threshold_enable = 0
rtp_press_threshold = 0x1f40
rtp_sensitive_level = 0xf
rtp_exchange_x_y_flag = 0
```

8.2. [ctp_para]

电容屏配置参数.

配置项	配置项含义
ctp_used	该选项为是否开启电容触摸，支持的话置 1，反之置 0
ctp_name	tp 的 name，必须配，与驱动保持一致
ctp_twi_id	用于选择 i2c adapter，可选 1，2
ctp_twi_addr	指明 i2c 设备地址，与具体硬件相关
ctp_screen_max_x	触摸板的 x 轴最大坐标
ctp_screen_max_y	触摸板的 y 轴最大坐标
ctp_revert_x_flag	是否需要翻转 x 坐标，需要则置 1，反之置 0
ctp_revert_y_flag	是否需要翻转 y 坐标，需要则置 1，反之置 0
ctp_exchange_x_y_flag	是否需要 x 轴 y 轴坐标对换
ctp_int_port	电容屏中断信号的 GPIO 配置
ctp_wakeup	电容屏唤醒信号的 GPIO 配置
ctp_power_ldo	电容屏供电 ldo
ctp_power_ldo_vol	电容屏供电 ldo 电压
ctp_power_io	电容屏供电 gpio
ctp_gesture_wakeup	是否支持手势唤醒. 0: 不支持, 1: 支持.

配置举例：

```
[ctp_para]
ctp_used                 = 1
ctp_gesture_wakeup   = 0
ctp_twi_id               = 0
ctp_twi_addr             = 0x5d
ctp_name                  = "gt9271_mb976a9"
ctp_screen_max_x      = 1200
ctp_screen_max_y      = 1920
ctp_revert_x_flag     = 0
ctp_revert_y_flag     = 0
ctp_exchange_x_y_flag = 1
```

```
ctp_int_port            = port:PH07<6><default><default><default>
ctp_wakeup               = port:PH06<1><default><default><1>
ctp_power_ldo           = "vcc-ctp"
ctp_power_ldo_vol      = 3000
ctp_power_io             =
```

8.3. [ctp_list_para]

触摸屏自动扫描配置.

配置项	配置项含义
ctp_det_used	0 or 1，是否使用自动检测功能
ft5x_ts	0 or 1，检测时是否扫描此类触屏
gt82x	0 or 1，检测时是否扫描此类触屏
gslX680	0 or 1，检测时是否扫描此类触屏
gt9xx_ts	0 or 1，检测时是否扫描此类触屏
gt811	0 or 1，检测时是否扫描此类触屏
zet622x	0 or 1，检测时是否扫描此类触屏
aw5306_ts	0 or 1，检测时是否扫描此类触屏

配置举例：

```
[ctp_list_para]
ctp_det_used                = 1
ft5x_ts                      = 1
gt82x                        = 1
gslX680                     = 1
gt9xx_ts                    = 0
gt9xxnew_ts               = 1
gt811                        = 1
zet622x                    = 1
aw5306_ts                  = 1
```

8.4. [tkey_para]

TP 按键配置，对于支持 TP 按键的触摸屏才需要.

配置项	配置项含义
tkey_used=xx	是否启支持 TP 按键, 0: 不支持, 1: 支持. A83T 不支持
tkey_twi_id=xx	TP 按键使用的 twi 通道
tkey_twi_addr=xx	TP 按键对应的 twi 设备地址
tkey_int=xx	TP 按键中断号

配置举例:

[tkey_para]
tkey_used = 0
tkey_twi_id =
tkey_twi_addr =
tkey_int =

9. 温度

9.1. [ths_para]

配置项	配置项含义
ths_used = 1	总开关，为 0 thermal 功能关闭
ths_trend = 0	Thermal 趋势判断功能，目前关闭
ths_trip_count = 6	Trip_point 数目（注意必须和下面的 ths_trip1_*一致）
ths_trip1_0 = 50 ths_trip1_1 = 60 ths_trip1_2 = 70 ths_trip1_3 = 85 ths_trip1_4 = 95 ths_trip1_5 = 105	对应的 trip_point 温度 注意，最后一个强制作为关机使用
ths_trip1_0_min = 0 ths_trip1_0_max = 1 ths_trip1_1_min = 1 ths_trip1_1_max = 2 ths_trip1_2_min = 2 ths_trip1_2_max = 3 ths_trip1_3_min = 3 ths_trip1_3_max = 5 ths_trip1_4_min = 5 ths_trip1_4_max = 7 ths_trip1_5_min = 7 ths_trip1_5_max = 7	和上面的表对应，注意 1） 0 和 3 对应的 cooler_count=4 (下节描述)的对应关系 2) ths_trip1_2 是强制作为关机用的 3) 注意必须 trip1_0_max == trip1_1_min， 后面以此类推

配置举例：
[ths_para]
ths_used = 1
ths_trend = 0
ths_trip1_count = 6
ths_trip1_0 = 50
ths_trip1_1 = 60
ths_trip1_2 = 70

ths_trip1_3 = 85
ths_trip1_4 = 95
ths_trip1_5 = 105
ths_trip1_6 = 0
ths_trip1_7 = 0
ths_trip1_0_min = 0
ths_trip1_0_max = 1
ths_trip1_1_min = 1
ths_trip1_1_max = 2
ths_trip1_2_min = 2
ths_trip1_2_max = 3
ths_trip1_3_min = 3
ths_trip1_3_max = 5
ths_trip1_4_min = 5
ths_trip1_4_max = 7
ths_trip1_5_min = 7
ths_trip1_5_max = 7
ths_trip1_6_min = 0
ths_trip1_6_max = 0
ths_trip2_count = 1
ths_trip2_0 = 105

9.2. [cooler_table]

cpu cooler 等级的设置. 由当前温度来决定 cpu 运行在哪个等级.

配置项	配置项含义
cooler_count = 8	Cooler 支持的 state 数目，每个代表一档
cooler0 = "2016000 4 2016000 4" cooler1 = "1800000 4 1800000 4" cooler2 = "1608000 4 1608000 4" cooler3 = "1200000 4 1200000 4" cooler4 = "1200000 4 1200000 2" cooler5 = "1200000 4 4294967295 0" cooler6 = "1200000 2 4294967295 0 1" cooler7 = "1200000 1 4294967295 0 1"	每个 cooler 代表一档，分别表示 允许 cpu 允许的最大频率，允许 cpu 的最大数目，保留做其它用途，保留做其它用途 4294967295 用来表示 0xffffffff，用来区分正常频率; 最后两行末尾的 1 表示对 GPU 进行频率限制.

配置举例：

[cooler_table]
cooler_count = 8
cooler0 = "2016000 4 2016000 4"
cooler1 = "1800000 4 1800000 4"
cooler2 = "1608000 4 1608000 4"
cooler3 = "1200000 4 1200000 4"
cooler4 = "1200000 4 1200000 2"

cooler5 = "1200000 4 4294967295 0"
cooler6 = "1200000 2 4294967295 0 1"
cooler7 = "1200000 1 4294967295 0 1"

10. 闪存

10.1. [nand0_para]

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

配置举例：

```
[nand0_para]
nand_support_2ch       = 0

nand0_used              = 0
nand0_we                = port:PC00<2><default><default><default>
nand0_ale               = port:PC01<2><default><default><default>
nand0_cle               = port:PC02<2><default><default><default>
nand0_ce1               = port:PC03<2><default><default><default>
nand0_ce0               = port:PC04<2><default><default><default>
nand0_nre               = port:PC05<2><default><default><default>
```

nand0_rb0 = port:PC06<2><default><default><default>
nand0_rb1 = port:PC07<2><default><default><default>
nand0_d0 = port:PC08<2><default><default><default>
nand0_d1 = port:PC09<2><default><default><default>
nand0_d2 = port:PC10<2><default><default><default>
nand0_d3 = port:PC11<2><default><default><default>
nand0_d4 = port:PC12<2><default><default><default>
nand0_d5 = port:PC13<2><default><default><default>
nand0_d6 = port:PC14<2><default><default><default>
nand0_d7 = port:PC15<2><default><default><default>
nand0_ce2 = port:PC17<2><default><default><default>
nand0_ce3 = port:PC18<2><default><default><default>
nand0_ndqs = port:PC16<2><default><default><default>

11. 显示

11.1. [disp_init]

配置项	配置项含义
disp_init_enable=xx	是否进行显示的初始化设置
disp_mode =xx	显示模式： 0:screen0<screen0,fb0>
screen0_output_type=xx	屏 0 输出类型(0:none; 1:lcd; 2:tv; 3:hdmi; 4:vga)
screen0_output_mode =xx	屏 0 输出模式(used for tv/hdmi output, 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＝xx	屏 1 输出类型(0:none; 1:lcd; 2:tv; 3:hdmi; 4:vga)
screen1_output_mode=xx	屏 1 输出模式(used for tv/hdmi output, 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=xx	fb0 的格式(4:RGB655 5:RGB565 6:RGB556 7:ARGB1555 8:RGBA5551 9:RGB888 10:ARGB8888 12:ARGB4444)
fb0_pixel_sequence=xx	fb0 的 pixel sequence(0:ARGB 1:BGRA 2:ABGR 3:RGBA)
fb0_scaler_mode_enable=xx	fb0 是否使用 scaler mode，即使用 FE
fb0_width=xx	fb0 的宽度,为 0 时将按照输出设备的分辨率
fb0_height=xx	fb0 的高度，为 0 时将按照输出设备的分辨率
fb1_format=xx	fb1 的格式(4:RGB655 5:RGB565 6:RGB556 7:ARGB1555 8:RGBA5551 9:RGB888 10:ARGB8888 12:ARGB4444)
fb1_pixel_sequence=xx	fb1 的 pixel sequence(0:ARGB 1:BGRA 2:ABGR 3:RGBA)
fb1_scaler_mode_enable=xx	fb1 是否使用 scaler mode，即使用 FE
fb1_width=xx	Fb1 的宽度,为 0 时将按照输出设备的分辨率
fb1_height=xx	Fb1 的高度，为 0 时将按照输出设备的分辨率

lcd0_backlight	Lcd0 的背光初始值，0~255
lcd1_backlight	Lcd1 的背光初始值，0~255

配置举例：

```
[disp_init]
disp_init_enable            = 1
disp_mode                   = 0

screen0_output_type         = 1
screen0_output_mode         = 4

screen1_output_type         = 3
screen1_output_mode         = 4

fb0_format                  = 0
fb0_width                   = 0
fb0_height                  = 0

fb1_format                  = 0
fb1_width                   = 0
fb1_height                  = 0

lcd0_backlight              = 50
lcd1_backlight              = 50
```

11.2. [lcd0_para]

LCD 屏 0 的配置.

配置项	配置项含义
lcd_used=xx	是否使用 lcd0
lcd_driver_name = "xx"	定义驱动名称
lcd_bl_0_percent = 0	背光矫正，亮度 0%对应背光等级的 0% (255 * 0% = 0)
lcd_bl_40_percent = 23	背光矫正，亮度 40%对应背光等级的 23% (255 * 23% = 58.6)
lcd_bl_100_percent = 100	背光矫正，亮度 100%对应背光等级的 100% (255 * 100% = 255)
lcd_if =xx	lcd 接口(0:hv(sync+de); 1:8080; 2:ttl; 3:lvds，4:dsi; 5:edp)
lcd_x=xx	Lcd 分辨率 x
lcd_y =xx	Lcd 分辨率 y
lcd_width = xx	Lcd 屏宽度
lcd_height = xx	Lcd 屏高度
lcd_dclk_freq = xx	Lcd 频率
lcd_pwm_used =	Pwm 是否使用
lcd_pwm_ch =	Pwm 通道
lcd_pwm_freq＝xx	Pwm 频率
lcd_pwm_pol =xx	pwm 属性, 0:positive; 1:negative

lcd_pwm_max_limit = 255	Lcd backlight PWM 最高限制，以亮度值表示. 比如 150，表示背光最高只能调到 150， 0~255 范围内的亮度值将会被线性映射到 0~150 范围内
lcd_hbp=xx	Lcd 行后沿时间
lcd_ht=xx	Lcd 行时间
lcd_hspw = xx	Lcd 行同步脉宽
lcd_vbp=xx	Lcd 场后沿时间
lcd_vt=xx	Lcd 场时间
lcd_vspw=xx	Lcd 场同步脉宽
lcd_frm=xx	Lcd 格式，0:disable; 1:enable rgb666 dither; 2:enable rgb656 dither
lcd_dsi_if = 2	Lcd MIPI DSI panel Interface，这个参数只有在 lcd_if=4 时才有效。定义 MIPI DSI 屏的两种类型，设置相应值的对应含义为： 0：Video mode 1：Command mode Video mode 的 LCD 屏，是实时刷屏的，有 ht，hbp 等时序参数的定义；Command mode 的屏，屏上带有显示 Buffer，一般会有一个 TE 引脚.
lcd_dsi_lane = 4	Lcd MIPI DSI panel Data Lane number，这个参数只有在 lcd_if=4 时才有效。设置相应值的对应含义为： 1：1 data lane 2：2 data lane 3：3 data lane 4：4 data lane
lcd_dsi_format = 0	Lcd MIPI DSI panel Data Pixel Format，这个参数只有在 lcd_if=4 时才有效。设置相应值的对应含义为： 0：Package Pixel Stream, 24bit RGB 1：Loosely Package Pixel Stream, 18bit RGB 2：Package Pixel Stream, 18bit RGB 3：Package Pixel Stream, 16bit RGB
lcd_dsi_te = 0	Lcd MIPI DSI panel Tear Effect，这个参数只有在 lcd_if=4 时才有效。设置相应值的对应含义为： 0：frame trigged automatically 1：frame trigged by te rising edge 2：frame trigged by te falling edge 设置为 0 时，刷屏间隔时间为 lcd_ht × lcd_vt；设置为 1 或 2 时，刷屏间隔时间为两个 te 脉冲。
lcd_gamma_en=xx	Lcdgamma 校正使能
lcd_bright_curve_en=xx	Lcd 亮度曲线校正使能
lcd_cmap_en=xx	Lcd 调色板函数使能
lcdgamma4iep=xx	智能背光参数, lcd gamma vale*10;decrease it while lcd is not bright enough; increase while lcd is too bright
lcd_bl_en=xx	背光使能的 GPIO 配置
lcd_power=xx	Lcd 电源
lcd_power1 = "vcc-lcd-0"	lcd 使用的供电.

lcd_power2 = "vcc_dsi"	dsi 使用的供电.
lcd_gpio_0 = xxx	lcd_gpio_0 的引脚配置.
lcd_gpio_1 = xxx	lcd_gpio_1 的引脚配置.
lcd_io_regulator1 = "vcc-pd"	lcd 的 io 供电配置

配置举例：

```
[lcd0_para]
lcd_used                = 1

lcd_driver_name         = "lt070me05000"
lcd_bl_0_percent        = 0
lcd_bl_40_percent       = 23
lcd_bl_100_percent      = 100
lcd_if                  = 4
lcd_x                   = 1200
lcd_y                   = 1920
lcd_width               = 94
lcd_height              = 150
lcd_dclk_freq           = 156
lcd_pwm_used            = 1
lcd_pwm_ch              = 0
lcd_pwm_freq            = 50000
lcd_pwm_pol             = 1
lcd_pwm_max_limit       = 255
lcd_hbp                 = 80
lcd_ht                  = 1320
lcd_hspw                = 20
lcd_vbp                 = 20
lcd_vt                  = 1960
lcd_vspw                = 1
lcd_frm                 = 0
lcd_cmap_en             = 0
lcd_dsi_if              = 2
lcd_dsi_lane            = 4
lcd_dsi_format          = 0
lcd_dsi_te              = 0
lcd_gamma_en            = 0
lcd_bright_curve_en = 0
lcdgamma4iep            = 22

lcd_bl_en               = port:PD24<1><0><default><1>
lcd_power               = "vcc-dsi-18"
lcd_power1              = "vcc-lcd-0"
lcd_power2              = "vcc_dsi"
```

lcd_gpio_0 = port:PD25<1><0><default><0>
lcd_gpio_1 = port:PD26<1><0><default><0>
lcd_io_regulator1 = "vcc-pd"

11.3. [hdmi_para]

hdmi 的配置，平板未使用.

[hdmi_para]
hdmi_used = 0

11.4. [pwm0_para]

[pwm0_para]
pwm_used = 0
pwm_positive = port:PD28<2><0><default><default>

11.5. [tvout_para]

tvout 的配置，平板未使用.

[tvout_para]
tvout_used =
tvout_channel_num =
tv_en =

11.6. [tvin_para]

tvout 的配置，平板未使用.

tvin_used =
tvin_channel_num =

12. 摄像头(CSI)

A83T CSI 的 sysconfig 部分对应的字段为：[csi0]。

下面举例说明在实际使用中应该如何配置：需要配置[csi0]的公用部分和[csi0]的 vip_dev(x)部分。

12.1. [csi0]

配置项	配置项含义

vip_used	使用 mipi 接口的 sensor 请填 1。
vip_mode	一般填 0。
vip_dev_qty	mipi 接口暂时不支持复用，填 1。
vip_define_sensor_list	如果配置了 system/etc/hawkview/sensor_list_ cfg.ini 文件，填 1，默认填 0。
vip_csi_mck	Mipi mclk 信号的 GPIO 配置。
vip_csi_sck	CSI0 CCI 时钟信号的 GPIO 配置。 如果使用 CSI0 内部 CCI 需要配置该项
vip_csi_sda	CSI0 CCI 数据信号的 GPIO 配置。 如果使用 CSI0 内部 CCI 需要配置该项
vip_csi_pck	pck 时钟输入引脚.
vip_csi_hsync	hsync 同步信号输入引脚
vip_csi_vsync	vsync 同步信号输入引脚
vip_csi_d0	数据输入引脚 0
vip_csi_d1	数据输入引脚 1
vip_csi_d2	数据输入引脚 2
vip_csi_d3	数据输入引脚 3
vip_csi_d4	数据输入引脚 4
vip_csi_d5	数据输入引脚 5
vip_csi_d6	数据输入引脚 6
vip_csi_d7	数据输入引脚 7
vip_csi_d8	数据输入引脚 8
vip_csi_d9	数据输入引脚 9
vip_dev0_mname	设置 sensor 0 名称，如“ov5648 “。
vip_dev0_pos	摄像头位置前置填"front"，后置填"rear"。
vip_dev0_lane	请参考实际模组的 lane 数目填写。
vip_dev0_twi_id	CSI 使用的 IIC 通道序号，查看具体方案原理图，使用 twi0 填 0，如果使用 CSI 内部 CCI 接口则可以不填
vip_dev0_twi_addr	请参考实际模组的 8bit ID 填写，如 0x6c。
vip_dev0_isp_used	如果是 RAW sensor 必须填 1，YUV 填 0。
vip_dev0_fmt	如果是 RAW 格式填 1，YUV 填 0。
vip_dev0_stby_mode	填 0。
vip_dev0_vflip	Sensor 图像垂直翻转。
vip_dev0_hflip	Sensor 图像水平翻转。
vip_dev0_iovdd	IOVDD 配置，请参考实际原理图填写。
vip_dev0_iovdd_vol	IOVDD 电压值一般为 2.8V(2800000)。
vip_dev0_avdd	AVDD 配置，如”axp15_aldo2”。
vip_dev0_avdd_vol	AVDD 电压值，一般为 2.8V(2800000)。
vip_dev0_dvdd	DVDD 配置，如"axp22_eldo1"。
vip_dev0_dvdd_vol	DVDD 电压值参考 datasheet，1.2/1.5/1.8V。
vip_dev0_afvdd	VCM 电源配置一般不用配置。
vip_dev0_afvdd_vol	VCM 电压值为 2.8V。
vip_dev0_power_en	Sensor power enable 引脚 GPIO 配置。

vip_dev0_reset	Sensor reset 引脚 GPIO 配置。
vip_dev0_pwdn	Sensor power down 引脚 GPIO 配置。
vip_dev0_flash_en	闪光灯 enable 引脚 GPIO 配置。
vip_dev0_flash_mode	闪光灯 flash mode 引脚 GPIO 配置。
vip_dev0_af_pwdn	VCM driver power down 引脚 GPIO 配置。
vip_dev0_act_used	模组包含 VCM driver 时候填 1。
vip_dev0_act_name	VCM driver 名字，如“ad5820_act “。
vip_dev0_act_slave	VCM driver slave 地址。
vip_dev1_mname	设置 sensor 1 名称，如“ov5648 “。
vip_dev1_pos	摄像头位置前置填"front"，后置填"rear"。
vip_dev1_lane	请参考实际模组的 lane 数目填写。
vip_dev1_twi_id	CSI 使用的 IIC 通道序号，查看具体方案原理图，使用 twi0 填 0，如果使用 CSI 内部 CCI 接口则可以不填。
vip_dev1_twi_addr	请参考实际模组的 8bit ID 填写，如 0x6c。
vip_dev1_isp_used	如果是 RAW sensor 必须填 1，YUV 填 0。
vip_dev1_fmt	如果是 RAW 格式填 1，YUV 填 0。
vip_dev1_stby_mode	填 0。
vip_dev1_vflip	Sensor 图像垂直翻转。
vip_dev1_hflip	Sensor 图像水平翻转。
vip_dev1_iovdd	IOVDD 配置，请参考实际原理图填写。
vip_dev1_iovdd_vol	IOVDD 电压值一般为 2.8V(2800000)。
vip_dev1_avdd	AVDD 配置，如”axp15_aldo2”。
vip_dev1_avdd_vol	AVDD 电压值，一般为 2.8V(2800000)。
vip_dev1_dvdd	DVDD 配置，如"axp22_eldo1"。
vip_dev1_dvdd_vol	DVDD 电压值参考 datasheet，1.2/1.5/1.8V。
vip_dev1_afvdd	VCM 电源配置一般不用配置。
vip_dev1_afvdd_vol	VCM 电压值为 2.8V。
vip_dev1_power_en	Sensor power enable 引脚 GPIO 配置。
vip_dev1_reset	Sensor reset 引脚 GPIO 配置。
vip_dev1_pwdn	Sensor power down 引脚 GPIO 配置。
vip_dev1_flash_en	闪光灯 enable 引脚 GPIO 配置。
vip_dev1_flash_mode	闪光灯 flash mode 引脚 GPIO 配置。
vip_dev1_af_pwdn	VCM driver power down 引脚 GPIO 配置。
vip_dev1_act_used	模组包含 VCM driver 时候填 1。
vip_dev1_act_name	VCM driver 名字，如“ad5820_act “。
vip_dev1_act_slave	VCM driver slave 地址。

配置举例：

[csi0]

vip_used = 1
vip_mode = 0
vip_dev_qty = 2

```
vip_define_sensor_list        = 0

vip_csi_pck                    = port:PE00<2><default><default><default>
vip_csi_mck                    = port:PE01<2><default><default><default>
vip_csi_hsync                  = port:PE02<2><default><default><default>
vip_csi_vsync                  = port:PE03<2><default><default><default>
vip_csi_d0                     =
vip_csi_d1                     =
vip_csi_d2                     = port:PE06<2><default><default><default>
vip_csi_d3                     = port:PE07<2><default><default><default>
vip_csi_d4                     = port:PE08<2><default><default><default>
vip_csi_d5                     = port:PE09<2><default><default><default>
vip_csi_d6                     = port:PE10<2><default><default><default>
vip_csi_d7                     = port:PE11<2><default><default><default>
vip_csi_d8                     = port:PE12<2><default><default><default>
vip_csi_d9                     = port:PE13<2><default><default><default>

vip_csi_sck                    = port:PE14<2><default><default><default>
vip_csi_sda                    = port:PE15<2><default><default><default>


vip_dev0_mname                 = "ov8858_4lane"
vip_dev0_pos                   = "rear"
vip_dev0_lane                  = 4
vip_dev0_twi_id                = 0
vip_dev0_twi_addr              = 0x20
vip_dev0_isp_used              = 1
vip_dev0_fmt                   = 1
vip_dev0_stby_mode             = 1
vip_dev0_vflip                 = 0
vip_dev0_hflip                 = 0
vip_dev0_iovdd                 = "axp81x_dldo3"
vip_dev0_iovdd_vol             = 2800000
vip_dev0_avdd                  = "axp81x_dldo4"
vip_dev0_avdd_vol              = 2800000
vip_dev0_dvdd                  = "axp81x_eldo1"
vip_dev0_dvdd_vol              = 1200000
vip_dev0_afvdd                 = "axp81x_dldo3"
vip_dev0_afvdd_vol             = 2800000
vip_dev0_power_en              =
vip_dev0_reset                 = port:PE16<1><default><default><default>
vip_dev0_pwdn                  = port:PE17<1><default><default><default>
vip_dev0_flash_en              =
vip_dev0_flash_mode            =
vip_dev0_af_pwdn               =
```

vip_dev0_act_used = 1
vip_dev0_act_name = "dw9714_act"
vip_dev0_act_slave = 0x18

vip_dev1_mname = "hi253"
vip_dev1_pos = "front"
vip_dev1_lane = 1
vip_dev1_twi_id = 0
vip_dev1_twi_addr = 0x40
vip_dev1_isp_used = 0
vip_dev1_fmt = 0
vip_dev1_stby_mode = 1
vip_dev1_vflip = 0
vip_dev1_hflip = 0
vip_dev1_iovdd = "axp81x_dldo3"
vip_dev1_iovdd_vol = 2800000
vip_dev1_avdd = "axp81x_dldo4"
vip_dev1_avdd_vol = 2800000
vip_dev1_dvdd = "axp81x_eldo3"
vip_dev1_dvdd_vol = 1800000
vip_dev1_afvdd = ""
vip_dev1_afvdd_vol = 2800000
vip_dev1_power_en =
vip_dev1_reset = port:PE18<1><default><default><default>
vip_dev1_pwdn = port:PE19<1><default><default><default>
vip_dev1_flash_en =
vip_dev1_flash_mode =
vip_dev1_af_pwdn =

13. SD / MMC

13.1. [mmc0_para]

配置项	配置项含义
sdc_used=xx	SDC 使用控制：1 使用，0 不用
sdc_detmode=xx	检测模式：1-gpio 轮询检测，2-gpio 中断检测，3-卡常在(不可拔插)，4 - manual mode(from proc file system node)，5-DAT3 检测
Sdc_buswidth=xx	位宽：1-1bit，4-4bit
sdc_d1=xx	SDC DATA1 的 GPIO 配置

sdc_d0＝xx	SDC DATA0 的 GPIO 配置
sdc_clk=xx	SDC CLK 的 GPIO 配置
sdc_cmd=xx	SDC CMD 的 GPIO 配置
sdc_d3=xx	SDC DATA3 的 GPIO 配置
sdc_d2=xx	SDC DATA2 的 GPIO 配置
sdc_det=xx	SDC DET 的 GPIO 配置
sdc_use_wp=xx	SDC 写保护配置：1 使用，0 不用
sdc_wp=xx	SDC WP 的 GPIO 配置
sdc_isio=xx	是否是 sdio card,0:不是，1：是
sdc_regulator=xx	假如过卡支持 SD3.0 或者 emmc4.5 的 UHS-I/DDR、HS200，这里就要写成 sdc_regulator = "axp22_eldo2"（原理图上对应的 PMU 供电输出）
sdc_power_supply=xx	SD 卡供电选择

注意 sdc_regulator 和 sdc_power_supply 的区别：前者给卡内部控制器供电，后者给卡供电；

配置举例：

```
[mmc0_para]
sdc_used            = 1
sdc_detmode         = 1
sdc_buswidth        = 4
sdc_clk             = port:PF02<2><1><2><default>
sdc_cmd             = port:PF03<2><1><2><default>
sdc_d0              = port:PF01<2><1><2><default>
sdc_d1              = port:PF00<2><1><2><default>
sdc_d2              = port:PF05<2><1><2><default>
sdc_d3              = port:PF04<2><1><2><default>
sdc_det             = port:PF06<0><1><2><default>
sdc_use_wp          = 0
sdc_wp              =
sdc_isio            = 0
sdc_regulator       = "none"
sdc_power_supply    = "vcc-card"
```

13.2. [mmc1_para]

配置项	配置项含义
sdc_used=xx	SDC 使用控制：1 使用，0 不用
sdc_detmode=xx	检测模式：1-gpio 轮询检测，2-gpio 中断检测，3-卡常在(不可拔插)，4 - manual mode(from proc file system node)，5-DAT3 检测
bus_width=xx	位宽：1-1bit，4-4bit
sdc_d1=xx	SDC DATA1 GPIO 配置
sdc_d0＝xx	SDC DATA0 GPIO 配置

sdc_clk=xx	SDC CLK GPIO 配置
sdc_cmd=xx	SDC CMD GPIO 配置
sdc_d3=xx	SDC DATA3 GPIO 配置
sdc_d2=xx	SDC DATA2 GPIO 配置
sdc_det=xx	SDC DET GPIO 配置
sdc_use_wp=xx	SDC 写保护配置：1 使用，0 不用
sdc_wp=xx	SDC WP GPIO 配置
sdc_isio =xx	是否是 sdio card,0:不是，1：是
sdc_regulator =xx	假如过卡支持 SD3.0 或者 emmc4.5 的 UHS-I/DDR 、 HS200 ， 这里就要写成 sdc_regulator = "axp22_eldo2"（原理图上对应的 PMU 供电输出）
sdc_power_supply=xx	卡供电选择

配置举例：

```
[mmc1_para]
sdc_used            = 1
sdc_detmode         = 4
sdc_buswidth        = 4
sdc_clk             = port:PG00<2><1><2><default>
sdc_cmd              = port:PG01<2><1><2><default>
sdc_d0              = port:PG02<2><1><2><default>
sdc_d1              = port:PG03<2><1><2><default>
sdc_d2              = port:PG04<2><1><2><default>
sdc_d3              = port:PG05<2><1><2><default>
sdc_det             =
sdc_use_wp          = 0
sdc_wp              =
sdc_isio            = 1
sdc_regulator       = "none"
sdc_power_supply    = "none"
```

13.3. [mmc2_para]

配置项	配置项含义
sdc_used=xx	SDC 使用控制：1 使用，0 不用
sdc_detmode=xx	检测模式：1-gpio 轮询检测，2-gpio 中断检测，3-卡常在(不可拔插)，4 - manual mode(from proc file system node)，5-DAT3 检测
Sdc_bus_width=xx	位宽：1-1bit，4-4bit，8-8bit
sdc_d1=xx	SDC DATA1 GPIO 配置
sdc_d0＝xx	SDC DATA0 GPIO 配置
sdc_clk=xx	SDC CLK GPIO 配置
sdc_cmd=xx	SDC CMD GPIO 配置

sdc_d3=xx	SDC DATA3 GPIO 配置
sdc_d2=xx	SDC DATA2 GPIO 配置
sdc_d4 =xx	SDC DATA4GPIO 配置
sdc_d5 =xx	SDC DATA5 GPIO 配置
sdc_d6 =xx	SDC DATA6 GPIO 配置
sdc_d7 =xx	SDC DATA7 GPIO 配置
sdc_det=xx	SDC DET GPIO 配置
sdc_use_wp=xx	SDC 写保护配置：1 使用，0 不用
sdc_wp=xx	SDC WP GPIO 配置
emmc_rst = xx	Emmc 的 reset 管脚
sdc_isio=xx	是否是 sdio card,0:不是，1：是
sdc_regulator=xx	假如过卡支持 SD3.0 或者 emmc4.5 的 UHS-I/DDR 、 HS200 ， 这里就要写成 sdc_regulator = "axp22_eldo2"
sdc_power_supply=xx	卡供电选择
sdc_2xmode=xx	设置内部控制器的采样模式，默认配置为 1
sdc_ddrmode=xx	是否使用 ddr 模式，默认配置为 1

配置举例：

```
[mmc2_para]
sdc_used            = 0
sdc_detmode         = 3
sdc_buswidth        = 8
sdc_clk             = port:PC05<3><1><2><default>
sdc_cmd              = port:PC06<3><1><2><default>
sdc_d0               = port:PC08<3><1><2><default>
sdc_d1               = port:PC09<3><1><2><default>
sdc_d2               = port:PC10<3><1><2><default>
sdc_d3               = port:PC11<3><1><2><default>
sdc_d4               = port:PC12<3><1><2><default>
sdc_d5               = port:PC13<3><1><2><default>
sdc_d6               = port:PC14<3><1><2><default>
sdc_d7               = port:PC15<3><1><2><default>
emmc_rst             = port:PC16<3><1><2><default>
sdc_det             =
sdc_use_wp           = 0
sdc_wp               =
sdc_isio            = 0
sdc_regulator       = "none"
sdc_power_supply    = "vcc-emmc"
sdc_2xmode           = 1
sdc_ddrmode          = 1
```

14. SIM 卡

sim 卡的配置，平板未使用.

[smc_para]
smc_used =
smc_rst =
smc_vppen =
smc_vppp =
smc_det =
smc_vccen =
smc_sck =
smc_sda =

15. USB 控制标志

15.1. [usbc0]

配置项	配置项含义
usb_used =xx	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 =xx	USB 端口的使用情况。(xx=0/1/2) 0：device only 1：host only 2：OTG
usb_detect_type=xx	USB 端口的检查方式。 0：无检查方式 1：vbus/id 检查
usb_id_gpio=xx	USB ID pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_det_vbus_gpio＝xx	USB DET_VBUS pin 脚配置。如果 GPIO 提供 pin，请参考 gpio 配置说明《配置与 GPIO 管理.doc》。如果的 AXP 提供 pin,则配置为："axp_ctrl"。
usb_drv_vbus_gpio=xx	USB DRY_VBUS pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_restrict_gpio=xx	USB 限流控制 pin 脚 USB RESTRICT_GPIO pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_host_init_state=xx	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_restric_flag=xx	Usb 限流标志位

	0：不使能限流功能　1：使能限流功能
usb_restric_voltage=xx	限流开启的条件 电压值小于设置值，则开启限流
usb_restric_capacity=xx	限流开启的条件 电量值小于设置值，则开启限流

配置举例：

```
[usbc0]
usb_used                = 1
usb_port_type           = 2
usb_detect_type         = 1
usb_id_gpio             = port:PH11<0><1><default><default>
usb_det_vbus_gpio       = "axp_ctrl"
usb_drv_vbus_gpio       = port:power4<1><0><default><0>
usb_restrict_gpio       =
usb_host_init_state = 0
usb_restric_flag        = 0
usb_restric_voltage = 3550000
usb_restric_capacity= 5
```

15.2. [usbc1]

配置项	配置项含义
usb_used =xx	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 =xx	USB 端口的使用情况。(xx=0/1/2) 0：device only　1：host only　2：OTG
usb_detect_type=xx	USB 端口的检查方式。 0：无检查方式　1：vbus/id 检查
usb_drv_vbus_gpio＝xx	USB DET_VBUS pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_restrict_gpio=xx	USB 限流控制 pin 脚 USB RESTRICT_GPIO pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_host_init_state=xx	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_restric_flag=xx	Usb 限流标志位 0：表不设限流，1 开启限流
usb_not_suspend=xx	该控制器是否支持 remote wakeup 1：支持；0：不支持

配置举例：

[usbc1]
usb_used = 0
usb_port_type = 1
usb_detect_type = 0
usb_drv_vbus_gpio =
usb_restrict_gpio =
usb_host_init_state = 1
usb_restric_flag = 0
usb_not_suspend = 0

15.3. [usbc2]

配置项	配置项含义
usb_used =xx	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 =xx	USB 端口的使用情况。(xx=0/1/2) 0：device only 1：host only 2：OTG
usb_detect_type=xx	USB 端口的检查方式。 0：无检查方式 1：vbus/id 检查
usb_drv_vbus_gpio＝xx	USB DET_VBUS pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_restrict_gpio=xx	USB 限流控制 pin 脚 USB RESTRICT_GPIO pin 脚配置。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
usb_host_init_state=xx	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_restric_flag=xx	Usb 限流标志位 0：表不设限流，1 开启限流

配置举例:
[usbc2]
usb_used = 0
usb_port_type = 1
usb_detect_type = 0
usb_drv_vbus_gpio =
usb_restrict_gpio =
usb_host_init_state = 1
usb_restric_flag = 0

16. USB Device

16.1. [usb_feature]

配置项	配置项含义
vendor_id=xx	USB 厂商 ID
mass_storage_id =xx	U 盘 ID
adb_id =xx	USB 调试桥 ID
manufacturer_name=xx	USB 厂商名
product_name＝xx	USB 产品名
serial_number=xx	USB 序列号

配置举例：

```
[usb_feature]
vendor_id                = 0x18D1
mass_storage_id          = 0x0001
adb_id                   = 0x0002

manufacturer_name        = "USB Developer"
product_name             = "Android"
serial_number            = "20080411"
```

16.2. [msc_feature]

配置项	配置项含义
vendor_name=xx	U 盘 厂商名
product_name=xx	U 盘产品名
release=xx	发布版本
luns=xx	U 盘逻辑单元的个数（PC 可以看到的 U 盘盘符的个数）

配置举例：

```
[msc_feature]
vendor_name              = "USB 2.0"
product_name             = "USB Flash Driver"
release                  = 100
luns                     = 3
```

17. USB Serial Feature

17.1. [serial_feature]

配置项	配置项含义
serial_unique	usb 序列号开关，0：不用，1：用

配置举例：

[serial_feature]

serial_unique = 1

18. 马达

18.1. [motor_para]

配置项	配置项含义
motor_used =xx	是否启用马达，启用置 1，反之置 0
motor_shake=xx	马达使用的 GPIO 配置. 用于给马达供电.
motor_ldo = xx	指明马达由 axp 哪一路电压供电
motor_ldo_voltage = xx	Axp 供电的该路电压为 xx mv

配置举例：

[motor_para]

motor_used = 1

motor_shake = 0

motor_ldo = vcc-vibrator

motor_ldo_voltage = 3300

注意事项：

motor_shake = port:power3<1><default><default><1>

默认 io 口的输出应该为 1，这样就不会初始化之后就开始震动了。

假设 motor_shake = 0 ，说明没有指定 gpio 引脚，那么就会设置 axp 的引脚为马达供电，优先考虑 gpio 配置。

19. 重力感应(G Sensor)

19.1. [gsensor_para]

配置项	配置项含义

gsensor_used=xx	是否支持 gsensor
gsensor_twi_id =xx	I2C 的 BUS 控制选择，0：TWI0;1:TWI1;2:TWI2
gsensor_twi_addr=xx	芯片的 I2C 地址
gsensor_int1＝xx	中断 1 的 GPIO 配置
gsensor_int2=xx	中断 2 的 GPIO 配置

配置举例：

```
[gsensor_para]
gsensor_used            = 1
gsensor_twi_id          = 1
gsensor_twi_addr        = 0x1E
gsensor_int1            = port:PH08<6><1><default><default>
gsensor_int2            = port:PH09<6><1><default><default>
```

20. 重力感应(G Sensor)自动扫描配置

注：目前方案中支持 gsensor 的类型有以下列表，作为自动检测加载的时候使用，为‘1’时检测加载，为‘0’时不检测。

20.1. [gsensor_list_para]

配置项	配置项含义
gsensor_det_used	0 or 1，是否使用自动检测功能
lsm9ds0_acc_mag	0 or 1，是否使用自动检测功能
bma250	0 or 1，检测时是否扫描此类触屏
mma8452	0 or 1，检测时是否扫描此类触屏
mma7660	0 or 1，检测时是否扫描此类触屏
mma865x	0 or 1，检测时是否扫描此类触屏
afa750	0 or 1，检测时是否扫描此类触屏
lis3de_acc	0 or 1，检测时是否扫描此类触屏
lis3dh_acc	0 or 1，检测时是否扫描此类触屏
kxtik	0 or 1，检测时是否扫描此类触屏
dmard10	0 or 1，检测时是否扫描此类触屏
dmard06	0 or 1，检测时是否扫描此类触屏
mxc622x	0 or 1，检测时是否扫描此类触屏
fxos8700	0 or 1，检测时是否扫描此类触屏
lsm303d	0 or 1，检测时是否扫描此类触屏

配置举例：

```
[gsensor_list_para]
gsensor_det_used            = 1
lsm9ds0_acc_mag             = 1
```

bma250 = 1
mma8452 = 1
mma7660 = 1
mma865x = 1
afa750 = 1
lis3de_acc = 1
lis3dh_acc = 1
kxtik = 1
dmard10 = 0
dmard06 = 1
mxc622x = 1
fxos8700 = 1
lsm303d = 0

21. WiFi

21.1. [rf_para]

配置项	配置项含义
module_num	选择的模组： 0- none 1- ap6181(wifi) 2- ap6210(wifi+bt) 3- rtl8188eu(wifi) 4- rtl8723au(wifi+bt) 5- rtl8723bs(wifi+bt) 6- esp8089(wifi) 7- ap6476(wifi+bt+fm+gps) 8- ap6330(wifi+bt+fm) 9- gb9663(wifi+bt+fm)
module_power1	模组供电1，请参考 PMU 使用文档
module_power1_vol	模组供电1电压
module_power2	模组供电2，请参考 PMU 使用文档
module_power2_vol	模组供电2电压
module_power3	模组供电3，请参考 PMU 使用文档
module_power3_vol	模组供电3电压
power_switch	
chip_en	8723bs WiFi、BT 功能总使能脚
lpo_use_apclk	模组使用的系统32k 时钟

配置举例：

```
[rf_para]
module_num          = 9
module_power1       = "vcc-wifi-io"
module_power1_vol   =
module_power2       = ""
module_power2_vol   =
module_power3       = ""
module_power3_vol   =
power_switch        =
chip_en             =
lpo_use_apclk       = "ac10032k2"
```

21.2. [wifi_para]

配置项	配置项含义
wifi_used	是否使用 WiFi
wifi_sdc_id	WiFi 使用的 sdio 接口编号

wifi_usbc_id	WiFi 使用的 USB 接口编号
wifi_usbc_type	WiFi 使用的 USB 控制器类型 1- EHCI(speed 2.0) 2- OHCI(speed 1.0)
wl_reg_on	WiFi 功能使能 GPIO
wl_host_wake	WiFi 唤醒主控 GPIO
wl_host_wake_invert	wl_host_wake 是否接反相器

说明：[wifi_para]下的配置项是 usb 和 sdio 接口 wifi 共用的。

配置举例：

[wifi_para]

wifi_used = 1

wifi_sdc_id = 1

wifi_usbc_id = 1

wifi_usbc_type = 1

wl_reg_on = port:PL08<1><default><default><0>

wl_host_wake = port:PL04<4><default><default><0>

wl_host_wake_invert = 0

22. 蓝牙(blueteeth)

22.1. [bt_para]

配置项	配置项含义
bt_used	是否使用 BT
bt_uart_id	BT 连接的 UART 接口编号
bt_rst_n	BT_RESET 脚
bt_wake	BT_WAKE_AP 脚
bt_host_wake	AP_WAKE_BT 脚
bt_host_wake_invert	bt_host_wake 是否接反相器

配置举例：

bt_used = 1

bt_uart_id = 1

bt_rst_n = port:PL07<1><default><default><0>

bt_wake = port:PL06<1><default><default><0>

bt_host_wake = port:PL05<4><default><default><0>

bt_host_wake_invert = 0

23. GPS

23.1. [gps_para]

配置项	配置项含义
gps_used	GPS 使能标志位。0：禁用；1：使能
gps_uard_id	GPS 使用的串口号
gps_vbat	GPS 的供电选择. 比如 vcc-io
gps_standby_n	GPS 的 standby 引脚. 0: standby, 1: 非 standby.
gps_rst_n	GPS 的 reset 引脚. 0: reset, 1: 非 reset.
gps_clk	GPS 的时钟源.

配置举例：

[gps_para]
gps_used = 1
gps_uard_id = 2
gps_vbat =
gps_standby_n = port:PC17<1><default><default><0>
gps_rst_n = port:PC18<1><default><default><0>
gps_clk = "ac10032k3"

24. 3G

24.1. [3g_para]

配置项	配置项含义
3g_used	3G 使能标志位。 0：禁用；1：使能
3g_usbc_num	3G 使用到的 USB 控制器编号。 0：USB0；1：USB1；2：USB2；3：USB3 等
3g_uart_num	3G 使用到的 UART 控制器编号。 0：UART0；1：UART1；2：UART2；3：UART3 等
bb_name	3G 模组名称。如“mu509”
bb_vbat	gpio 配置，电池引脚。
bb_on	保留
bb_pwr_on	gpio 配置，供电引脚。
bb_wake	gpio 配置，A31 睡眠唤醒 3G 模组。
bb_rf_dis	gpio 配置，用来控制无线发射模块。
bb_rst	gpio 配置，用来复位 3G 模组。
bb_dldo	电源控制配置
bb_dldo_min_uV	电压下限

bb_dldl_max_uV	电压上限

配置举例：

```
[3g_para]
3g_used         = 0
3g_usbc_num   = 3
3g_uart_num   = 4
bb_name         = "mu509"
bb_vbat         =
bb_on           =
bb_pwr_on       =
bb_wake         =
bb_rf_dis =
bb_rst          =
bb_dldo             = "axp22_sw0"
bb_dldo_min_uV     = 5000000
bb_dldo_max_uV     = 5000000
```

25. 陀螺仪

25.1. [gy_para]

配置项	配置项含义
gy_used	陀螺仪使能标志位。0：禁用；1：使能
gy_twi_id	陀螺仪使用的 twi 通道号.
gy_twi_addr	陀螺仪 twi 设备地址.
gy_int1	陀螺仪中断 GPIO 引脚一.
gy_int2	陀螺仪中断 GPIO 引脚二.

配置举例：

```
[gy_para]
gy_used                 = 1
gy_twi_id               = 1
gy_twi_addr             = 0x6a
gy_int1                 = port:PH08<6><1><default><default>
gy_int2                 = port:PH09<6><1><default><default>
```

25.2. [gy_list_para]

陀螺仪自动扫描列表.

配置项	配置项含义

gy_det_used	是否支持自动扫描; 0: 不支持, 1: 支持
lsm9ds0_gyr	是否扫描 lsm9ds0
l3gd20_gyr	是否扫描 l3gd20
bmg160_gyr	是否扫描 bmg160

配置举例：

[gy_list_para]

gy_det_used = 1

lsm9ds0_gyr = 1

l3gd20_gyr = 0

bmg160_gyr = 1

26. 光感(light sensor)

26.1. [ls_para]

配置项	配置项含义
ls_used =xx	是否支持 ls
ls_twi_id=xx	I2C 的 BUS 控制选择，0：TWI0;1:TWI1;2:TWI2
ls_twi_addr =xx	芯片的 I2C 地址
ls_int=xx	中断的 GPIO 配置

配置举例：

[ls_para]

ls_used = 1

ls_twi_id = 1

ls_twi_addr = 0x39

ls_int = port:PH10<6><1><default><default>

26.2. [ls_list_para]

光感自动扫描列表.

配置项	配置项含义
[ls_list_para]	是否支持自动扫描; 0: 不支持, 1: 支持
ltr_501als	是否扫描 ltr_501als
jsa1212	是否扫描 jsa1212
jsa1127	是否扫描 jsa1127

配置举例：

[ls_list_para]

ls_det_used = 1

ltr_501als = 1

jsa1212 = 0
jsa1127 = 1

27. 罗盘 Compass

27.1. [compass_para]

配置项	配置项含义
compass_used=xx	是否支持 compass
compass_twi_id=xx	I2C 的 BUS 控制选择，0：TWI0;1:TWI1;2:TWI2
compass_twi_addr =xx	芯片的 I2C 地址
compass_int1=xx	中断的 GPIO 配置一
compass_int2=xx	中断的 GPIO 配置二

目前都用轮询，不用中断方式.

配置举例：

[compass_para]
compass_used = 1
compass_twi_id = 1
compass_twi_addr = 0x1E
compass_int1 = port:PH08<6><1><default><default>
compass_int2 = port:PH09<6><1><default><default>

27.2. [compass_list_para]

配置项	配置项含义
compass_det_used	是否支持自动扫描; 0: 不支持, 1: 支持
lsm9ds0	是否扫描 lsm9ds0
lsm303d	是否扫描 lsm303d
akm8963	是否扫描 akm8963

配置举例：

[compass_list_para]
compass_det_used = 1
lsm9ds0 = 1
lsm303d = 0
akm8963 = 1

28. 音频

28.1. [audio0]

配置项	配置项含义
audio_int_ctrl	耳机检测中断引脚配置
audio_pa_ctrl	功放控制引脚配置
speaker_val	暂时无用
headset_val	耳机增益
single_speaker_val	单喇叭增益
double_speaker_val	双喇叭增益
speaker_double_used	双喇叭标志，0 单喇叭
earpiece_val	听筒增益
mainmic_val	主 mic 增益
headsetmic_val	耳机 mic 增益
ts3a225_gpio_ctrl	富士康方案耳机检测脚
dmic_used	数字 mic 使用标志
adc_digital_val	数字 mic 增益
agc_used	硬件 agc 使用标志
drc_used	硬件 drc 使用标志
analog_bb	模拟 bb 使用标志
digital_bb	数字 bb 使用标志

配置举例:

```
[audio0]
audio_int_ctrl              = port:PL12<6><default><default><0>
audio_pa_ctrl               = port:PG13<1><default><default><0>
;aif3_voltage               = "axp15_cldo3"
speaker_val = 0x1b
headset_val = 0x3b
single_speaker_val = 0x19
double_speaker_val = 0x19
speaker_double_used = 1
earpiece_val = 0x1e
mainmic_val = 0x4
headsetmic_val = 0x4
ts3a225_gpio_ctrl           = port:PG12<6><default><default><0>
dmic_used = 1
adc_digital_val = 0xc0c0
agc_used = 0
drc_used = 1
analog_bb = 0
digital_bb = 0
```

29. 红外

29.1. [s_cir0]

接收端配置.

配置项	配置项含义
ir_used	平台是否使用 ir. 0: 不使用, 1: 使用.
ir_rx	红外 IC 的接收引脚
ir_power_key_code	power 按键对应的键值
ir_addr_code	接收的设备地址码

配置举例：

[s_cir0]

ir_used = 0

ir_rx =

ir_power_key_code = 0x0

ir_addr_code = 0x0

29.2. [cir]

发送端配置.

配置项	配置项含义
ir_used	平台是否使用 ir. 0: 不使用, 1: 使用.
ir_tx	红外 IC 的发送引脚

配置举例：

[cir]

ir_used = 1

ir_tx = port:PH07<2><default><default><default>

30. PMU 电源

30.1. [pmu1_para]

配置项	相关说明
pmu_used	是否使用 AXPxx：0:不使用,1:使用
pmu_id	0:axp19x,1:axp22x,2:axp806,3:axp808,4:axp809,5:axp803,6:axp813
pmu_twi_addr	AXPxx 通信 I2C 地址
pmu_twi_id	AXPxx 挂接在主控的哪个 I2C 控制口（0，1，2 ...）
pmu_irq_id	irq 号 (0 irq0,1 irq1,……)

pmu_battery_rdc	电池通路内阻，单位 mΩ
pmu_battery_cap	电池容量,单位 mAh，如果配置改值，计量方式为库仑计方式，否则为电压方式
pmu_batdeten	电池检查使能控制：0:使能 1:使能
pmu_chg_ic_temp	智能充电，PMU 温度设置。配置为 0 关闭此功能. 目标是在 PMU 温度恒定的情况下，使充电电流达到最大值
pmu_runtime_chgcur	设置开机时充电电流大小,单位 mA，仅支持:300/450/600/750/900/1050/1200/1350/1500/1650/1800/1950/2100
pmu_earlysuspend_chgcur	设置关屏时充电电流大小，单位 mA，仅支持：300/4500/600/750/900/1050/1200/1350/1500/1650/1800/1950/2100
pmu_suspend_chgcur	设置待机时充电电流大小，单位 mA，仅支持：300/4500/600/750/900/1050/1200/1350/1500/1650/1800/1950/2100
pmu_shutdown_chgcur	设置关机时充电电流大小，单位 mA，仅支持：300/4500/600/750/900/1050/1200/1350/1500/1650/1800/1950/2100
pmu_init_chgvol	设置充电完成时电池目标电压,仅支持：4100/4200/4220/4240mV
pmu_init_chgend_rate	设置充电结束时电流占恒流值的百分比：10/15
pmu_init_chg_enabled	开机后充电使能初始值：0：不开充电，1：开充电
pmu_init_bc_en	BC 模块的使能控制, BC 模块为硬件自动识别 USB 及 AC 口的模块. 1: 使能, 0: 不使能.
pmu_init_adc_freq	ADC 采样频率设定值：100/200/400/800 Hz
pmu_init_adcts_freq	TS ADC 采样频率设定值：100/200/400/800 Hz
pmu_init_chg_pretime	涓流充电超时时间：40/50/60/70 分钟
pmu_init_chg_csttime	恒流超时时间：360/480/600/720 分钟
pmu_batt_cap_correct	满足电池容量校正条件后是否校正电池容量控制 0：不校正 1：校正
pmu_bat_regu_en	充电结束时，充电开关是否关闭：0：关闭 1：不关闭
pmu_bat_para1	电池空载电压为 3.13V 对应的电量值
pmu_bat_para2	电池空载电压为 3.27V 对应的电量值
pmu_bat_para3	电池空载电压为 3.34V 对应的电量值
pmu_bat_para4	电池空载电压为 3.41V 对应的电量值
pmu_bat_para5	电池空载电压为 3.58V 对应的电量值
pmu_bat_para6	电池空载电压为 3.52V 对应的电量值
pmu_bat_para7	电池空载电压为 3.55V 对应的电量值
pmu_bat_para8	电池空载电压为 3.57V 对应的电量值
pmu_bat_para9	电池空载电压为 3.59V 对应的电量值
pmu_bat_para10	电池空载电压为 3.61V 对应的电量值
pmu_bat_para11	电池空载电压为 3.63V 对应的电量值
pmu_bat_para12	电池空载电压为 3.64V 对应的电量值
pmu_bat_para13	电池空载电压为 3.66V 对应的电量值
pmu_bat_para14	电池空载电压为 3.7V 对应的电量值
pmu_bat_para15	电池空载电压为 3.73V 对应的电量值

pmu_bat_para16	电池空载电压为 3.77V 对应的电量值
pmu_bat_para17	电池空载电压为 3.78V 对应的电量值
pmu_bat_para18	电池空载电压为 3.8V 对应的电量值
pmu_bat_para19	电池空载电压为 3.82V 对应的电量值
pmu_bat_para20	电池空载电压为 3.84V 对应的电量值
pmu_bat_para21	电池空载电压为 3.85V 对应的电量值
pmu_bat_para22	电池空载电压为 3.87V 对应的电量值
pmu_bat_para23	电池空载电压为 3.91V 对应的电量值
pmu_bat_para24	电池空载电压为 3.94V 对应的电量值
pmu_bat_para25	电池空载电压为 3.98V 对应的电量值
pmu_bat_para26	电池空载电压为 4.01V 对应的电量值
pmu_bat_para27	电池空载电压为 4.05V 对应的电量值
pmu_bat_para28	电池空载电压为 4.08V 对应的电量值
pmu_bat_para29	电池空载电压为 4.1V 对应的电量值
pmu_bat_para30	电池空载电压为 4.12V 对应的电量值
pmu_bat_para31	电池空载电压为 4.14V 对应的电量值
pmu_bat_para32	电池空载电压为 4.15V 对应的电量值
pmu_usbvol_limit	USB 适配器限压功能控制 0：不使能 1：使能
pmu_usbcur_limit	USB 适配器限流功能控制 0：不使能 1：使能
pmu_usbvol	设置 USB 适配器限压值：4000/4100/4200/4300/4400/4500/4600 4700 mV，0-不限压
pmu_usbcur	设置 USB 适配器限流值：500/900mA，0-不限流
pmu_usbvol_pc	设置 USB 连接 PC 时限压值：4000/4100/4200/4300/4400/4500 4600/4700 mV，0-不限压
pmu_usbcur_pc	设置 USB 连接 PC 时限流值：500/900mA，0-不限流
pmu_pwroff_vol	PMU 关机时，硬件低电保护电压设置值：2600/2700/2800/2900 /3000/3100/3200/3300 mV
pmu_pwron_vol	PMU 开机后，硬件低电保护电压设置值：2600/2700/2800/2900 /3000/3100/3200/3300 mV
pmu_pekoff_time	长按键关机时间设置值：4000/6000/8000/10000 ms
pmu_pekoff_func	长按键功能配置项：0：长按键后关机 1：长按键后重启
pmu_pekoff_en	长按键后是否关闭 PMU：0：不关闭 1：关闭
pmu_pekoff_delay_time	长按键关机激活时间设置， 0/10/20/30/40/50/60/70 秒
pmu_peklong_time	报长按键消息时间设定值：1000/1500/2000/2500 ms
pmu_pekon_time	关机情况下按键多长时间后启动设置：128/1000/2000/3000 ms
pmu_pwrok_time	PWROK 启动延时时间设置值：8/16/32/64 ms
pmu_pwrok_shutdown_en	长按 PWROK 键 6s 是否关机，使能位
pmu_reset_shutdown_en	Reset 键重启时，是否关闭所有的 ldo/dcdc，0：不关闭，1：关闭。
pmu_battery_warning_level1	低电报警门限 level 1 设置值百分比：5~20，每步设置 1%
pmu_battery_warning_level2	低电报警门限 level 2 设置值百分比：0~15，每步设置 1%
pmu_restvol_adjust_time	电池电量更新时间设置值：30/60/120 s
pmu_ocv_cou_adjust_time	根据 OCV 校正电池电量更新时间值：30/60/120 s
pmu_chgled_func	CHGLED 功能控制：0：马达驱动 1：充电状态指示

pmu_chgled_type	CHGLED 作为充电状态指示时指示功能控制：0：方式 A 1：方式 B
pmu_vbusen_func	N_VBUSEN 工作方式控制：0：作为输入脚 1：作为输出脚
pmu_reset	长按键 16s 后 PMU 是否重启控制：0：不重启 1：重启
pmu_IRQ_wakeup	在关机和休眠状态下 IRQ 为低电平时是否触发开机和唤醒控制 0：不开机或不唤醒 1：开机或唤醒
pmu_hot_shutdowm	PMU 过温后是否关机 0：不关机 1：关机
pmu_inshort	是否手动设置 ACIN/VBUS 短路控制 0：PMU 自动检测 1：手动设置 ACIN 和 VBUS 为短路
power_start	火牛开机选择 0: 不允许插火牛直接开机，必须通过判断：满足以下条件可以直接开机：长按 power 按键，前次是系统状态，如果电池电量过低，则不允许开机 1: 任意状态下，允许插火牛直接开机，同时要求电池电量足够高 2: 不允许插火牛直接开机，必须通过判断：满足以下条件可以直接开机：长按 power 按键，前次是系统状态，不要求电池电量 3: 任意状态下，允许插火牛直接开机，不要求电池电量
pmu_temp_enable	电池温度检测使能控制：0：disable 1：enable
pmu_charge_ltf	充电下限电池温度对应的电压
pmu_charge_htf	充电上限电池温度对应的电压
pmu_discharge_ltf	关机下限电池温度对应的电压
pmu_discharge_htf	关机上限电池温度对应的电压
pmu_temp_para1	电池温度-25 度对应的电压
pmu_temp_para2	电池温度-15 度对应的电压
pmu_temp_para3	电池温度-10 度对应的电压
pmu_temp_para4	电池温度-5 度对应的电压
pmu_temp_para5	电池温度 0 度对应的电压
pmu_temp_para6	电池温度 5 度对应的电压
pmu_temp_para7	电池温度 10 度对应的电压
pmu_temp_para8	电池温度 20 度对应的电压
pmu_temp_para9	电池温度 30 度对应的电压
pmu_temp_para10	电池温度 40 度对应的电压
pmu_temp_para11	电池温度 45 度对应的电压
pmu_temp_para12	电池温度 50 度对应的电压
pmu_temp_para13	电池温度 55 度对应的电压
pmu_temp_para14	电池温度 60 度对应的电压
pmu_temp_para15	电池温度 70 度对应的电压
pmu_temp_para16	电池温度 80 度对应的电压

配置举例：

[pmu1_para]

pmu_used = 1

pmu_id = 6

pmu_twi_addr = 0x34
pmu_twi_id = 0
pmu_irq_id = 0
pmu_battery_rdc = 133
pmu_battery_cap = 2962
pmu_batdeten = 1
pmu_chg_ic_temp = 0
pmu_runtime_chgcur = 1000
pmu_earlysuspend_chgcur = 1000
pmu_suspend_chgcur = 1800
pmu_shutdown_chgcur = 1950
pmu_init_chgvol = 4200
pmu_init_chgend_rate = 20
pmu_init_chg_enabled = 1
pmu_init_bc_en = 0
pmu_init_adc_freq = 800
pmu_init_adcts_freq = 800
pmu_init_chg_pretime = 70
pmu_init_chg_csttime = 720
pmu_batt_cap_correct = 1
pmu_bat_regu_en = 1

pmu_bat_para1 = 0
pmu_bat_para2 = 0
pmu_bat_para3 = 0
pmu_bat_para4 = 0
pmu_bat_para5 = 0
pmu_bat_para6 = 0
pmu_bat_para7 = 1
pmu_bat_para8 = 1
pmu_bat_para9 = 2
pmu_bat_para10 = 3
pmu_bat_para11 = 4
pmu_bat_para12 = 12
pmu_bat_para13 = 17
pmu_bat_para14 = 29
pmu_bat_para15 = 43
pmu_bat_para16 = 48
pmu_bat_para17 = 53
pmu_bat_para18 = 56
pmu_bat_para19 = 60
pmu_bat_para20 = 66
pmu_bat_para21 = 71
pmu_bat_para22 = 77
pmu_bat_para23 = 81

pmu_bat_para24 = 85
pmu_bat_para25 = 90
pmu_bat_para26 = 95
pmu_bat_para27 = 98
pmu_bat_para28 = 100
pmu_bat_para29 = 100
pmu_bat_para30 = 100
pmu_bat_para31 = 100
pmu_bat_para32 = 100

pmu_usbvol_limit = 0
pmu_usbcur_limit = 0
pmu_usbvol = 4000
pmu_usbcur = 0
pmu_usbvol_pc = 4400
pmu_usbcur_pc = 500
pmu_pwroff_vol = 3300
pmu_pwron_vol = 2600
pmu_pekoff_time = 6000
pmu_pekoff_func = 0
pmu_pekoff_en = 1
pmu_pekoff_delay_time = 0
pmu_peklong_time = 1500
pmu_pekon_time = 1000
pmu_pwrok_time = 64
pmu_pwrok_shutdown_en = 0
pmu_reset_shutdown_en = 1
pmu_battery_warning_level1 = 15
pmu_battery_warning_level2 = 0
pmu_restvol_adjust_time = 60
pmu_ocv_cou_adjust_time = 60
pmu_chgled_func = 0
pmu_chgled_type = 0
pmu_vbusen_func = 1
pmu_reset = 0
pmu_IRQ_wakeup = 1
pmu_hot_shutdowm = 1
pmu_inshort = 0
power_start = 0

pmu_temp_enable = 0
pmu_charge_ltf = 2261
pmu_charge_htf = 388
pmu_discharge_ltf = 3200
pmu_discharge_htf = 237

```
pmu_temp_para1          = 7466
pmu_temp_para2          = 4480
pmu_temp_para3          = 3518
pmu_temp_para4          = 2786
pmu_temp_para5          = 2223
pmu_temp_para6          = 1788
pmu_temp_para7          = 1448
pmu_temp_para8          = 969
pmu_temp_para9          = 664
pmu_temp_para10         = 466
pmu_temp_para11         = 393
pmu_temp_para12         = 333
pmu_temp_para13         = 283
pmu_temp_para14         = 242
pmu_temp_para15         = 179
pmu_temp_para16         = 134
```

30.2. [pmu1_regu]

供电依赖关系表.

```
[pmu1_regu]
regulator_count = 23
regulator1      = "axp81x_dcdc1 none vcc-emmc vcc-usb0-33 vcc-io vcc-io-gps vcc-io1 vcc-sensor"
regulator2      = "axp81x_dcdc2 none vdd-cpua"
regulator3      = "axp81x_dcdc3 none vdd-cpub"
regulator4      = "axp81x_dcdc4 none vdd-gpu"
regulator5      = "axp81x_dcdc5 none vcc-dram"
regulator6      = "axp81x_dcdc6 none vdd-sys vdd-usb0-09 vdd-hdmi-09"
regulator7      = "axp81x_dcdc7 none"
regulator8      = "axp81x_rtc none"
regulator9      = "axp81x_aldo1 none vcc-dsi-18 vcc-csi2-18 vcc-lvds-18 vcc-efuse-18 vcc-hdmi-18 vcc-pd"
regulator10     = "axp81x_aldo2 none vdd-drampll vdd-lpddr-18 vcc-pll vcc-adc vcc-cpvdd vcc-ldoin"
regulator11     = "axp81x_aldo3 none vcc-avcc vcc-pl"
regulator12     = "axp81x_dldo1 none vcc-wifi-io vcc-io2"
regulator13     = "axp81x_dldo2 none vcc-lcd-0"
regulator14     = "axp81x_dldo3 none afvcc-csi iovdd-csi"
regulator15     = "axp81x_dldo4 none avdd-csi"
regulator16     = "axp81x_eldo1 1 dvdd-csi-12"
regulator17     = "axp81x_eldo2 1 vcc_dsi"
regulator18     = "axp81x_eldo3 1 dvdd-csi-18"
regulator19     = "axp81x_fldo1 none vcc-hsic-12"
regulator20     = "axp81x_fldo2 none vdd-cpus"
regulator21     = "axp81x_gpio0ldo none vcc-ctp"
regulator22     = "axp81x_gpio1ldo none vcc-card"
```

regulator23 = "axp81x_dc1sw 1"

此表，为电源拓扑结构的配置。

第一列，为 ldo/dcdc 的 name。如 axp81x_dcdc1。

第二列，为 ldo 之间依赖关系的配置项，如 eldo 的输入端为 dcdc1,则 eldo 此配置为 1，代表此路 ldo 依赖于 ldo1 对应的 axp81x_dcdc1。如果，不依赖于其他 ldo，则配置成 none。

第三列，为系统电配置项，当为系统电时，配置为 sys，否则配置成 none。系统电，是指除了 super standby 外，处于常开状态，驱动中各个模块调用 regulator_disable 函数，无法关闭此路输出。

从第四列，开始，为各个模块 supply id 的配置项，供各个驱动模块调用，regulator 获得句柄时使用。supply id 之间以空格隔开，每个 supply id 规定为 20 个字符以内，每个 ldo 支持的 supply id 个数无限制。

注意：第一列的 ldo/dcdc 的 name，不要去修改，standby 会根据这个 name 来区分各路电。
第二列的 ldo 之间依赖关系，请根据实际原理图去配置。
第三列，系统电，与芯片内部的 power domin 有关，系统进入 standby 时，会决定是否关闭此路电。修改请慎重。

31. Recovery 键配置

配置项		配置项含义
key_min	= 3	作为 recovery 功能的按键的键值范围下限
key_max	= 5	作为 recovery 功能的按键的键值范围上限

配置举例：

[recovery_key]

key_min = 3

key_max = 5

说明：

通常情况下，一块方案板上的按键个数不同，或者排列不同，这都导致了方案商在选择作为开机阶段 recovery 功能的按键有所不同。该键值配置用于作为 recovery 功能的按键的键值范围落在 key_min 到 key_max 之间。

32. DVFS

32.1. CPU DVFS

A83T 平台有三份 VF 表，分别对应 slow，normal，fast 的 IC，不同批次的 IC 的频率性能有差异，用这三份表来区分.

32.1.1. [dvfs_table]

配置项	配置项含义
vf_table_count	平台包含的 vf 表的个数.

配置示例:

[dvfs_table]

vf_table_count = 3

32.1.2. [vf_table0]

对应频率性能为 slow 的 IC.

配置项	配置项含义
L_max_freq	cluster0 的 cpu 支持的最高频率
L_boot_freq	内核启动阶段 cluster0 的 cpu 频率. 即从 cpufreq 驱动加载到 android 进入主界面之前的 cpu 频率
L_min_freq	cluster0 的 cpu 支持的最低频率
L_LV_count	cluster0 的 cpu 频率电压等级数
L_LV1_freq	cluster0 的 cpu 第一级 (最高) 频率
L_LV1_volt	L_LV1_freq 频点对应的电压
L_LV2_freq	cluster0 的 cpu 第二级 (次高) 频率
L_LV2_volt	L_LV2_freq 频点对应的电压
L_LV3_freq	cluster0 的 cpu 第三级频率
L_LV3_volt	L_LV3_freq 频点对应的电压
L_LV4_freq	cluster0 的 cpu 第四级频率
L_LV4_volt	L_LV4_freq 频点对应的电压
L_LV5_freq	cluster0 的 cpu 第五级频率
L_LV5_volt	L_LV5_freq 频点对应的电压
L_LV6_freq	cluster0 的 cpu 第六级频率
L_LV6_volt	L_LV6_freq 频点对应的电压
L_LV7_freq	cluster0 的 cpu 第七级频率
L_LV7_volt	L_LV7_freq 频点对应的电压
L_LV8_freq	cluster0 的 cpu 第八级频率
L_LV8_volt	L_LV8_freq 频点对应的电压
B_max_freq	Cluster1 的 cpu 支持的最高频率
B_boot_freq	内核启动阶段 cluster1 的 cpu 频率. 即从 cpufreq 驱动加载到 android 进入主界面之前的 cpu 频率
B_min_freq	Cluster1 的 cpu 支持的最低频率
B_LV_count	Cluster1 的 cpu 频率电压等级数
B_LV1_freq	Cluster1 的 cpu 第一级 (最高) 频率
B_LV1_volt	L_LV1_freq 频点对应的电压

B_LV2_freq	Cluster1 的 cpu 第二级 (次高) 频率
B_LV2_volt	L_LV2_freq 频点对应的电压
B_LV3_freq	Cluster1 的 cpu 第三级频率
B_LV3_volt	L_LV3_freq 频点对应的电压
B_LV4_freq	Cluster1 的 cpu 第四级频率
B_LV4_volt	L_LV4_freq 频点对应的电压
B_LV5_freq	Cluster1 的 cpu 第五级频率
B_LV5_volt	L_LV5_freq 频点对应的电压
B_LV6_freq	Cluster1 的 cpu 第六级频率
B_LV6_volt	L_LV6_freq 频点对应的电压
B_LV7_freq	Cluster1 的 cpu 第七级频率
B_LV7_volt	L_LV7_freq 频点对应的电压
B_LV8_freq	Cluster1 的 cpu 第八级频率
B_LV8_volt	L_LV8_freq 频点对应的电压

注: A83T 为 cluster0 和 cluster1 同频. 也就是八个核任何时候频率都一样, 除非核处于关闭状态.
对 vf_tbale1 和 vf_table2 也一样.

配置示例:

```
[vf_table0]
;little
L_max_freq = 1800000000
L_boot_freq = 1608000000
L_min_freq = 480000000

L_LV_count = 8

L_LV1_freq = 1800000000
L_LV1_volt = 1080

L_LV2_freq = 1608000000
L_LV2_volt = 1000

L_LV3_freq = 1412000000
L_LV3_volt = 920

L_LV4_freq = 1008000000
L_LV4_volt = 840

L_LV5_freq = 0
L_LV5_volt = 840

L_LV6_freq = 0
L_LV6_volt = 840
```

L_LV7_freq = 0
L_LV7_volt = 840

L_LV8_freq = 0
L_LV8_volt = 840

;big
B_max_freq = 1800000000
B_boot_freq = 1608000000
B_min_freq = 480000000

B_LV_count = 8

B_LV1_freq = 1800000000
B_LV1_volt = 1080

B_LV2_freq = 1608000000
B_LV2_volt = 1000

B_LV3_freq = 1412000000
B_LV3_volt = 920

B_LV4_freq = 1008000000
B_LV4_volt = 840

B_LV5_freq = 0
B_LV5_volt = 840

B_LV6_freq = 0
B_LV6_volt = 840

B_LV7_freq = 0
B_LV7_volt = 840

B_LV8_freq = 0
B_LV8_volt = 840

32.1.3. [vf_table1]

对应频率性能为 normal 的 IC.

配置项	配置项含义
L_max_freq	cluster0 的 cpu 支持的最高频率

L_boot_freq	内核启动阶段 cluster0 的 cpu 频率. 即从 cpufreq 驱动加载到 android 进入主界面之前的 cpu 频率
L_min_freq	cluster0 的 cpu 支持的最低频率
L_LV_count	cluster0 的 cpu 频率电压等级数
L_LV1_freq	cluster0 的 cpu 第一级 (最高) 频率
L_LV1_volt	L_LV1_freq 频点对应的电压
L_LV2_freq	cluster0 的 cpu 第二级 (次高) 频率
L_LV2_volt	L_LV2_freq 频点对应的电压
L_LV3_freq	cluster0 的 cpu 第三级频率
L_LV3_volt	L_LV3_freq 频点对应的电压
L_LV4_freq	cluster0 的 cpu 第四级频率
L_LV4_volt	L_LV4_freq 频点对应的电压
L_LV5_freq	cluster0 的 cpu 第五级频率
L_LV5_volt	L_LV5_freq 频点对应的电压
L_LV6_freq	cluster0 的 cpu 第六级频率
L_LV6_volt	L_LV6_freq 频点对应的电压
L_LV7_freq	cluster0 的 cpu 第七级频率
L_LV7_volt	L_LV7_freq 频点对应的电压
L_LV8_freq	cluster0 的 cpu 第八级频率
L_LV8_volt	L_LV8_freq 频点对应的电压
B_max_freq	Cluster1 的 cpu 支持的最高频率
B_boot_freq	内核启动阶段 cluster1 的 cpu 频率. 即从 cpufreq 驱动加载到 android 进入主界面之前的 cpu 频率
B_min_freq	Cluster1 的 cpu 支持的最低频率
B_LV_count	Cluster1 的 cpu 频率电压等级数
B_LV1_freq	Cluster1 的 cpu 第一级 (最高) 频率
B_LV1_volt	L_LV1_freq 频点对应的电压
B_LV2_freq	Cluster1 的 cpu 第二级 (次高) 频率
B_LV2_volt	L_LV2_freq 频点对应的电压
B_LV3_freq	Cluster1 的 cpu 第三级频率
B_LV3_volt	L_LV3_freq 频点对应的电压
B_LV4_freq	Cluster1 的 cpu 第四级频率
B_LV4_volt	L_LV4_freq 频点对应的电压
B_LV5_freq	Cluster1 的 cpu 第五级频率
B_LV5_volt	L_LV5_freq 频点对应的电压
B_LV6_freq	Cluster1 的 cpu 第六级频率
B_LV6_volt	L_LV6_freq 频点对应的电压
B_LV7_freq	Cluster1 的 cpu 第七级频率
B_LV7_volt	L_LV7_freq 频点对应的电压
B_LV8_freq	Cluster1 的 cpu 第八级频率
B_LV8_volt	L_LV8_freq 频点对应的电压

配置示例:

[vf_table1]
;little
L_max_freq = 2016000000
L_min_freq = 480000000

L_LV_count = 8

L_LV1_freq = 2016000000
L_LV1_volt = 1080

L_LV2_freq = 1800000000
L_LV2_volt = 1000

L_LV3_freq = 1608000000
L_LV3_volt = 920

L_LV4_freq = 1200000000
L_LV4_volt = 840

L_LV5_freq = 0
L_LV5_volt = 840

L_LV6_freq = 0
L_LV6_volt = 840

L_LV7_freq = 0
L_LV7_volt = 840

L_LV8_freq = 0
L_LV8_volt = 840

;big
B_max_freq = 2016000000
B_min_freq = 480000000

B_LV_count = 8

B_LV1_freq = 2016000000
B_LV1_volt = 1080

B_LV2_freq = 1800000000
B_LV2_volt = 1000

B_LV3_freq = 1608000000
B_LV3_volt = 920

B_LV4_freq = 1200000000
B_LV4_volt = 840

B_LV5_freq = 0
B_LV5_volt = 840

B_LV6_freq = 0
B_LV6_volt = 840

B_LV7_freq = 0
B_LV7_volt = 840

B_LV8_freq = 0
B_LV8_volt = 840

32.1.4. [vf_table2]

对应频率性能为 fast 的 IC.

配置项	配置项含义
L_max_freq	cluster0 的 cpu 支持的最高频率
L_boot_freq	内核启动阶段 cluster0 的 cpu 频率. 即从 cpufreq 驱动加载到 android 进入主界面之前的 cpu 频率
L_min_freq	cluster0 的 cpu 支持的最低频率
L_LV_count	cluster0 的 cpu 频率电压等级数
L_LV1_freq	cluster0 的 cpu 第一级 (最高) 频率
L_LV1_volt	L_LV1_freq 频点对应的电压
L_LV2_freq	cluster0 的 cpu 第二级 (次高) 频率
L_LV2_volt	L_LV2_freq 频点对应的电压
L_LV3_freq	cluster0 的 cpu 第三级频率
L_LV3_volt	L_LV3_freq 频点对应的电压
L_LV4_freq	cluster0 的 cpu 第四级频率
L_LV4_volt	L_LV4_freq 频点对应的电压
L_LV5_freq	cluster0 的 cpu 第五级频率
L_LV5_volt	L_LV5_freq 频点对应的电压
L_LV6_freq	cluster0 的 cpu 第六级频率
L_LV6_volt	L_LV6_freq 频点对应的电压
L_LV7_freq	cluster0 的 cpu 第七级频率
L_LV7_volt	L_LV7_freq 频点对应的电压
L_LV8_freq	cluster0 的 cpu 第八级频率
L_LV8_volt	L_LV8_freq 频点对应的电压

B_max_freq	Cluster1 的 cpu 支持的最高频率
B_boot_freq	内核启动阶段 cluster1 的 cpu 频率. 即从 cpufreq 驱动加载到 android 进入主界面之前的 cpu 频率
B_min_freq	Cluster1 的 cpu 支持的最低频率
B_LV_count	Cluster1 的 cpu 频率电压等级数
B_LV1_freq	Cluster1 的 cpu 第一级 (最高) 频率
B_LV1_volt	L_LV1_freq 频点对应的电压
B_LV2_freq	Cluster1 的 cpu 第二级 (次高) 频率
B_LV2_volt	L_LV2_freq 频点对应的电压
B_LV3_freq	Cluster1 的 cpu 第三级频率
B_LV3_volt	L_LV3_freq 频点对应的电压
B_LV4_freq	Cluster1 的 cpu 第四级频率
B_LV4_volt	L_LV4_freq 频点对应的电压
B_LV5_freq	Cluster1 的 cpu 第五级频率
B_LV5_volt	L_LV5_freq 频点对应的电压
B_LV6_freq	Cluster1 的 cpu 第六级频率
B_LV6_volt	L_LV6_freq 频点对应的电压
B_LV7_freq	Cluster1 的 cpu 第七级频率
B_LV7_volt	L_LV7_freq 频点对应的电压
B_LV8_freq	Cluster1 的 cpu 第八级频率
B_LV8_volt	L_LV8_freq 频点对应的电压

配置示例:

```
[vf_table2]
;little
L_max_freq = 2016000000
L_min_freq = 480000000

L_LV_count = 8

L_LV1_freq = 2016000000
L_LV1_volt = 1080

L_LV2_freq = 1800000000
L_LV2_volt = 1000

L_LV3_freq = 1608000000
L_LV3_volt = 920

L_LV4_freq = 1200000000
L_LV4_volt = 840

L_LV5_freq = 0
L_LV5_volt = 840
```

L_LV6_freq = 0
L_LV6_volt = 840

L_LV7_freq = 0
L_LV7_volt = 840

L_LV8_freq = 0
L_LV8_volt = 840

;big
B_max_freq = 2016000000
B_min_freq = 480000000

B_LV_count = 8

B_LV1_freq = 2016000000
B_LV1_volt = 1080

B_LV2_freq = 1800000000
B_LV2_volt = 1000

B_LV3_freq = 1608000000
B_LV3_volt = 920

B_LV4_freq = 1200000000
B_LV4_volt = 840

B_LV5_freq = 0
B_LV5_volt = 840

B_LV6_freq = 0
B_LV6_volt = 840

B_LV7_freq = 0
B_LV7_volt = 840

B_LV8_freq = 0
B_LV8_volt = 840

32.2. GPU DVFS

32.2.1. [gpu_dvfs_table]

配置项	配置项含义
G_normal_level	gpu 正常运行的等级.
G_dvfs_enable	gpu dvfs 使能. 1: 使能, 0: 不使能
G_LV_count	gpu VF 表的等级数
G_LV0_freq	第一级 VF 表的频率 (最低)
G_LV0_volt	第一级 VF 表的电压
G_LV1_freq	第二级 VF 表的频率
G_LV1_volt	第二级 VF 表的电压
G_LV2_freq	第三级 VF 表的频率
G_LV2volt	第三级 VF 表的电压
G_LV3_freq	第四级 VF 表的频率
G_LV3_volt	第四级 VF 表的电压
G_LV4_freq	第五级 VF 表的频率
G_LV4_volt	第五级 VF 表的电压
G_LV5_freq	第六级 VF 表的频率
G_LV5_volt	第六级 VF 表的电压
G_LV6_freq	第七级 VF 表的频率
G_LV6_volt	第七级 VF 表的电压
G_LV7_freq	第八级 VF 表的频率
G_LV7_volt	第八级 VF 表的电压
G_LV8_freq	第九级 VF 表的频率 (最高)
G_LV8_volt	第九级 VF 表的电压

android 层根据场景来指定当前设置的等级.

配置示例:

```
[gpu_dvfs_table]
G_normal_level = 4
G_dvfs_enable    = 1
G_LV_count = 9

G_LV0_freq = 252
G_LV0_volt = 700

G_LV1_freq = 288
G_LV1_volt = 740

G_LV2_freq = 456
G_LV2_volt = 800
```

G_LV3_freq = 504
G_LV3_volt = 840

G_LV4_freq = 624
G_LV4_volt = 900

G_LV5_freq = 648
G_LV5_volt = 940

G_LV6_freq = 672
G_LV6_volt = 1000

G_LV7_freq = 696
G_LV7_volt = 1040

G_LV8_freq = 744
G_LV8_volt = 1100

32.3. DRAM DVFS

32.3.1. [dram_scene_table]

配置项	配置项含义
LV_count = 2	LV 数目
LV1_scene = 1	第一个场景的编号
LV1_freq = 456000000	第一个场景的 dram 频率
LV2_scene = 2	第二个场景的编号
LV2_freq = 384000000	第二个场景的 dram 频率

配置举例：

```
;-----------------------------------------------------------------------------
; dram scene frequency table configuration
;
; LV_count: count of LV_scene/LV_freq
;
; LV1: dram frequency default is 456MHz in home
; LV2: dram frequency default is 384MHz in video play
;
;-----------------------------------------------------------------------------
[dram_scene_table]
LV_count = 2
```

LV1_scene = 1
LV1_freq = 456000000

LV2_scene = 2
LV2_freq = 384000000

注: A83T android 层定义的四个场景如下:

场景名称	DRAM 频率
黑屏音乐	168M (最小)
本地视频播放	384M
主界面	456M
旋转/退出 (normal 场景)	732M (最大)

在不同的场景下, android 层通过 sysfs 节点, 通知 dram dvfs 驱动去设置 dram 频率.

33. Pinctrl 测试

配置项	配置项含义
Vdevice_used	作为 pinctrl test 的虚拟设备，为 1 使能
Vdevice_0	虚拟设备的 gpio0 脚设置
Vdevice_1	虚拟设备的 gpio1 脚设置

配置举例：

```
[Vdevice]
Vdevice_used        = 0
Vdevice_0           = port:PH10<5><1><2><default>
Vdevice_1           = port:PH11<5><1><2><default>
```

34. [s_uart0]

配置项	配置项含义
s_uart_used	使能 cpus 的 uart，为 1 使能，为 0 关闭
s_uart_tx	Uart 口发送引脚配置
s_uart_rx	Uart 接收引脚配置

配置举例：

```
[s_uart0]
s_uart_used         = 1
s_uart_tx           = port:PL02<2><default><default><default>
s_uart_rx           = port:PL03<2><default><default><default>
```

35. [s_rsb0]

配置项	配置项含义
s_rsb_used	使能 cpus 使用 rsb 总线，为 1 使能，为 0 关闭
s_rsb_sck	Rsb 时钟引脚设置
s_rsb_sda	Rsb 数据引脚设置

配置举例：

[s_rsb0]

s_rsb_used = 1

s_rsb_sck = port:PL00<2><1><2><default>

s_rsb_sda = port:PL01<2><1><2><default>

36. [s_jtag0]

配置项	配置项含义
s_jtag_used=xx	JTAG 使能
s_jtag_tms=xx	测试模式选择输入(TMS) 的 GPIO 配置
s_jtag_tck=xx	测试时钟输入(TMS) 的 GPIO 配置
s_jtag_tdo=xx	测试数据输出(TDO) 的 GPIO 配置
s_jtag_tdi=xx	测试数据输入（TDI）的 GPIO 配置

配置举例：

[s_jtag0]

s_jtag_used = 0

s_jtag_tms = port:PL04<2><1><2><default>

s_jtag_tck = port:PL05<2><1><2><default>

s_jtag_tdo = port:PL06<2><1><2><default>

s_jtag_tdi = port:PL07<2><1><2><default>

37. [s_powchk]

配置项	配置项含义
s_powchk_used= 0x80000000	是否打开这个功能，如果 bit31 为 1 则使用这个功能，bit0 和 bit1 为 1 分别代表当电源状态和功耗异常时唤醒系统
s_power_reg=0x02309621	电源状态的描述，一般为 1 代表，这里电在休眠时是打开的状态，bit 的具体定义要看 aw_pm.h 里的定义

s_system_power=50	是休眠是允许的最大功耗是 50mW，如果大于此数，代表异常

配置举例：

```
;-----------------------------------------------------------------------------
;s_powchk cpus power check
;s_powchk_used    --power check whether used for arisc in super standby
;    bit31:enable power updat, bit1:wakeup when power state exception
;    bit0:wakeup when power consumption exception
;s_power_reg the expected regs stand for power on/off state
;s_system_power the limit maxmum power consumption when super standby (unit: mw)
;;-----------------------------------------------------------------------------
[s_powchk]
s_powchk_used          = 0x80000000
s_power_reg            = 0x00880010
s_system_power         = 50
```

38. Declaration