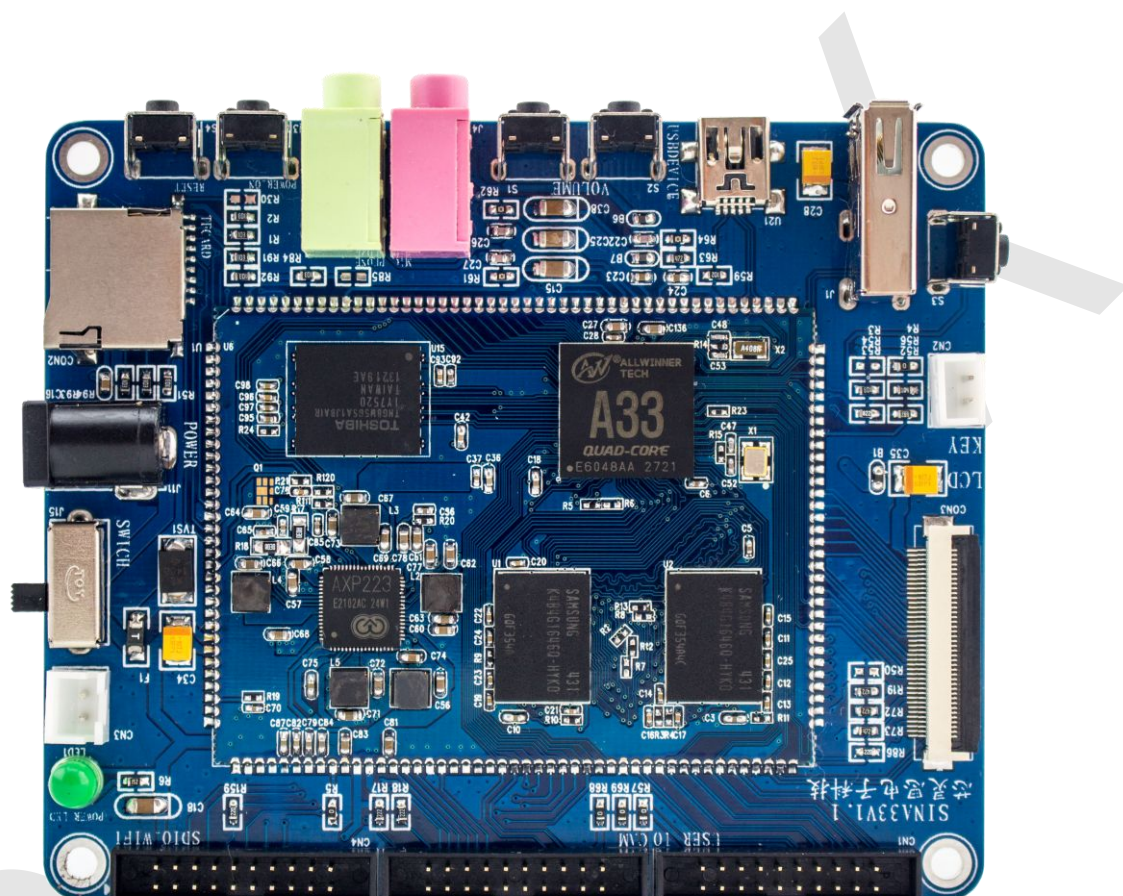


SINA33 用户手册 V1.0



目录

SINA33 用户手册 V1.0.....	1
一、 前言.....	5
二、 烧写指南.....	6
(一) 烧写介绍.....	6
(二) 通过 USB 烧写到 EMMC.....	6
1. 将开发板的 USB OTG 与 PC 相连.....	6
2. 打开烧写软件 PhoenixSuit.....	6
3. 按住开发板的 vol 按键后, 打开电源开关, 快速并多次点击 power 按键, 直到出现下图所示为止.....	7
三、 编译环境搭建.....	8
(一) PC 上的操作系统.....	8
(二) 安装编译环境.....	8
1. 安装相应的库支持.....	8
2. 安装 JAVA6-JDK.....	8
四、 编译 Android 系统.....	10
(一) 解压 Android 源码.....	10
(二) 编译 lichee 目录.....	11
(三) 编译 android 目录.....	12
五、 系统的订制.....	18
(一) 系统配置文件.....	18
(二) 系统配置文件说明.....	18
1. 配置文件说明.....	18
2. 配置文件修改后的编译.....	18
六、 系统(System).....	19
(一) [platform].....	19
(二) [target].....	19
(三) [pm_para].....	19
(四) [card_boot].....	20
(五) [card_boot0_para].....	20
(六) [card_boot2_para].....	20
(七) [twi_para].....	21
(八) [uart_para].....	21
(九) [jtag_para].....	22
(十) [clock].....	22
七、 I2C 总线.....	23
(一) [twi0_para].....	23
(二) [twi1_para].....	23
(三) [twi2_para].....	23
(四) [twi3_para].....	24
八、 串口(UART).....	25
(一) [uart_para0].....	25
(二) [uart_para1].....	25

(三) [uart_para2].....	26
(四) [uart_para3].....	26
(五) [uart_para4].....	27
(六) [uart_para5].....	27
(七) [uart_para6].....	27
(八) [uart_para7].....	28
九、 SPI 总线.....	29
(一) [spi0_para].....	29
(二) [spi1_para].....	29
(三) [spi2_para].....	30
(四) [spi3_para].....	30
(五) [spi_devices].....	30
(六) [spi_board0].....	31
十、 电容屏(capacitor tp).....	32
(一) [ctp_para].....	32
十一、 触摸按键(touch key).....	33
(一) [tkey_para].....	33
十二、 马达(motor).....	34
(一) [motor_para].....	34
十三、 显示初始化(disp init).....	35
(一) [disp_init].....	35
十四、 LCD 屏 0.....	37
(一) [lcd0_para].....	37
十五、 LCD 屏 1.....	41
(一) [lcd1_para].....	41
十六、 HDMI.....	42
(一) [hdmi_para].....	42
十七、 摄像头(CSI).....	43
(一) [csi0_para].....	43
(二) [csi1_para].....	43
十八、 SD / MMC.....	48
(一) [mmc0_para].....	48
(二) [mmc1_para].....	49
(三) [mmc2_para].....	49
(四) [mmc3_para].....	50
十九、 SIM 卡.....	52
(一) [smc_para].....	52
二十、 USB 控制标志.....	53
(一) [usbc0].....	53
(二) [usbc1].....	54
(三) [usbc2].....	55
二十一、 USB Device.....	56
(一) [usb_feature].....	56
(二) [msc_feature].....	56

二十二、 重力感应(G Sensor).....	57
(一) [gsensor_para].....	57
二十三、 WIFI.....	58
(一) [wifi_para].....	58
(二) sdio 接口 wifi rtl8723as demo.....	58
(三) usb 接口 wifi rtl8188eu demo.....	59
二十四、 3G.....	60
(一) [3g_para].....	60
二十五、 gyroscope.....	61
(一) [gy_para].....	61
二十六、 光感(light sensor).....	62
(一) [ls_para].....	62
二十七、 罗盘 Compass.....	63
(一) [compass_para].....	63
二十八、 蓝牙(bluteeth).....	64
(一) [bt_para].....	64
二十九、 数字音频总线 (I2S)	65
(一) [i2s_para].....	65
三十、 数字音频总线(pcm).....	66
(一) [pcm_para].....	66
三十一、 数字音频总线 (S/PDIF)	67
(一) [spdif_para].....	67
三十二、 喇叭控制.....	68
(一) [audio_para].....	68
三十三、 红外(ir).....	69
(一) [ir_para].....	69
三十四、 PMU 电源.....	70
(一) [pmu_para].....	70

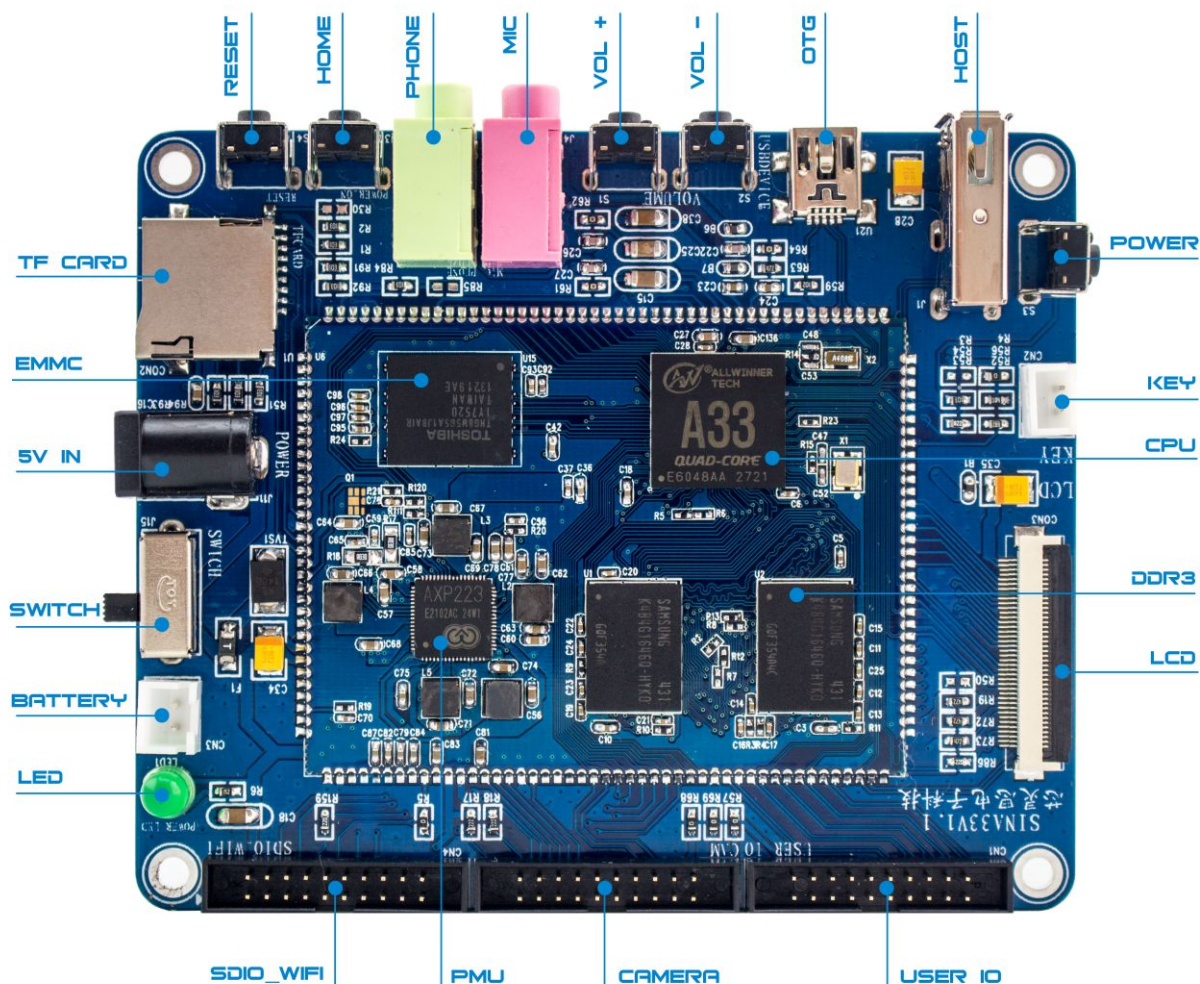
一、前言

A33 是珠海全志科技最新推出的四核处理器。该处理器基于四核 Cortex-A7 CPU 以及 Mali400MP2 GPU 架构，支持 OpenGL ES 2.0 / VG 1.1 standards。

作为全志四核移动应用处理器 A33 新增了对各种无线数据传输技术如 3G, 2G, LTE, WIFI, BT, FM, GPS, AGPS 等的支持。

A33s 处理器具有以下特性:

- 四核 Cortex-A7 CPU 架构;
- 最大支持 1GB 系统内存;
- 配备 Mali400MP2 GPU
- 专为多核 CPU 设计的低功耗 PMU AXP223



A33 主要功能图

二、烧写指南

（一）烧写介绍

SinA33 开发板可以通过 USB OTG 烧写到 EMMC 后启动

在烧写前，请先安装相应软件：

USB 烧写固件到 EMMC：光盘\烧写工具\USB_update_and_produce\PhoenixSuit 工具，请根据您的系统选择安装。

（二）通过 USB 烧写到 EMMC

1. 将开发板的 USB OTG 与 PC 相连
2. 打开烧写软件 PhoenixSuit



选择“一件刷机”

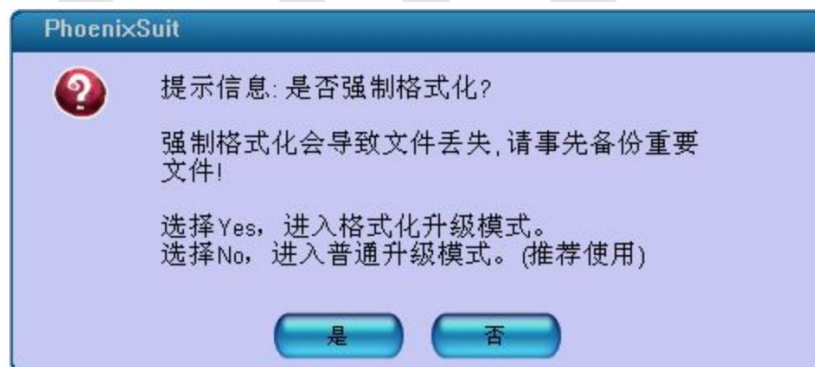


注：图中的镜像名称只是参考，不一定与实际文件名称相符！

在浏览处选择要烧写的固件

光盘固件位置：光盘\烧写固件

3. 按住开发板的 **vol** 按键后，打开电源开关，快速并多次点击 **power** 按键，直到出现下图所示为止



选择“是”后自动进行更新，请耐心等待。

系统更新后，会自动启动。

三、编译环境搭建

（一）PC 上的操作系统

由于 Android4.0 系统编译需求，以及编译时间较长，建议大家使用 64 位的 CPU 及操作系统。

推荐安装 Ubuntu12.04 64 位操作系统。Ubuntu 系统的安装这里不再冗述，如果有不明白的请登陆 Ubuntu 官网，该操作系统也可以在官网中免费下载。

<http://www.ubuntu.com/>

（二）安装编译环境

1. 安装相应的库支持

```
sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev gcc-multilib g++-multilib  
libc6-dev-i386 lib32ncurses5-dev ia32-libs x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev
```

2. 安装 JAVA6-JDK

注：环境变量安装过程中，JDK 的安装比较繁琐，是 Android 开发过程中最困难的，请用户特别注意，这个安装不成功，会导致 Android 编译过程中出现各种奇怪问题。下面提供了三种安装方法，请客户根据自己的实际情况进行选择。建议使用手动安装方式。

（1） `sudo apt-get install sun-java6-jdk`

注：如果你的 Ubuntu 使用了 open-java6-jdk，请手动下载安装 sun-java6-jdk

（2）手动安装 sun-java6-jdk

ORACLE 官网下载地址

<http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html#jdk-6u45-oth-JPR>

这里我们下载的是 jdk-6u45-linux-x64.bin

我们将该文件放置到/home/sinlinx/jdk1.6 目录下

注：sinlinx 为我的工作目录，你可以根据自己的实际目录来创建

增加执行权限 `chmod 777 jdk-6u45-linux-x64.bin`

执行该文件进行安装 `./jdk-6u45-linux-x64.bin`，在安装解压过程中，如有提示 yes/on，请输入 y 回车。

安装完成后，修改环境变量 `bashrc` 或 `profile`

```
export JAVA_HOME=/home/sinlinx/jdk1.6/
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

```
export classPath=/home/sinlinx/jdk1.6/
```

（3）Ubuntu 官网提供的安装方法

添加 partner 源


```
sudo add-apt-repository "deb http://archive.canonical.com/ubuntu maverick partner"
```

更新系统

```
sudo apt-get update
```

安装 jdk

```
sudo apt-get install sun-java6-jdk sun-java6-plugin
```

查看版本信息

```
java -version
```

设置默认 JAVA

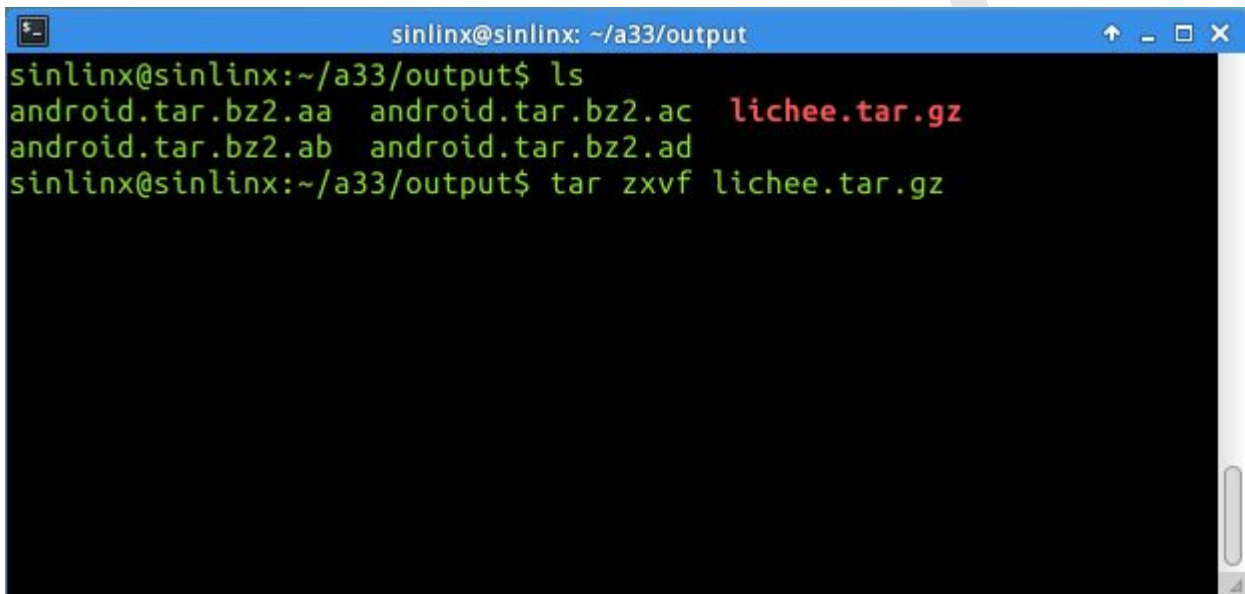
```
sudo update-alternatives --config java
```

四、编译 Android 系统

（一）解压 Android 源码

- 1、lichee: 源码位置: 光盘\源码\lichee.tar.gz
将其拷贝到自己的工作目录下, 进行解压

```
tar zxvf lichee.tar.gz
```



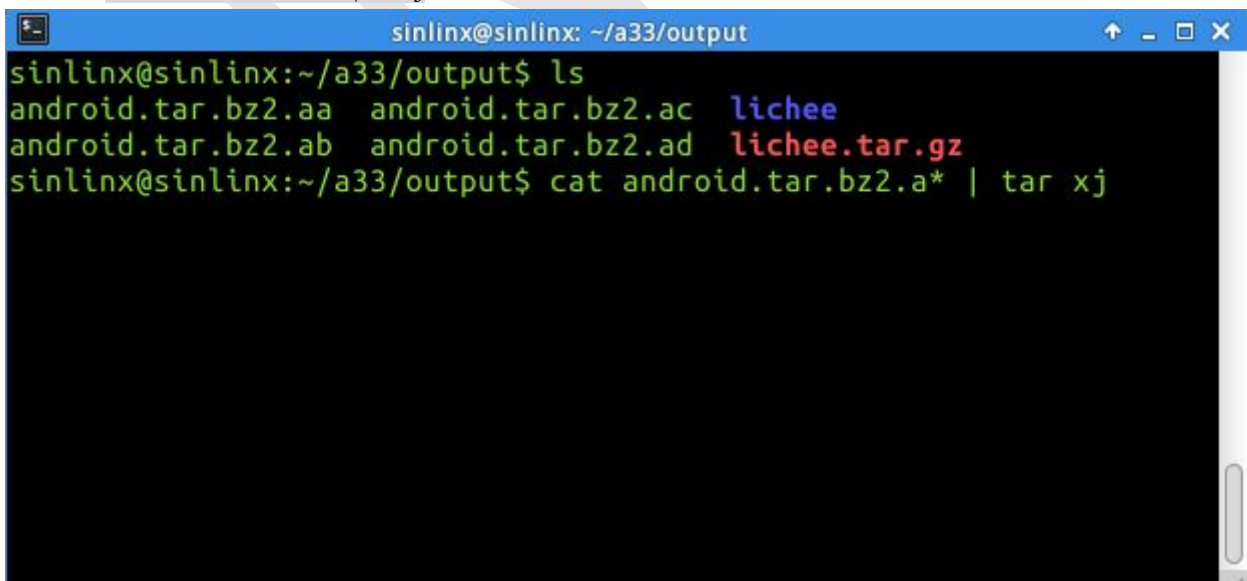
A terminal window titled 'sinlinx@sinlinx: ~/a33/output' showing the following commands and output:

```
sinlinx@sinlinx:~/a33/output$ ls
android.tar.bz2.aa  android.tar.bz2.ac  lichee.tar.gz
android.tar.bz2.ab  android.tar.bz2.ad
sinlinx@sinlinx:~/a33/output$ tar zxvf lichee.tar.gz
```

- 2、Android: 源码位置: 光盘\源码\android.tar.bz2.*

将其拷贝到自己的工作目录下, 进行解压

```
cat android.tar.bz2.a* | tar xj
```



A terminal window titled 'sinlinx@sinlinx: ~/a33/output' showing the following commands and output:

```
sinlinx@sinlinx:~/a33/output$ ls
android.tar.bz2.aa  android.tar.bz2.ac  lichee
android.tar.bz2.ab  android.tar.bz2.ad  lichee.tar.gz
sinlinx@sinlinx:~/a33/output$ cat android.tar.bz2.a* | tar xj
```

- 3、其中 lichee 目录为 uboot、Linux 以及一些脚本配置的源码, android 目录为 android 部分的源码。

这两部分需要分开编译。

```
sinlinx@sinlinx: ~/a33/output
sinlinx@sinlinx:~/a33/output$ ls
android.tar.bz2.aa  android.tar.bz2.ac  lichee
android.tar.bz2.ab  android.tar.bz2.ad  lichee.tar.gz
sinlinx@sinlinx:~/a33/output$ cat android.tar.bz2.a* | tar xj
sinlinx@sinlinx:~/a33/output$ ls
android          android.tar.bz2.ac  lichee.tar.gz
android.tar.bz2.aa  android.tar.bz2.ad
android.tar.bz2.ab  lichee
sinlinx@sinlinx:~/a33/output$
```

(二) 编译 lichee 目录

- 1、进入 lichee 目录

```
cd lichee
```

- 2、第一次编译前，执行配置命令，后续再次编译可省略

```
./build.sh config
```

选择 0 sun8iw5p1 回车

选择 0 android 回车

选择 0 linux-3.4 回车

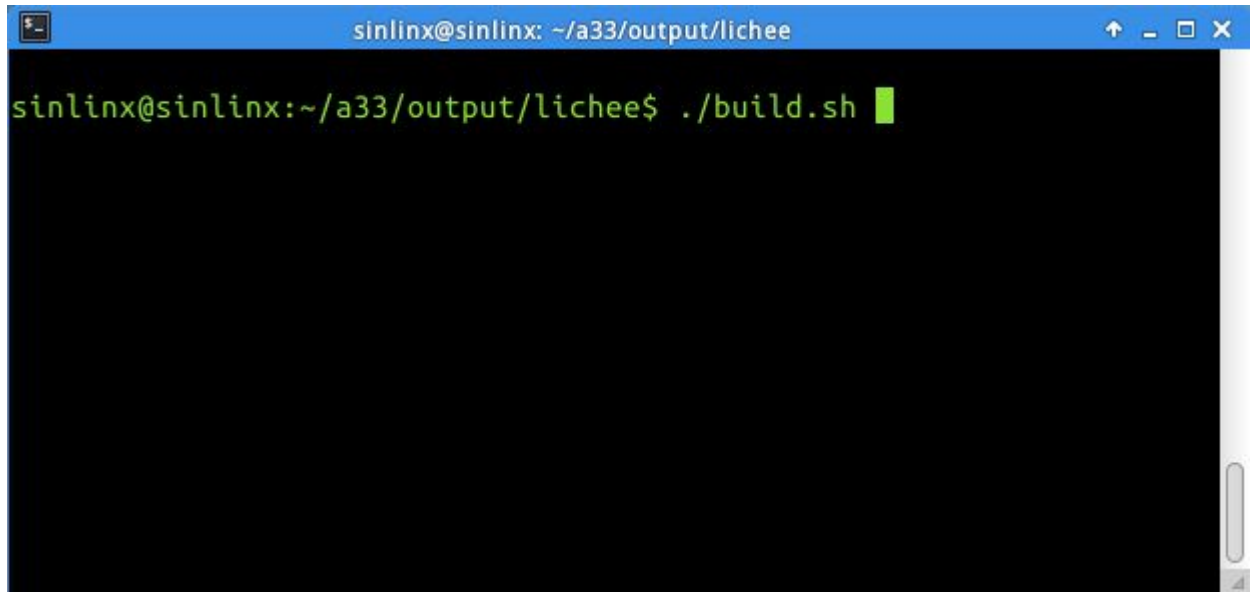
选择 4 y3 回车

```
sinlinx@sinlinx: ~/a33/output/lichee
sinlinx@sinlinx:~/a33/output/lichee$ ./build.sh config

Welcome to mkscrip setup progress
All available chips:
  0. sun8iw5p1
Choice: 0
All available platforms:
  0. android
  1. dragonboard
  2. linux
Choice: 0
All available kernel:
  0. linux-3.4
Choice: 0
All available boards:
  0. evb
  1. maple
  2. redwood
  3. y2
  4. y3
Choice: 4
sinlinx@sinlinx:~/a33/output/lichee$
```

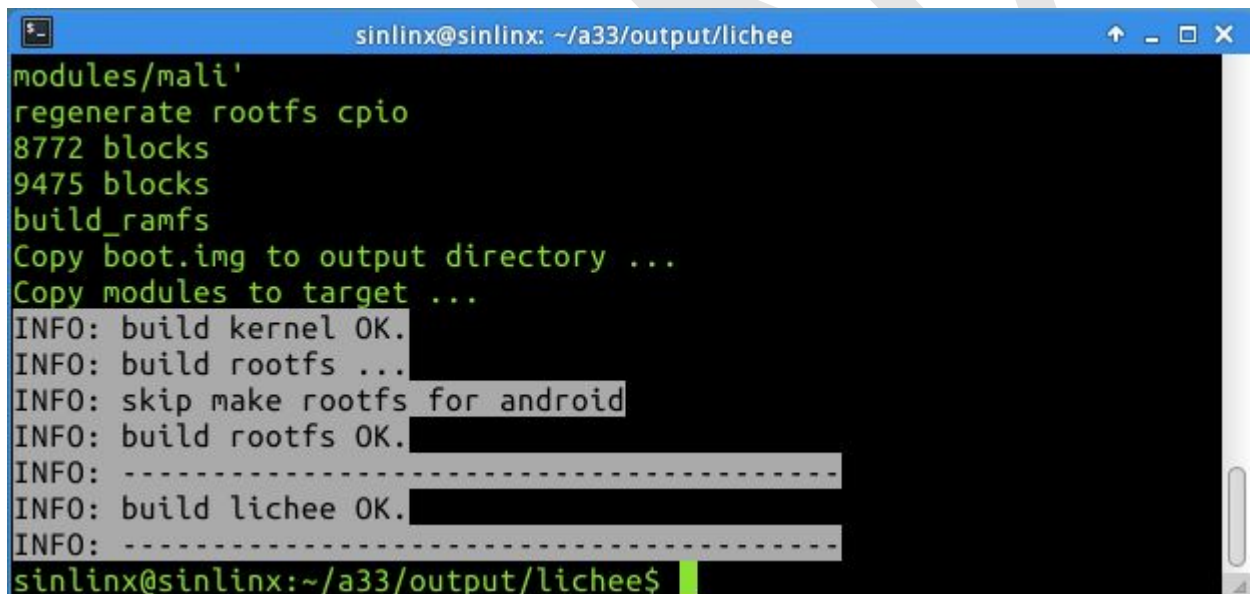
- 3、编译 lichee 目录

```
./build.sh
```



A terminal window titled 'sinlinx@sinlinx: ~/a33/output/lichee'. The prompt is 'sinlinx@sinlinx:~/a33/output/lichee\$' and the command './build.sh' has been entered, with a green cursor at the end of the line.

4、编译完成



A terminal window titled 'sinlinx@sinlinx: ~/a33/output/lichee' showing the output of the build process. The output includes: 'modules/mali'', 'regenerate rootfs cpio', '8772 blocks', '9475 blocks', 'build_ramfs', 'Copy boot.img to output directory ...', 'Copy modules to target ...', 'INFO: build kernel OK.', 'INFO: build rootfs ...', 'INFO: skip make rootfs for android', 'INFO: build rootfs OK.', 'INFO: -----', 'INFO: build lichee OK.', 'INFO: -----', and the prompt 'sinlinx@sinlinx:~/a33/output/lichee\$'.

(三) 编译 android 目录

- 1、进入 android 目录
`cd android`
- 2、设置环境变量
`Source build/envsetup.sh`

```
sinlinx@sinlinx: ~/a33/output/android
sinlinx@sinlinx:~/a33/output/android$ source build/envsetup.sh
```

```
sinlinx@sinlinx: ~/a33/output/android
sinlinx@sinlinx:~/a33/output/android$ source build/envsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/deb/vendorsetup.sh
including device/asus/flo/vendorsetup.sh
including device/softwinner/polaris-common/vendorsetup.sh
including device/softwinner/astar-y3/vendorsetup.sh
including device/samsung/manta/vendorsetup.sh
including device/lge/hammerhead/vendorsetup.sh
including device/lge/mako/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including sdk/bash_completion/adb.bash
sinlinx@sinlinx:~/a33/output/android$
```

3、选择配置方案

Lunch

输入我们选择的配置方案

选择 9 回车


```
sinlinx@sinlinx: ~/a33/output/android
sinlinx@sinlinx:~/a33/output/android$ lunch
```

```
sinlinx@sinlinx: ~/a33/output/android
4. vbox_x86-eng
5. aosp_tilapia-userdebug
6. aosp_grouper-userdebug
7. aosp_deb-userdebug
8. aosp_flo-userdebug
9. astar_y3-eng
10. astar_y3-user
11. aosp_manta-userdebug
12. aosp_hammerhead-userdebug
13. aosp_mako-userdebug
14. mini_x86-userdebug
15. mini_mips-userdebug
16. mini_armv7a_neon-userdebug

Which would you like? [aosp_arm-eng] 9
```

```
sinlinx@sinlinx: ~/a33/output/android
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
TARGET_CPU_VARIANT=cortex-a7
HOST_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-3.13.0-24-generic-x86_64-with-Ubuntu-14.04-trusty
HOST_BUILD_TYPE=release
BUILD_ID=KVT49L
OUT_DIR=out
=====
sinlinx@sinlinx:~/a33/output/android$
```

4、拷贝 lichee 目录下编译好的 uboot 跟 kernel

`extract-bsp`

```
sinlinx@sinlinx: ~/a33/output/android
sinlinx@sinlinx:~/a33/output/android$ extract-bsp
/home/sinlinx/a33/output/android/device/*/astar-y3/bImage copied!
/home/sinlinx/a33/output/android/device/*/astar-y3/modules copied!
sinlinx@sinlinx:~/a33/output/android$
```

5、编译（j8 为 8 线程编译，请根据自己 PC 机器的 CPU 能力选择线程数）

`make -j8`

```
sinlinx@sinlinx: ~/a33/output/android
sinlinx@sinlinx:~/a33/output/android$ make -j8
```

```
sinlinx@sinlinx: ~/a33/output/android
Blocks per group: 32768
Inodes per group: 8192
Inode size: 256
Journal blocks: 3072
Label:
Blocks: 196608
Block groups: 6
Reserved block group size: 47
Created filesystem with 1669/49152 inodes and 143149/196608 blocks
+ '[' 0 -ne 0 ']'
Install system fs image: out/target/product/astar-y3/system.img
out/target/product/astar-y3/system.img+out/target/product/astar-y3/
obj/PACKAGING/recovery_patch_intermediates/recovery_from_boot.p max
size=822163584 blocksize=4224 total=577593476 reserve=8308608
sinlinx@sinlinx:~/a33/output/android$
```

- 6、编译完之后，进行打包
pack

```
sinlinx@sinlinx: ~/a33/output/android
sinlinx@sinlinx:~/a33/output/android$ pack
```

```
sinlinx@sinlinx: ~/a33/output/android
Vboot.fex Len: 0x4
system.fex Len: 0x224a4bdc
Vsystem.fex Len: 0x4
recovery.fex Len: 0xdb3800
Vrecovery.fex Len: 0x4
diskfs.fex Len: 0x200
Vdiskfs.fex Len: 0x4
BuildImg 0
Dragon execute image.cfg SUCCESS !
-----image is at-----

/home/sinlinx/a33/output/lichee/tools/pack/sun8iw5p1_android_y3.img

pack finish
sinlinx@sinlinx:~/a33/output/android$
```

最后生成可以烧写的镜像 sun8iw5p1_android_y3.img

镜像所在目录为 lichee/tools/pack

五、系统的订制

（一）系统配置文件

为了方便用户快速解决由于硬件不同而造成的设备驱动不同的问题，全志科技采用了一个全局配置文件，用户只需要修改系统配置文件，即可解决大部分驱动问题。

该文件的目录为： `lichee/tools/pack/chips/sun8iw5p1/configs/y3` 目录下

其中包括三个文件： `bootlogo.bmp` `sys_config.fex` `sys_partition.fex`

其中 `bootlogo.bmp` 为系统开机时所使用的开机图片

`Sys_config.fex` 就是系统配置文件

`Sys_partition.fex` 为系统的分区配置文件

（二）系统配置文件说明

1. 配置文件说明

备注：蓝色为模块芯片引脚配置，黑色为模块内部控制配置项

描述 `gpio` 的 `GPIO` 配置的形式：

Port:端口+组内序号<功能分配><内部电阻状态><驱动能力><输出电平状态>

2. 配置文件修改后的编译

修改完配置文件之后，只需要执行 `android` 部分的编译即可，其中 `make -j8` 可以省略。

六、系统(System)

(一) [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）

配置举例：

[platform]

eraseflag = 1

(二) [target]

配置项	配置项含义
boot_clock=xx	启动频率; xx 表示多少 MHZ
dc2c1_vol=1400	Dcdc1(IO)的输出电压, mV
dc2c2_vol=1400	Dcdc2(GPU)的输出电压, mV,
dc2c3_vol=1250	Dcdc3(CPU)的输出电压, mV,
Storage_type = -1	启动介质选择 0 : nand, 1: card0,2: card2,-1 (default) 自动扫描启动介质:

配置举例：

[target]

boot_clock = 1008

dc2c1_vol = 300

dc2c2_vol = 1400

dc2c3_vol = 1250

storage_type = -1

(三) [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
```

(四) [card_boot]

配置项	配置项含义
Logical_start=xx	
Sprite_gpio0=	

配置举例：

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

(五) [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：

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

(六) [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=4	4 线卡

<code>sdc_cmd=xx</code>	sdc 命令信号的 GPIO 配置
<code>sdc_clk=xx</code>	sdc 卡时钟信号的 GPIO 配置
<code>sdc_d0=xx</code>	sdc 卡数据 0 线信号的 GPIO 配置
<code>sdc_d1=xx</code>	sdc 卡数据 1 线信号的 GPIO 配置
<code>sdc_d3=xx</code>	sdc 卡数据 3 线信号的 GPIO 配置
<code>sdc_d2=xx</code>	sdc 卡数据 2 线信号的 GPIO 配置

配置举例：

```
card_ctrl                = 2
card_high_speed          = 1
card_line                = 4
sdc_cmd                  = port:PC6<3><1>
sdc_clk                   = port:PC7<3><1>
sdc_d0                    = port:PC8<3><1>
sdc_d1                    = port:PC9<3><1>
sdc_d2                    = port:PC10<3><1>
sdc_d3                    = port:PC11<3><1>
```

（七）[twi_para]

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

配置举例：

```
twi_port                  = 0
twi_scl                   = port:PB0<2><default><default><default>
twi_sda                   = port:PB1<2><default><default><default>
```

（八）[uart_para]

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

配置举例：

```
uart_debug_port           = 0
uart_debug_tx              = port:PB22<2>
uart_debug_rx              = port:PB23<2>
```

(九) [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:PB14<3>
jtag_ck              = port:PB15<3>
jtag_do              = port:PB16<3>
jtag_di              = port:PB17<3>
```

(十) [clock]

配置项	配置项含义
Pl13 =297	Video0 时钟频率
Pl14 =300	Ve 时钟频率
Pl16 =600	Peripherals 时钟频率
Pl17 =297	Video1 时钟频率
Pl18 =360	GPU (通信) 时钟频率
Pl19 =297	GPU (运算) 时钟频率
Pl110 297	De 时钟频率

配置举例:

[clock]

```
pll3          = 297
pll4          = 300
pll6          = 600
pll7          = 297
pll8          = 360
pll9          = 297
pll10         = 297
```

七、I2C 总线

主控有 4 个 I2C (twi) 控制器

(一) [twi0_para]

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

配置举例:

```
twi0_used          = 1
twi0_scl           = port:PH14<2><default><default><default>
twi0_sda           = port:PH15<2><default><default><default>
```

(二) [twi1_para]

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

配置举例:

```
[twi1_para]
twi1_used          = 1
twi1_scl           = port:PH16<2><default><default><default>
twi1_sda           = port:PH17<2><default><default><default>
```

(三) [twi2_para]

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

配置举例:

```
[twi2_para]
twi2_used          = 1
twi2_scl           = port:PH18<2><default><default><default>
twi2_sda           = port:PH19<2><default><default><default>
```


(四) [twi3_para]

配置项	配置项含义
twi3_used =xx	TWI 使用控制: 1 使用, 0 不用
twi3_scl =xx	TWI SCK 的 GPIO 配置
twi3_sda=xx	TWI SDA 的 GPIO 配置

配置举例:

[twi2_para]

twi2_used

= 1

twi2_scl

= port:PB05<4><default><default><default>

twi2_sda

= port:PB06<4><default><default><default>

八、串口(UART)

主控有 6 路 uart 接口，其中 uart1 支持完整的 8 线通讯，而其他 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

（一）[uart_para0]

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
uart0_tx =xx	UART TX 的 GPIO 配置
uart0_rx=xx	UART RX 的 GPIO 配置

配置举例：

[uart_para0]

```
uart_used          = 1
uart_port          = 0
uart0_tx           = port:PB22<2>
uart0_rx           = port:PB23<2>
```

（二）[uart_para1]

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
uart1_tx =xx	UART TX 的 GPIO 配置
uart1_rx =xx	UART RX 的 GPIO 配置
uart1_rts=xx	UART RTS 的 GPIO 配置
uart1_cts=xx	UART CTS 的 GPIO 配置
uart1_dtr=xx	UART DTR 的 GPIO 配置
uart1_dsr=xx	UART DSR 的 GPIO 配置
uart1_dcd=xx	UART DCD 的 GPIO 配置
uart1_ring=xx	UART RING 的 GPIO 配置

配置举例：

[uart_para1]

```
uart_used          = 0
uart_port          = 1
uart_type          = 8
```

```

uart1_tx          = port:PA10<4><default><default><default>
uart1_rx          = port:PA11<4><default><default><default>
uart1_rts         = port:PA12<4><default><default><default>
uart1_cts         = port:PA13<4><default><default><default>
uart1_dtr         = port:PA14<4><default><default><default>
uart1_dsr         = port:PA15<4><default><default><default>
uart1_dcd         = port:PA16<4><default><default><default>
uart1_ring        = port:PA17<4><default><default><default>

```

(三) [uart_para2]

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
uart2_tx =xx	UART TX 的 GPIO 配置
uart2_rx =xx	UART RX 的 GPIO 配置
uart2_rts =xx	UART RTS 的 GPIO 配置
uart2_cts =xx	UART CTS 的 GPIO 配置

配置举例：

```

[uart_para2]
uart_used          = 0
uart_port          = 2
uart_type          = 4
uart2_tx           = port:PI18<3><default><default><default>
uart2_rx           = port:PI19<3><default><default><default>
uart2_rts          = port:PI16<3><default><default><default>
uart2_cts          = port:PI17<3><default><default><default>

```

(四) [uart_para3]

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
uart3_tx =xx	UART TX 的 GPIO 配置
uart3_rx =xx	UART RX 的 GPIO 配置
uart3_rts =xx	UART RTS 的 GPIO 配置
uart3_cts =xx	UART CTS 的 GPIO 配置

配置举例：

```

[uart_para3]
uart_used          = 0
uart_port          = 3

```

```
uart_type          = 4
uart3_tx           = port:PH00<4><default><default><default>
uart3_rx           = port:PH01<4><default><default><default>
uart3_rts          = port:PH02<4><default><default><default>
uart3_cts          = port:PH03<4><default><default><default>
```

(五) [uart_para4]

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

配置举例：

```
[uart_para4]
uart_used          = 0
uart_port          = 4
uart_type          = 2
uart4_tx           = port:PH04<4><default><default><default>
uart4_rx           = port:PH05<4><default><default><default>
```

(六) [uart_para5]

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
uart5_tx =xx	UART TX 的 GPIO 配置
uart5_rx =xx	UART RX 的 GPIO 配置

配置举例：

```
[uart_para5]
uart_used          = 0
uart_port          = 5
uart_type          = 2
uart5_tx           = port:PH06<4><default><default><default>
uart5_rx           = port:PH07<4><default><default><default>
```

(七) [uart_para6]

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
uart6_tx =xx	UART TX 的 GPIO 配置
uart6_rx =xx	UART RX 的 GPIO 配置

配置举例：

[uart_para6]

uart_used = 0

uart_port = 6

uart_type = 2

uart6_tx = port:PA12<4><default><default><default>

uart6_rx = port:PA13<4><default><default><default>

（八）[uart_para7]

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
uart7_tx =xx	UART TX 的 GPIO 配置
uart7_rx =xx	UART RX 的 GPIO 配置

配置举例：

[uart_para7]

uart_used = 0

uart_port = 7

uart_type = 2

uart7_tx = port:PA14<4><default><default><default>

uart7_rx = port:PA15<4><default><default><default>

九、SPI 总线

(一) [spi0_para]

配置项	配置项含义
spi_used =xx	SPI 使用控制：1 使用，0 不用
spi_cs0 =xx	SPI CS0 的 GPIO 配置
spi_cs1 =xx	SPI CS1 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

配置举例：

[spi0_para]

spi_used = 0

spi_cs_bitmap = 1

;--- spi0 mapping0 ---

spi_cs0 = port:PI10<3><default><default><default>

;spi_cs1 = port:PI14<3><default><default><default>

spi_sclk = port:PI11<3><default><default><default>

spi_mosi = port:PI12<3><default><default><default>

spi_miso = port:PI13<3><default><default><default>

(二) [spi1_para]

配置项	配置项含义
spi_used =xx	SPI 使用控制：1 使用，0 不用
spi_cs0 =xx	SPI CS0 的 GPIO 配置
spi_cs1 =xx	SPI CS1 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

配置举例：

[spi1_para]

spi_used = 0

spi_cs_bitmap = 1

spi_cs0 = port:PA00<4><default><default><default>

spi_sclk = port:PA01<4><default><default><default>

spi_mosi = port:PA02<4><default><default><default>

spi_miso = port:PA03<4><default><default><default>

(三) [spi2_para]

配置项	配置项含义
spi_used =xx	SPI 使用控制：1 使用，0 不用
spi_cs0 =xx	SPI CS0 的 GPIO 配置
spi_cs1 =xx	SPI CS1 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

配置举例：

```
spi_used                = 0
spi_cs_bitmap           = 1
spi_cs0                 = port:PB14<2><default><default><default>
spi_sclk                = port:PB15<2><default><default><default>
spi_mosi                = port:PB16<2><default><default><default>
spi_miso                = port:PB17<2><default><default><default>
```

(四) [spi3_para]

配置项	配置项含义
spi_used =xx	SPI 使用控制：1 使用，0 不用
spi_cs0 =xx	SPI CS0 的 GPIO 配置
spi_cs1 =xx	SPI CS1 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

配置举例：

```
[spi3_para]
spi_used                = 0
spi_cs_bitmap           = 1
;--- spi3 mapping0 ---
spi_cs0                 = port:PA05<3><default><default><default>
spi_sclk                = port:PI06<3><default><default><default>
spi_mosi                = port:PI07<3><default><default><default>
spi_miso                = port:PI08<3><default><default><default>
spi_cs1                 = port:PA09<3><default><default><default>
```

(五) [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
--	---

(六) [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

十、电容屏(capacitor tp)

(一) [ctp_para]

配置项	配置项含义
ctp_used=xx	该选项为是否开启电容触摸，支持的话置 1，反之置 0
ctp_twi_id=xx	用于选择 i2c adapter, 可选 1, 2
ctp_twi_addr=xx	指明 i2c 设备地址，与具体硬件相关
ctp_screen_max_x=xx	触摸板的 x 轴最大坐标
ctp_screen_max_y=xx	触摸板的 y 轴最大坐标
ctp_revert_x_flag=xx	是否需要翻转 x 坐标，需要则置 1，反之置 0
ctp_revert_y_flag=xx	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=xx	电容屏中断信号的 GPIO 配置
ctp_wakeup=xx	电容屏唤醒信号的 GPIO 配置

配置举例：

[ctp_para]

```

ctp_used                = 1
ctp_twi_id              = 1
ctp_twi_addr            = 0x5d
ctp_screen_max_x        = 1280
ctp_screen_max_y        = 800
ctp_revert_x_flag       = 1
ctp_revert_y_flag       = 1
ctp_exchange_x_y_flag   = 1
ctp_int_port            = port:PA03<6><default><default><default>
ctp_wakeup              = port:PA02<1><default><default><1>

```

注意事项：

若要支持新的电容触控 ic，在原有电容触控 ic 的代码基础上，须结合 A31 bsp 层的配置情况，作相应修改。具体说来，

1. 在 sys_config 中：ctp_twi_id 应与硬件连接一致；
2. 在驱动部分代码中：sysconfig 中的其他子键也要正确配置，在程序中，要对这些配置进行相应的处理；

十一、触摸按键(touch key)

(一) [tkey_para]

配置项	配置项含义
tkey_used=xx	支持触摸按键的置 1，反之置 0
tkey_twi_id=xx	用于选择 i2c adapter, 可选 1, 2
tkey_twi_addr=xx	指明 i2c 设备地址，与具体硬件相关
tkey_int=xx	触摸按键中断信号的 GPIO 配置

配置举例：

```
tkey_used          = 0
tkey_twi_id        = 2
tkey_twi_addr      = 0x62
tkey_int           = port:PI13<6><default><default><default>
```

注意事项：

若支持，则将 tkey_used 置 1 并配置相应子键值；否则，tkey_used 置 0；

十二、马达(motor)

(一) [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

配置举例：

motor_used = 1

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

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。

十三、显示初始化(disp init)

(一) [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:hdm; 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:hdm; 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
lcd0_bright	Lcd0 的亮度值, 0~100
lcd0_contrast	Lcd0 的对比度, 0~100

lcd0_saturation	Lcd0 的饱和度, 0~100
lcd0_hue	Lcd0 的色度, 0~100
lcd1_bright	Lcd1 的亮度值, 0~100
lcd1_contrast	Lcd1 的对比度, 0~100
lcd1_saturation	Lcd1 的饱和度, 0~100
lcd1_hue	Lcd1 的色度, 0~100

配置举例:

[disp_init]

```

disp_init_enable      = 1
disp_mode             = 0
screen0_output_type   = 1
screen0_output_mode   = 4
screen1_output_type   = 1
screen1_output_mode   = 4
fb0_format            = 10
fb0_pixel_sequence    = 0
fb0_scaler_mode_enable= 0
fb0_width             = 0
fb0_height            = 0
fb1_format            = 10
fb1_pixel_sequence    = 0
fb1_scaler_mode_enable= 0
fb1_width             = 0
fb1_height            = 0
lcd0_backlight        = 197
lcd1_backlight        = 197
lcd0_bright           = 50
lcd0_contrast         = 50
lcd0_saturation       = 57
lcd0_hue              = 50
lcd1_bright           = 50
lcd1_contrast         = 50
lcd1_saturation       = 57
lcd1_hue              = 50

```

十四、LCD 屏 0

(一) [lcd0_para]

配置项	配置项含义
lcd_used=xx	是否使用 lcd0
lcd_if=xx	lcd interface(0:lv(sync+de); 1:8080; 2:ttl; 3:lvds; 4:dsi; 5:edp)
lcd_x=xx	lcd active width
lcd_y=xx	lcd active height
lcd_dclk_freq=xx	pixel clock, in MHZ unit
lcd_pwm_freq=xx	pwm freq, in HZ unit
lcd_pwm_pol=xx	pwm polarity, 0:positive; 1:negative
lcd_pwm_max_limit=xx	Lcd pwm max limit(<=255)
lcd_hbp=xx	hsync back porch
lcd_ht=xx	hsync total cycle
lcd_vbp=xx	vsync back porch
lcd_vt=xx	vsync total cycle
lcd_hv_vspw=xx	vsync plus width
lcd_hv_hspw=xx	hsync plus width
lcd_hv_if=xx	hv interface(0:parallel; 8:serial(8bit/3cycle); 10:dummysrgb(8bit/4cycle); 11:rgbdummy(8bit/4cycle); 12:ccir656)
lcd_hv_srgb_seq=xx	serial RGB output sequence
lcd_hv_syuv_seq=xx	serial YUV output sequence
lcd_hv_syuv_fdly	serial YUV output F line delay(0: no delay; 1: delay 2line[CCIR NTSC]; 2: delay 3line[CCIR PAL])
lcd_lvds_if=xx	0:single channel; 1:dual channel
lcd_lvds_colordepth=xx	0:8bit; 1:6bit
lcd_lvds_mode=xx	0:NS mode; 1:JEIDA mode
lcd_lvds_io_polarity=xx	0:normal; 1:pn cross
lcd_dsi_if=xx	0:video mode; 1:command mode
lcd_dsi_lane=xx	1/2/3/4lane
lcd_dsi_format=xx	0:RGB888; 1:RGB666; 2:RGB666P; 3:RGB565
lcd_dsi_eotp=xx	0:no ending symbol 1:insert ending symbol;
lcd_dsi_te=xx	0:disable te mode; 1:rising te mode; 2:falling te mode
lcd_cpu_if=xx	cpu i/f mode(0:18bit; 1:16bit mode0; 2:16bit

	mode1; 3:16bit mode2;4:16bit mode3; 5:9bit; 6:8bit 256K; 7:8bit 65K;)
lcd_cpu_te=xx	0:disable te mode; 1:enable rising te mode; 2:enable falling te mode
lcd_frm=xx	0:disable; 1:enable rgb666 dither; 2:enable rgb656 dither
lcd_edp_tx_ic=xx	0:anx9804; 1:anx6345
lcd_edp_tx_rate=xx	1:1.62G; 2:2.7G; 3:5.4G
lcd_edp_tx_lane=xx	1/2/4lane
lcd_io_phase=xx	0:noraml; 1:intert phase(0~3bit: vsync phase; 4~7bit:hsync phase;8~11bit:dclk phase; 12~15bit:de phase)
deu_mode=xx	Parameter for deu. 0:smoll lcd screen; 1:large lcd screen(larger than 10inch)
lcdgamma4iep=xx	Smart Backlight parameter, lcd gamma vale * 10;
smart_color=xx	90:normal lcd screen 65:retina lcd screen(9.7inch) (0~100)
lcd_bl_en=xx	LCD_BL_EN 的 GPIO 配置
lcd_power=xx	LCD_VCC control 的 GPIO 配置
lcd_pwm=xx	lcd PWM 的 GPIO 配置 (PWM0 固定使用 PB02, PWM1 固定使用 PI03,用户无需修改该项)
lcd_gpio_scl	iic SCL
lcd_gpio_sda	iic SDA
lcd_gpio_0/1/2/3=xx	LCD 额外需要使用的 GPIO 配置
lcd0~23=xx	lcd 数据的 GPIO 配置
lcdclk=xx	lcd 信号的 GPIO 配置(具体信号与实际电路相关)
lcdde=xx	lcd 信号的 GPIO 配置(具体信号与实际电路相关)
lcdhsync=xx	lcd 信号的 GPIO 配置(具体信号与实际电路相关)
lcdvsync=xx	lcd 信号的 GPIO 配置(具体信号与实际电路相关)

配置举例:

[lcd0_para]

```

lcd_used          = 1
lcd_if            = 0
lcd_x             = 1280
lcd_y             = 800

```

lcd_dclk_freq = 70
 lcd_pwm_freq = 50000
 lcd_pwm_pol = 0
 lcd_pwm_max_limit = 150
 lcd_hbp = 20
 lcd_ht = 1418
 lcd_hspw = 10
 lcd_vbp = 10
 lcd_vt = 814
 lcd_vspw = 5
 lcd_hv_if = 0
 lcd_hv_srgb_seq = 0
 lcd_hv_syuv_seq = 0
 lcd_hv_syuv_fdly = 0
 lcd_lvds_if = 0
 lcd_lvds_colordepth = 1
 lcd_lvds_mode = 0
 lcd_lvds_io_polarity = 0
 lcd_dsi_if = 0
 lcd_dsi_lane = 0
 lcd_dsi_format = 0
 lcd_dsi_eotp = 0
 lcd_dsi_te = 0
 lcd_cpu_if = 0
 lcd_cpu_te = 0
 lcd_frm = 1
 lcd_edp_tx_ic = 0
 lcd_edp_tx_rate = 0
 lcd_edp_tx_lane = 0
 lcd_io_phase = 0x00
 deu_mode = 0
 lcdgamma4iep = 22
 Smart_color = 90
 lcd_bl_en = port:PA25<1><0><default><1>
 lcd_power = port:power2<1><0><default><1>
 lcd_pwm = port:PH13<2><0><default><default>
 lcd_gpio_scl =
 lcd_gpio_sda =
 lcd_gpio_0 =
 lcd_gpio_1 =
 lcd_gpio_2 =
 lcd_gpio_3 =
 lcdd0 = port:PD00<2><0><default><default>
 lcdd1 = port:PD01<2><0><default><default>

lcdd2	= port:PD02<2><0><default><default>
lcdd3	= port:PD03<2><0><default><default>
lcdd4	= port:PD04<2><0><default><default>
lcdd5	= port:PD05<2><0><default><default>
lcdd6	= port:PD06<2><0><default><default>
lcdd7	= port:PD07<2><0><default><default>
lcdd8	= port:PD08<2><0><default><default>
lcdd9	= port:PD09<2><0><default><default>
lcdd10	= port:PD10<2><0><default><default>
lcdd11	= port:PD11<2><0><default><default>
lcdd12	= port:PD12<2><0><default><default>
lcdd13	= port:PD13<2><0><default><default>
lcdd14	= port:PD14<2><0><default><default>
lcdd15	= port:PD15<2><0><default><default>
lcdd16	= port:PD16<2><0><default><default>
lcdd17	= port:PD17<2><0><default><default>
lcdd18	= port:PD18<2><0><default><default>
lcdd19	= port:PD19<2><0><default><default>
lcdd20	= port:PD20<2><0><default><default>
lcdd21	= port:PD21<2><0><default><default>
lcdd22	= port:PD22<2><0><default><default>
lcdd23	= port:PD23<2><0><default><default>
lcdclk	= port:PD24<2><0><default><default>
lcdde	= port:PD25<2><0><default><default>
lcdhsync	= port:PD26<2><0><default><default>
lcdvsync	= port:PD27<2><0><default><default>

十五、LCD 屏 1

（一）[lcd1_para]

所有配置跟 lcd0 一样

十六、HDMI

(一) [hdmi_para]

配置项	配置项含义
para_used =xx	是否使用 hdmi

十七、摄像头(CSI)

(一) [csi0_para]

留空，不要填写，如下：

[csi0_para]

csi_used = 0

(二) [csi1_para]

特别注意事项：

在 A31 以及后续项目中（因为内核对 GPIO 资源的管理有修改），如果两个 sensor 制作 2 合 1 模组的时候**请注意将两个模组的 reset 控制脚分开(包括)，stby 控制脚也分开**，仅有电源，数据线、clock 线、地可以复用。

如果是使用 RAW 格式的 sensor，硬件上需要 CSI_D[11:2]共 10 条数据线，请不要将 CSI_D3 和 CSI_D2 用做 GPIO 功能，模组上的 D[3:2]也要注意从 sensor 端引出来。

配置项	配置项含义
csi_used =xx	是否使用 csi1
csi_twi_id =xx	csi 使用的 IIC 通道序号，查看具体方案原理图，使用 twi0 填 0
csi_mname=xx	csi 使用的模组名称，需要与驱动匹配，可以查看驱动目录里面的 readme 目前有 gc0307, gc0308, gc2035, gt2005, hi253, ov5640, s5k4ec 可选
csi_twi_addr=xx	csi 使用的模组的 IIC 地址（8bit 地址），可以查看驱动目录里面的 readme
csi_if	配置目前使用模组的接口时序： 0:8bit 数据线，带 Hsync,Vsync 1:16bit 数据线，带 Hsync,Vsync 2:24bit 数据线，带 Hsync,Vsync 3:8bit 数据线,BT656 内嵌同步,单通道 4:8bit 数据线,BT656 内嵌同步,双通道

	5:8bit 数据线,BT656 内嵌同步,四通道
csi_mode	配置 csi 接收 buffer 的模式: 0: 一个 CSI 接收对应一个 buffer 1: 两个 CSI 接收内容拼接成一个 buffer
csi_dev_qty	配置 csi 目前连接的器件数量,目前只能配置为 1 或 2
csi_vflip	配置 csi 接收图像默认情况下, 上下颠倒情况: 0: 正常 1: 上下颠倒
csi_hflip	配置 csi 接收图像默认情况下, 左右颠倒情况: 0: 正常 1: 左右颠倒
csi_stby_mode	配置 csi 在进入 standby 时的处理: 0: 不关闭电源, 只拉 standby io 1: 关闭电源, 同时拉 standby io
csi_iovdd	配置 csi iovdd 电源来源: 请查看对应方案原理图, 一般填写的名字为"axp22_XldoN"等(注意带英文字符的双引号, 不使用 axp 电源供电时候请务必留空引号") 如 EVB 上, 配置成"axp22_eldo3"
csi_avdd	配置 csi avdd 电源来源: 请查看对应方案原理图, 一般填写的名字为"axp22_XldoN"等(注意带英文字符的双引号, 不使用 axp 电源供电时候请务必留空引号"), 这个地方请特别注意, 因为此电源对于 sensor 图像质量关系较大, 对于高像素 sensor 建议使用 axp22_ldoio0 或 axp22_ldoio1 这两组电源或者采用外挂带 EN 控制的 LDO
csi_dvdd	配置 csi dvdd 电源来源: 请查看对应方案原理图, 一般填写的名字为"axp22_XldoN"等(注意带英文字符的双引号, 不使用 axp 电源供电时候请务必留空引号")
csi_vol_iovdd	配置 csi iovdd 电源电压 如果 csi_iovdd 配置不为空时会配置对应的 axp 电源为相应电压 配置为 2800 表示 2.8V, 范围不要超过 1800~2800, 请查看具体 sensor 的 datasheet 填写此电压

csi_vol_avdd	配置 csi avdd 电源电压 如果 csi_avdd 配置不为空时会配置对应的 axp 电源为相应电压 配置为 2800 表示 2.8V ，一般不要修改此数值
csi_vol_dvdd	配置 csi dvdd 电源电压 如果 csi_dvdd 配置不为空时会配置对应的 axp 电源为相应电压 配置为 1500 表示 1.5V ，范围不要超过 1200~1800 ，请查看具体 sensor 的 datasheet 填写此电压
csi_pck=xx	模组送给 csi 的 clock 的 GPIO 配置
csi_ck=xx	csi 送给模组的 clock 的 GPIO 配置
csi_hsync=xx	模组送给 csi 的行同步信号 GPIO 配置
csi_vsync=xx	模组送给 csi 的帧同步信号 GPIO 配置
csi_d0=xx ... csi_d23=xx	模组送给 csi 的 8bit/16bit/24bit 数据的 GPIO 配置，使用 YUV 格式的 sensor 方案中，csi_d0/d1/d2/d3 会被配置成普通 GPIO，用来控制 sensor 的 pwn/reset 信号，使用 RAW 格式的 sensor 只能用 csi_d0/d1 作 GPIO 用途。
csi_reset=xx	控制模组的 reset 的 GPIO 配置，默认值为 reset 有效（高或低有效需要取决于模组）
csi_power_en=xx	控制模组的电源的 GPIO 配置，若 csi_stby_mode 配置成 0，则 csi_power_en 的默认值一般配置成 1；若 csi_stby_mode 配置成 1，则 csi_power_en 的默认值一般配置成 0。
csi_stby=xx	控制模组的 standby 的 GPIO 配置，默认值为 standby 有效（高或低有效需要取决于模组）
csi_reset_b=xx	如果有两个模组同时连接到一个 CSI，需要额外的 IO 控制；控制模组的 reset 的 GPIO 配置，默认值为 reset 有效（高或低有效需要取决于模组）
csi_power_en_b=xx	如果有两个模组同时连接到一个 CSI，需要额外的 IO 控制；控制模组的电源的 GPIO 配置，若 csi_stby_mode 配置成 0，则 csi_power_en 的默认值一般配置成 1；若 csi_stby_mode 配置成 1，则 csi_power_en 的默认值一般配置成 0。
csi_stby_b=xx	如果有两个模组同时连接到一个 CSI，需要额外的 IO 控制；控制模组的 standby 的 GPIO 配置，默认值为 standby 有效（高或低有效需要取决于模组）

配置举例：

[csi1_para]

csi_used = 1
csi_mode = 0
csi_dev_qty = 2
csi_stby_mode = 0

csi_mname = "ov5640"
csi_twi_id = 0
csi_twi_addr = 0x78
csi_if = 0
csi_vflip = 0
csi_hflip = 1
csi_iovdd = "axp22_eldo3"
csi_avdd = "axp22_dldo4"
csi_dvdd = "axp22_eldo2"
csi_vol_iovdd = 2800
csi_vol_avdd = 2800
csi_vol_dvdd = 1800
csi_flash_pol = 1

csi_mname_b = "gc0307"
csi_twi_id_b = 0
csi_twi_addr_b = 0x42
csi_if_b = 0
csi_vflip_b = 1
csi_hflip_b = 1
csi_iovdd_b = "axp22_eldo3"
csi_avdd_b = "axp22_dldo4"
csi_dvdd_b = "axp22_eldo2"
csi_vol_iovdd_b = 2800
csi_vol_avdd_b = 2800
csi_vol_dvdd_b = 1800
csi_flash_pol_b = 1

csi_pck = port:PE00<2><default><default><default>
csi_mck = port:PE01<2><default><default><default>
csi_hsync = port:PE02<2><default><default><default>
csi_vsync = port:PE03<2><default><default><default>
csi_d0 =
csi_d1 =
csi_d2 =
csi_d3 =
csi_d4 = port:PE08<2><default><default><default>

csi_d5	= port:PE09<2><default><default><default>
csi_d6	= port:PE10<2><default><default><default>
csi_d7	= port:PE11<2><default><default><default>
csi_d8	= port:PE12<2><default><default><default>
csi_d9	= port:PE13<2><default><default><default>
csi_d10	= port:PE14<2><default><default><default>
csi_d11	= port:PE15<2><default><default><default>
csi_reset	= port:PE04<1><default><default><0>
csi_power_en	=
csi_stby	= port:PE05<1><default><default><1>
csi_flash	=
csi_af_en	=
csi_reset_b	= port:PE06<1><default><default><0>
csi_power_en_b	=
csi_stby_b	= port:PE07<1><default><default><1>
csi_flash_b	=
csi_af_en_b	=

十八、SD / MMC

(一) [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 检测，2-data3 检测，3-无检测，卡常在(不卡拔插)，4 - manual mode(from proc file system node)
bus_width=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"

配置举例：

```
[mmc0_para]
sdc_used          = 1
sdc_detmode       = 1
bus_width         = 4
sdc_d1            = port:PF0<2><1><default><default>
sdc_d0            = port:PF1<2><1><default><default>
sdc_clk           = port:PF2<2><1><default><default>
sdc_cmd           = port:PF3<2><1><default><default>
sdc_d3            = port:PF4<2><1><default><default>
sdc_d2            = port:PF5<2><1><default><default>
sdc_det           = port:PH1<0><1><default><default>
sdc_use_wp        = 0
sdc_wp            =
```

(二) [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 检测, 2-data3 检测, 3-无检测, 卡常在(不卡拔插), 4 - manual mode(from proc file system node)
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

配置举例:

```
[mmc1_para]
sdc_used          = 1
sdc_detmode       = 1
bus_width         = 4
sdc_cmd           = port:PH22<5><1><default><default>
sdc_clk           = port:PH23<5><1><default><default>
sdc_d0            = port:PH24<5><1><default><default>
sdc_d1            = port:PH25<5><1><default><default>
sdc_d2            = port:PH26<5><1><default><default>
sdc_d3            = port:PH27<5><1><default><default>
sdc_det           = port:PH2<0><1><default><default>
sdc_use_wp        = 0
sdc_wp            =
```

(三) [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 检测, 2-data3 检测, 3-无检测, 卡常在(不卡拔插), 4 - manual mode(from proc file system node)
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

配置举例:

[mmc2_para]

```

sdc_used                = 1
sdc_detmode             = 1
bus_width               = 4
sdc_cmd                 = port:PH22<5><1><default><default>
sdc_clk                 = port:PH23<5><1><default><default>
sdc_d0                  = port:PH24<5><1><default><default>
sdc_d1                  = port:PH25<5><1><default><default>
sdc_d2                  = port:PH26<5><1><default><default>
sdc_d3                  = port:PH27<5><1><default><default>
sdc_det                 = port:PH2<0><1><default><default>
sdc_use_wp               = 0
sdc_wp                  =

```

(四) [mmc3_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 检测, 2-data3 检测, 3-无检测, 卡常在(不卡拔插), 4 - manual mode(from proc file system node)
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

配置举例:

[mmc3_para]

```

sdc_used                = 1
sdc_detmode             = 1

```

bus_width = 4
sdc_cmd = port:PH22<5><1><default><default>
sdc_clk = port:PH23<5><1><default><default>
sdc_d0 = port:PH24<5><1><default><default>
sdc_d1 = port:PH25<5><1><default><default>
sdc_d2 = port:PH26<5><1><default><default>
sdc_d3 = port:PH27<5><1><default><default>
sdc_det = port:PH2<0><1><default><default>
sdc_use_wp = 0
sdc_wp =

十九、SIM 卡

(一) [smc_para]

配置项	配置项含义
smc_used =xx	
smc_rst=xx	
smc_vppen=xx	
smc_vppp=xx	
smc_det=xx	
smc_vccen=xx	
smc_sck=xx	
smc_sda=xx	

配置举例：

二十、USB 控制标志

(一) [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:PH4<0><1><default><default>
usb_det_vbus_gpio       = port:PH5<0><0><default><default>
usb_drv_vbus_gpio       = port:PB9<1><0><default><0>
```



```
usb_restrict_gpio      = port:PH26<1><0><default><0>
usb_host_init_state    = 0
usb_restric_flag       = 0
usb_restric_voltage    = 3550000
usb_restric_capacity   = 5
```

(二) [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_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。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_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_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：

```
[usbc1]
usb_used      = 1
usb_port_type = 1
usb_detect_type = 0
usb_id_gpio    =
usb_det_vbus_gpio =
usb_drv_vbus_gpio = port:PH6<1><0><default><0>
usb_restrict_gpio = port:PH26<1><0><default><0>
usb_host_init_state = 1
usb_restric_flag = 0
```

(三) [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_id_gpio=xx	USB 限流控制 pin 脚 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。具体请参考 gpio 配置说明。《配置与 GPIO 管理.doc》
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 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                = 1
usb_port_type           = 1
usb_detect_type         = 0
usb_id_gpio             =
usb_det_vbus_gpio       =
usb_drv_vbus_gpio       = port:PH3<1><0><default><0>
usb_restrict_gpio       = port:PH26<1><0><default><0>
usb_host_init_state     = 1
usb_restric_flag        = 0
```

二十一、USB Device

(一) [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"

```

(二) [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                 = 2

```

二十二、重力感应(G Sensor)

(一) [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        = 2
gsensor_twi_addr      = 0x18
gsensor_int1          = port:PA09<6><1><default><default>
gsensor_int2          =
```

二十三、WIFI

(一) [wifi_para]

配置项	配置项含义
wifi_used =xx	是否要使用 wifi
wifi_sdc_id =xx	sdio wifi 选用的是哪个 sdc 作为接口
wifi_usbc_id =xx	usb wifi 选用的是哪个 usb 作为接口
wifi_usbc_type=xx	usb 接口类型，1 为 ehci，0 为 ohci
wifi_mod_sel =xx	具体选择哪一款模组 1-bcm40181; 2-bcm40183; 3-rtl8723as; 4-rtl8189es; 5 - rtl8192cu; 6 - rtl8188eu; 7 - rtl8723au;
wifi_power=xx	给模组供电的 axp 引脚名

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

(二) sdio 接口 wifi rtl8723as demo

```
[wifi_para]
wifi_used          = 1
wifi_sdc_id        = 1
wifi_usbc_id       = 1
wifi_usbc_type     = 1
wifi_mod_sel       = 3
wifi_power         = "axp22_aldol"

; 3 - rtl8723as sdio wifi + bt gpio config
rtk_rtl8723as_wl_dis = port:PG10<1><default><default><0>
rtk_rtl8723as_bt_dis = port:PG11<1><default><default><0>
rtk_rtl8723as_wl_host_wake = port:PG12<0><default><default><0>
rtk_rtl8723as_bt_host_wake = port:PG17<0><default><default><0>
```

以上配置意思是要使用序号为 3 的 SDIO 接口 rtl8723as 模组，选用 SDC1 接口。

SDC1 对应是 mmc1，需要确定[mmc1_para]配置项如下：

```
[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"
```

(三) usb 接口 wifi rtl8188eu demo

```
[wifi_para]
wifi_used        = 1
wifi_sdc_id      = 1
wifi_usbc_id     = 1
wifi_usbc_type   = 1
wifi_mod_sel     = 6
wifi_power       = "axp22_aldol"
```

以上配置意思是要使用序号为 6 的 ehci USB 接口 rtl8188eu 模组，选用 usb1 接口。需要确定[usbc1]配置项如下：

```
[usbc1]
usb_used         = 1
usb_port_type    = 1
usb_detect_type  = 0
usb_id_gpio      =
usb_det_vbus_gpio =
usb_drv_vbus_gpio =
usb_restrict_gpio =
usb_host_init_state = 0
usb_restric_flag = 0
```

二十四、3G

(一) [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。

配置举例:

```
[3g_para]
3g_used          = 1
3g_usbc_num      = 2
3g_uart_num      = 0
bb_name          = "mu509"
bb_vbat          = port:PL03<1><default><default><0>
bb_on            = port:PM01<1><default><default><0>
bb_pwr_on        = port:PM03<1><default><default><0>
bb_wake          = port:PM04<1><default><default><0>
bb_rf_dis        = port:PM05<1><default><default><0>
bb_rst           = port:PM06<1><default><default><0>
```

二十五、gyroscope

(一) [gy_para]

配置项	配置项含义
gy_used=xx	是否支持 gyr
gy_twi_id=xx	I2C 的 BUS 控制选择，0 : TWI0;1:TWI1;2:TWI2
gy_twi_addr=xx	芯片的 I2C 地址
gy_int1=xx	中断 1 的 GPIO 配置
gy_int2=xx	中断 2 的 GPIO 配置

配置举例：

[gy_para]

gy_used = 1

gy_twi_id = 2

gy_twi_addr = 0x6a

gy_int1 = port:PA10<6><1><default><default>

gy_int2 =

二十六、光感(light sensor)

(一) [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 = 2

ls_twi_addr = 0x23

ls_int = port:PA12<6><1><default><default>

二十七、罗盘 Compass

(一) [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_int =xx	中断的 GPIO 配置

配置举例：

[compass_para]

compass_used = 1

compass_twi_id = 2

compass_twi_addr = 0x0d

compass_int = port:PA11<6><1><default><default>

二十八、蓝牙(blutetooth)

(一) [bt_para]

配置项	配置项含义
bt_used=xx	BLUETOOTH 使用控制: 1 使用, 0 不用
bt_uart_id=xx	BLUETOOTH 使用的 UART 控制器号
bt_wakeup =xx	BT WAKEUP GPIO 配置
bt_gpio=xx	BT 可选 GPIO 配置
bt_rst=xx	BT RESET GPIO 配置

配置举例:

[bt_para]

bt_used = 0

bt_uart_id = 2

bt_wakeup = port:PI20<1><default><default><default>

bt_gpio = port:PI21<1><default><default><default>

bt_rst = port:PB05<1><default><default><default>

二十九、数字音频总线（I2S）

（一）[i2s_para]

配置项	配置项含义
i2s_used=xx	xx 为 0 时加载该模块，为 0 是不加载
i2s_channel=xx	声道控制
i2s_mclk=xx	I2sMCLK 信号的 GPIO 配置
i2s_bclk=xx	I2sBCLK 信号的 GPIO 配置
i2s_lrclk=xx	I2sLRCK 信号的 GPIO 配置
i2s_dout0	I2S out0 的 GPIO 配置
i2s_dout1	暂不使用
i2s_dout2	暂不使用
i2s_dout3	暂不使用
i2s_din	I2sIN 信号的 GPIO 配置

配置举例：

```

i2s_used          = 0
i2s_channel       = 2
i2s_mclk          = port:PB5<2><1><default><default>
i2s_bclk          = port:PB6<2><1><default><default>
i2s_lrclk         = port:PB7<2><1><default><default>
i2s_dout0         = port:PB8<2><1><default><default>
i2s_dout1         =
i2s_dout2         =
i2s_dout3         =
i2s_din           = port:PB12<2><1><default><default>
  
```

三十、数字音频总线(pcm)

(一) [pcm_para]

配置项	配置项含义
pcm_used=xx	xx 为 0 时加载该模块，为 0 是不加载
pcm_channel=xx	声道控制
pcm_mclk =xx	暂不使用
pcm_bclk=xx	pcmBCLK 信号的 GPIO 配置
pcm_lrclk=xx	pcmLRCK 信号的 GPIO 配置
pcm_dout	pcm out 的 GPIO 配置
pcm_din	pcmIN 信号的 GPIO 配置

配置举例：

[pcm_para]

pcm_used = 1

pcm_channel = 2

pcm_mclk =

pcm_bclk = port:PG13<3><1><default><default>

pcm_lrclk = port:PG14<3><1><default><default>

pcm_dout = port:PG16<3><1><default><default>

pcm_din = port:PG15<3><1><default><default>

三十一、数字音频总线（S/PDIF）

（一）[spdif_para]

配置项	配置项含义
spdif_used=xx	xx 为 0 时加载该模块，为 0 是不加载
spdif_dout =xx	Spdif out 的 gpio 控制
spdif_din=xx	

配置举例：

[spdif_para]

spdif_used = 1

spdif_dout = port:PH28<3><1><default><default>

spdif_din =

三十二、喇叭控制

(一) [audio_para]

配置项	配置项含义
audio_used =xx	Audiocodec 是否使用， 1：打开（默认）0：关闭
audio_pa_ctrl=xx	喇叭的 gpio 口控制。
pa_vol	喇叭音量大小
cap_vol	录音音量大小

配置举例：

[audio_para]

audio_used = 1

audio_pa_ctrl = port:PH15<1><default><default><0>

pa_vol = 0x1b

cap_vol = 0x5

三十三、红外(ir)

(一) [ir_para]

配置项	配置项含义
ir_used=xx	是否支持 ir
ir0_rx=xx	ir 的接收管脚 GPIO 配置

配置举例：

[ir_para]

ir_used = 1

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

三十四、PMU 电源

(一) [pmu_para]

pmu_used=xx	Pmu 使能标志(xx=1 or 0), 0: 不使用, 1: 使用
pmu_twi_addr=xx	Pmu 设备地址
pmu_twi_id=xx	Pmu 挂载的 i2c 控制器号, 0: twi0, 1: twi1, 2: twi2
pmu_irq_id=xx	Pmu 中断号, 0: NMI, 1: 1 号中断 2: 2 号中断……
pmu_battery_rdc=xx	电池内阻, mΩ, 根据实际测试填写
pmu_battery_cap=xx	电池容量, mAh, 根据实际测试填写
pmu_batdeten	PMU 电池检测功能使能, 0: 不自动检测 1: 自动检测
pmu_runtime_chgcur=xx	设置开机充电电流, mA, 300/450/600/750/900/1050/1200 /1350/1500/1650/1800/1950
pmu_earlysuspend_chgcur=xx	设置关屏充电电流, mA, 300/450/600/750/900/1050/1200 /1350/1500/1650/1800/1950
pmu_suspend_chgcur=xx	设置休眠充电电流, mA, 300/450/600/750/900/1050/1200 /1350/1500/1650/1800/1950
pmu_shutdown_chgcur=xx	设置关机充电电流, mA 300/450/600/750/900/1050/1200 /1350/1500/1650/1800/1950
pmu_init_chgvol=xx	设置充电目标电压, mV, 4100/4220/4200/4240
pmu_init_chgend_rate=xx	设置结束充电电流的比率, %, 10, 15
pmu_init_chg_enabled=xx	设置充电功能, 0: 关闭, 1: 打开
pmu_init_adc_freq=xx	设置 adc 采样率, Hz, 100/200/400/800
pmu_init_adc_freqts=xx	设置 TS 引脚采样率, Hz, 100/200/400/800
pmu_init_chg_pretime=xx	设置预充电超时时间, min, 40/50/60/70
pmu_init_chg_csttime=xx	设置恒流充电超时时间, min, 360/480/600/720
pmu_batt_cap_correct	完成一次完成充放电后, 电池容量是否自校正功能使能, 0: 关闭 1: 开启
Pmu_bat_regu_en = x	充电完成后, bat regulator 是否常开, 0: 关闭, 1: 常开
pmu_bat_para1=xx	设置空载电池电压为 3.13V 对应的百分比, %
pmu_bat_para2=xx	设置空载电池电压为 3.27V 对应的百分比, %
pmu_bat_para3=xx	设置空载电池电压为 3.34V 对应的百分比, %

pmu_bat_para4=xx	设置空载电池电压为 3.41V 对应的百分比，%
pmu_bat_para5=xx	设置空载电池电压为 3.48V 对应的百分比，%
pmu_bat_para6=xx	设置空载电池电压为 3.52V 对应的百分比，%
pmu_bat_para7=xx	设置空载电池电压为 3.55V 对应的百分比，%
pmu_bat_para8=xx	设置空载电池电压为 3.57V 对应的百分比，%
pmu_bat_para9=xx	设置空载电池电压为 3.59V 对应的百分比，%
pmu_bat_para10=xx	设置空载电池电压为 3.61V 对应的百分比，%
pmu_bat_para11=xx	设置空载电池电压为 3.63V 对应的百分比，%
pmu_bat_para12=xx	设置空载电池电压为 3.64V 对应的百分比，%
pmu_bat_para13=xx	设置空载电池电压为 3.66V 对应的百分比，%
pmu_bat_para14=xx	设置空载电池电压为 3.7V 对应的百分比，%
pmu_bat_para15=xx	设置空载电池电压为 3.73V 对应的百分比，%
pmu_bat_para16=xx	设置空载电池电压为 3.77V 对应的百分比，%
pmu_bat_para17=xx	设置空载电池电压为 3.78V 对应的百分比，%
pmu_bat_para18=xx	设置空载电池电压为 3.8V 对应的百分比，%
pmu_bat_para19=xx	设置空载电池电压为 3.82V 对应的百分比，%
pmu_bat_para20=xx	设置空载电池电压为 3.84V 对应的百分比，%
pmu_bat_para21=xx	设置空载电池电压为 3.85V 对应的百分比，%
pmu_bat_para22=xx	设置空载电池电压为 3.87V 对应的百分比，%
pmu_bat_para23=xx	设置空载电池电压为 3.91V 对应的百分比，%
pmu_bat_para24=xx	设置空载电池电压为 3.94V 对应的百分比，%
pmu_bat_para25=xx	设置空载电池电压为 3.98V 对应的百分比，%
pmu_bat_para26=xx	设置空载电池电压为 4.01V 对应的百分比，%
pmu_bat_para27=xx	设置空载电池电压为 4.05V 对应的百分比，%
pmu_bat_para28=xx	设置空载电池电压为 4.08V 对应的百分比，%
pmu_bat_para29=xx	设置空载电池电压为 4.1V 对应的百分比，%
pmu_bat_para30=xx	设置空载电池电压为 4.12V 对应的百分比，%
pmu_bat_para31=xx	设置空载电池电压为 4.14V 对应的百分比，%
pmu_bat_para32=xx	设置空载电池电压为 4.15V 对应的百分比，%
pmu_usbvol_limit=xx	设置 usb 限压功能，0：关闭，1：打开
pmu_usbvol=xx	设置 usb 限压电压，mV，4000/4100/4200/4300/4400/4500/4600/4700
pmu_usbvol_pc =xx	设置连接至 PC 时 USB 限压值，mV，4000/4100/4200/4300/4400/4500/4600/4700
pmu_usbcu_r_limit=xx	设置 usb 限流功能，0：关闭，1：打开
pmu_usbcu_r=xx	设置 usb 限流电流，mA，500/900,若设置为 0，则不限流
pmu_usbcu_r_pc =xx	设置连接至 PC 时 USB 限流值，mA。500/900,若设置为 0，则不限流
pmu_pwroff_vol=xx	设置启动过程中硬件保护电压，mV，2600/2700/2800/2900/3000/3100/3200/3300
pmu_pwron_vol=xx	设置开机状态下的硬件保护电压，mV，2600/2700/2800/2900/3000/3100/3200/3300

pmu_pekoff_time=xx	设置硬件关机时长, ms, 4000/6000/8000/10000
pmu_pekoff_func=xx	设置长按键强制关机后是否自动启动功能, 0: 不自动启动 1: 自动启动
pmu_pekoff_en=xx	设置长按键硬件关机功能, 0: 关闭, 1: 打开
pmu_peklong_time=xx	设置长按键中断时间, ms, 1000/1500/2000/2500
pmu_pekcon_time=xx	设置开机时间, ms, 128/1000/2000/3000
pmu_pwrok_time=xx	设置电源启动完成后 pwrok 信号延时, ms, 8/16/32/64
pmu_battery_warning_level1 =xx	设置电池低电第一级报警门限: 可设范围 5%~20%, 1%/step
pmu_battery_warning_level2=xx	设置电池低电第二级报警门限: 可设范围 1%~15%, 1%/step
pmu_restvol_time=xx	设置电池电量更新时间间隔值, 可设范围 30s/60s/120s
pmu_restvol_adjust_time =xx	设置校正电池电量时间间隔值 1, 可设范围 30s/60s/120s
pmu_ocv_cou_adjust_time=xx	设置校正电池电量时间间隔值, 可设范围 30s/60s/120s
pmu_chgled_func=xx	设置 CHGLED 引脚功能, 0: 由驱动程序控制, 1: 由充电逻辑控制
pmu_chgled_type	设置 CHGLD 由充电逻辑控制时, 只是方式。0: 方式 A 1: 方式 B
pmu_vbusen_func	设置 N_VBUEN 引脚功能, 0: 作为输出功能 1: 作为输入功能
pmu_reset	设置长按键 16s 后是否复位 PMU, 0: 不复位 1: 复位
pmu_IRQ_wakeup	设置在 PMU 待机和关机情况下, IRQ 是否唤醒 PMU 功能: 0: 不唤醒 1: 唤醒
pmu_hot_shutdownm	设置 PMU 内部温度过高是否自动关机保护功能, 0: 不关机 1: 关机
pmu_inshort	设置 ACIN 和 VBUS 是否短路功能, 0-由 PMU 自动检测 1: 驱动设置为短路功能

配置举例:

```

pmu_used                = 1
pmu_twi_addr            = 0x34
pmu_twi_id              = 1
pmu_irq_id              = 0
pmu_battery_rdc         = 100
pmu_battery_cap         = 0
pmu_batdeten            = 1
pmu_runtime_chgcur      = 600
pmu_earlysuspend_chgcur = 900

```

pmu_suspend_chgcur	= 1500
pmu_shutdown_chgcur	= 1500
pmu_init_chgvol	= 4200
pmu_init_chgend_rate	= 15
pmu_init_chg_enabled	= 1
pmu_init_adc_freq	= 800
pmu_init_adcts_freq	= 800
pmu_init_chg_pretime	= 70
pmu_init_chg_cstime	= 720
pmu_batt_cap_correct	= 1
pmu_bat_regu_en	= 0

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	= 0
pmu_bat_para8	= 2
pmu_bat_para9	= 5
pmu_bat_para10	= 8
pmu_bat_para11	= 9
pmu_bat_para12	= 10
pmu_bat_para13	= 13
pmu_bat_para14	= 16
pmu_bat_para15	= 26
pmu_bat_para16	= 36
pmu_bat_para17	= 41
pmu_bat_para18	= 46
pmu_bat_para19	= 50
pmu_bat_para20	= 53
pmu_bat_para21	= 57
pmu_bat_para22	= 61
pmu_bat_para23	= 67
pmu_bat_para24	= 73
pmu_bat_para25	= 78
pmu_bat_para26	= 84
pmu_bat_para27	= 88
pmu_bat_para28	= 92
pmu_bat_para29	= 93
pmu_bat_para30	= 94
pmu_bat_para31	= 95
pmu_bat_para32	= 100

pmu_usbvol_limit	= 1
pmu_usbcur_limit	= 0
pmu_usbvol	= 4400
pmu_usbcur	= 0
pmu_usbvol_pc	= 4400
pmu_usbcur_pc	= 900
pmu_pwroff_vol	= 3300
pmu_pwron_vol	= 2600
pmu_pekoff_time	= 6000
pmu_pekoff_func	= 0
pmu_pekoff_en	= 1
pmu_peklong_time	= 1500
pmu_pekcon_time	= 1000
pmu_pwrok_time	= 64
pmu_battery_warning_level1	= 15
pmu_battery_warning_level2	= 0
pmu_restvol_adjust_time	= 30
pmu_ocv_cou_adjust_time	= 60
pmu_chgled_func	= 0
pmu_chgled_type	= 0
pmu_vbusen_func	= 1
pmu_reset	= 0
pmu_IRQ_wakeup	= 0
pmu_hot_shutdownm	= 1
pmu_inshort	= 0
power_start	= 0