



A20 Android 快速移植指南

V1.0

2013-12-27

Revision History

Version	Date	Changes compared to previous issue
V1.0	2013-12-27	初建版本



目录

1. A20 概述.....	6
1.1. A20 主控介绍.....	6
1.2. 外围设备介绍.....	6
1.3. 软件资源介绍.....	6
2. 建立开发环境.....	7
2.1. 硬件资源.....	7
2.2. 软件资源.....	7
2.2.1. 安装 JDK (ubuntu12.04)	7
2.2.2. 安装平台支持软件 (ubuntu12.04)	7
2.2.3. 安装编译工具链 (ubuntu12.04)	8
2.2.4. 安装 phoenixSuit (windows xp)	8
2.2.5. 其他软件 (windows xp)	8
3. 源码下载.....	9
3.1. wing 源码下载.....	9
3.2. 仓库的目录树.....	9
3.2.1. android 目录树.....	9
3.2.2. lichee 目录结构.....	10
3.2.2.1. buildroot 目录结构.....	10
3.2.2.2. linux-3.4 目录结构.....	11
3.2.2.3. u-boot 目录结构.....	12
3.2.2.4. tools 目录结构.....	13
3.2.2.5. boot 目录结构.....	13
4. 编译和打包.....	15
4.1. 源码编译.....	15
4.1.1. lichee 源码编译.....	15
4.1.2. android 源码编译.....	15
4.2. 打包固件.....	15
4.2.1. 完全打包.....	16
4.2.2. 局部打包.....	16
5. 固件烧写.....	17
5.1. 使用 PhoenixSuit 烧写固件.....	17
5.2. 使用 fastboot 更新系统.....	17
5.2.1. 进入 fastboot 模式.....	17
5.2.2. fastboot 命令使用.....	17
6. recovery 功能使用.....	18
6.1. 键值的查看.....	18
6.2. 按键选择.....	18
6.3. 功能使用.....	19
7. 调试.....	20
7.1. 调试 apk.....	20





7.2. 调试 linux 内核.....	20
7.3. 调试 android 系统.....	20
8. 系统定制.....	22
8.1. 启动 LOGO 修改.....	22
8.1.1. Android 初始化 LOGO.....	22
8.1.2. Android 引导动画.....	22
8.2. 预装 APK.....	23
8.2.1. 预装到 system/app 目录.....	23
8.2.2. 预装到 system/preinstall 目录.....	23
8.3. 预设桌面制定.....	24
8.4. boot 电池充电图标修改.....	24
8.5. 设备相关信息修改.....	25
8.5.1. 设备型号及软件版本修改.....	25
8.5.2. USB 相关信息修改.....	25
9. 模块配置.....	27
9.1. 自定义按键配置.....	27
9.1.1. KEY 的硬件原理.....	27
9.1.2. 驱动与硬件对应的关系.....	27
9.1.3. Android 按键功能的映射.....	28
9.2. WIFI 配置.....	28
9.2.1. USB WIFI 配置.....	28
9.2.2. SDIO WIFI 模块配置.....	29
9.3. LCD Panel 配置.....	30
9.3.1. 配置文件的修改.....	30
9.3.2. 配置系统 UI 方向属性.....	31
9.4. Touch Panel 配置.....	31
9.4.1. 配置文件的修改.....	32
9.4.2. Android 层的配置修改.....	33
9.4.3. touch panel 驱动使用说明.....	34
9.5. G Sensor 配置.....	35
9.5.1. 打包配置文件修改.....	36
9.5.2. Android 层配置修改.....	36
9.6. Camera 配置.....	38
9.6.1. 打包配置文件修改.....	38
9.6.2. Android 层的配置修改.....	43
9.6.3 Camera 参数配置.....	43
9.6.4 Camera 预览界面自适用配置.....	46
9.7. 震动马达配置.....	47
9.7.1. 配置文件修改.....	47
9.7.2. Android 层配置修改.....	47
9.8. SD 卡配置.....	47
9.8.1. 配置文件的修改.....	48



9.9. CTP 与 SENSOR 自动检测使用说明.....	49
9.9.1. CTP 自适应使用说明.....	49
9.9.2. GSENSOR 自适应使用说明.....	50
9.9.3. Recovery 功能 tp 的自适应使用说明.....	51
9.10. 安全控制配置.....	53
9.10.1. ADB 安全控制.....	53
10. Settings 与 Launcher 设置.....	54
10.1. 默认 LCD 关闭时间设置.....	54
10.2. 默认亮度设置.....	54
10.3. 默认字体大小设置.....	54
10.4. 电池电量百分比显示.....	54
10.5. 默认墙纸设置.....	54
10.6. 添加薄纸.....	55
10.7. Launcher 桌面默认图标和快捷栏设置.....	55
10.8. 下拉菜单 QuickSetting 中的开关显示.....	56
10.9. Settings 里面的蓝牙选项.....	56
10.10. Miracast 功能打开和关闭.....	56
10.11. 快速开关机功能使能和关闭.....	56
11. Declaration.....	57

1. A20 概述

A20 主控平台为珠海全志科技基于 ARM Cortex A7 开发的 Dual-Core 解决方案, GPU 采用 mali-400MP2, Memory 为 1G DDR3 (L) /LPDDR2, 在无线方面支持 WIFI、BT、3G, 该解决方案可以适用于 Tablet 和 Smart TV 等移动终端设备上。A20 与全志其他主控对比如下:

						
CPU	Quad-Core Cortex-A7	Dual-Core Cortex-A7	Single-Core Cortex-A8	Single-Core Cortex-A8	Single-Core Cortex-A8	Single-Core Cortex-A8
GPU	SGX544MP2	Mali400MP2	Mali400	Mali400	Mali400	Mali400
Video Decoder	4Kx2K	2160P	2160P	1080P	1080P	1080P
Video Encoder	H.264 1080P@60fps	H.264 1080P@30fps	H.264 1080P@30fps	H.264 1080P@30fps	H.264 1080P@30fps	H.264 1080P@30fps
Package	BGA609 18mmx18mm 0.65mm Pitch	BGA441 19mmx19mm 0.80mm Pitch	BGA441 19mmx19mm 0.80mm Pitch	BGA336 14mmx14mm 0.65mm Pitch	BGA336 14mmx14mm 0.65mm Pitch	eLQFP176 20mmx20mm
Application	Tablet Smartphone Smart TV	Tablet Smart TV	Tablet Smart TV	Smartphone	HDMI Dongle	Tablet E-reader

1.1. A20 主控介绍

A20 主控是采用双核 Cortex-A7 架构的 CPU, 主频可达 1G (1008MHz), 功耗控制出色。图形方面, GPU 采用 Mali400MP2, 兼容性更加出色。最高支持 2160P 的视频解码和 1080P@30fps 的编码, 多媒体性能优异。A 2 0 支持 1 G 告诉内存。另外 A20 还支持 1024x768 或 1024x600 等多种分辨率。

1.2. 外围设备介绍

A20 主控平台支持丰富的 Camera 模块, WIFI 模块, 蓝牙模块, 3G 模块 (电话系统), TF (SD) 卡扩展模块以及多种传感器。

1.3. 软件资源介绍

A20 主控的系统 and 软件平台是建立在 Android 4.2 平台基础上, Linux 内核版本为 3.3。Android 生态系统支持影音, 网络, 娱乐, 系统管理, 个人助手等丰富的扩展。

2. 建立开发环境

本节将介绍，A20 平台开发环境所需的软硬件资源及的搭建。

2.1. 硬件资源

- A20 主控开发板或主机一台
- 两台 PC：一台作为编译服务器（Linux 系统），令一台用于烧写固件（XP 系统）
- 串口线，12V 电源，小口 usb 各一个（条）

2.2. 软件资源

Linux 主机（因为 A20 的软件系统方案选择的是 android4.2，所以只能使用 64bit 系统，推荐使用 ubuntu12.04），硬盘空间至少 100G（可满足一次完全编译），一般来说 Linux 主机中需要：

- Python 的 2.6-2.7 版本
- GNU Make 的 3.81-3.82 版本
- JDK 6
- git 的 1.7 或更高版本

Windows XP 主机，作为固件烧写机器和本地调试环境，一般来说主机中需要：

- PhoenixSuit 一键烧写工具
- USB 转串口驱动
- Android SDK

下面以 ubuntu12.04 和 XP 为例，安装软件环境

2.2.1. 安装 JDK（ubuntu12.04）

JDK 安装命令

```
$ sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
$ sudo apt-get update
$ sudo apt-get install sun-java6-jdk
```

2.2.2. 安装平台支持软件（ubuntu12.04）

```
$ sudo apt-get install git gnupg flex bison gperf build-essential \
zip curl libncurses5-dev:i386 libncurses5-dev \
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
libgl1-mesa-dev g++-multilib mingw32 tofrodos \
python-markdown libxml2-utils xsltproc zlib1g-dev:i386
$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
```

2.2.3. 安装编译工具链（**ubuntu12.04**）

编译工具链已经集成在 Android SDK 中，工具链位于 Android SDK 中的 lichee/boot/config/gcc-linaro/中。

2.2.4. 安装 **phoenixSuit** （**windows xp**）

phoenixSuit 位于 lichee/tools/tools_win 中，将 PhoenixSuitPacket.msi 复制到 XP 主机上，按照安装向导提示安装，即可完成 phoenixSuit 的安装。

2.2.5. 其他软件（**windows xp**）

建议在 windows 系统下安装 putty，并且网络映射到上述 Linux 编译服务器进行 SDK 源码的编译。在开发过程中缺少相关驱动也优先从 SDK 中查找。

3. 源码下载

本节将介绍 A20 开发平台下源码的下载及源码结构

3.1. wing 源码下载

参照《A20 仓库下载说明 V1.0》。

3.2. 仓库的目录树

下面介绍仓库中的目录树结构，主要介绍 android 仓库的目录树结构，lichee 的目录树结构

3.2.1. android 目录树

android 目录下是 android 仓库中的 android 源码，A20 使用的 android 系统为 android4.2。查看 android 的目录树结构，在 android 的根目录下执行如下命令

```
$ tree -L 1
```

```
.
├── abi
├── bionic
├── bootable
├── build
├── cts
├── dalvik
├── development
├── device
├── docs
├── external
├── frameworks
├── gdk
├── hardware
├── libcore
├── libnativehelper
├── Makefile
├── ndk
├── packages
├── pdk
├── prebuilts
└── sdk
```

```
├── system
└── tools
```

上面目录中，与 android 官方源码相同

3.2.2. lichee 目录结构

lichee 目录下包含 buildroot 工具、linux 内核、以及 uboot 源码。查看 lichee 的目录结构，在 lichee 的根目录下执行下面的命令

```
$ tree -L 1
```

```
.
├── boot
├── buildroot
├── build.sh
├── linux-3.3
├── README
├── source
├── tools
└── u-boot
```

3.2.2.1. buildroot 目录结构

buildroot 的主要作用是

- 管理包之间的依赖关系
- 生成 ARM 交叉工具链
- 生成 U-Boot
- 制作与制定根文件系统
- 生成最终用于烧写的固件包

在 buildroot 的根目录下执行下列命令，查看 buildroot 目录结构如下

```
$ tree -L 1
```

```
.
├── board
├── boot
├── build.sh
├── CHANGES
├── Config.in
├── configs
├── COPYING
├── dl
├── docs
├── external-packages
├── fs
└── linux
```



- |— Makefile
- |— output
- |— package
- |— README
- |— scripts
- |— target
- |— toolchain

其中，boot 目录里存放 Boot 代码，config 目录里存放预定义好的配置文件，比如我们的 sun7i_defconfig，dl 目录里存放已经下载好的软件包，scripts 目录里存放 buildroot 运作的代码，target 目录里存放用于生成根文件系统的一些规则文件。对于我们来说最为重要的是 package 目录，里面存放了将近 3000 个软件包的生成规则，我们可以在里面添加我们自己的软件包或者是中间件。更多关于 buildroot 的介绍，可以到 buildroot 的官方网站获取

3.2.2.2. linux-3.4 目录结构

Linux-3.3 目录结构包含了 linux kernel 3.4 的源码，在 Linux-3.4 的根目录使用如下命令查看，该目录的结构

```
$ tree -L 1
```

```
.
|— arch
|— bImage
|— block
|— build.sh
|— COPYING
|— CREDITS
|— crypto
|— Documentation
|— drivers
|— firmware
|— fs
|— include
|— init
|— ipc
|— Kbuild
|— Kconfig
|— kernel
|— lib
|— MAINTAINERS
|— Makefile
|— mm
|— modules
```



```
├── net
├── output
├── README
├── REPORTING-BUGS
├── rootfs
├── samples
├── scripts
├── security
├── sound
├── tools
├── usr
└── virt
```

以上目录结构跟标准的 Linux 内核是一致的，除了多一个 modules 目录。modules 目录是我们扩展用来存放没有跟内核的 menuconfig 集成的外部模块的地方。我们目前放了 example, nand, mali 和 wifi 这 4 个外部模块，其中 example 是示例用的，mali 是我们的 GPU 驱动，test 是模块测试用例，目前只存放了 nand 的测试用例。

在 modules 的根目录下执行如下命令，结果如下

```
$ tree -L 1
```

```
.
├── example
├── mali
├── nand
└── wifi
```

3.2.2.3. u-boot 目录结构

u-boot 目录中存放的是 A20 主控平台 Linux 内核引导代码，在 u-boot 的根目录下执行下列命令，结果如下

```
$ tree -L 1
```

```
.
├── api
├── arch
├── board
├── boards.cfg
├── build.sh
├── common
├── config.mk
├── COPYING
├── CREDITS
├── disk
├── doc
└── drivers
```



```
|— examples
|— fs
|— include
|— lib
|— MAINTAINERS
|— MAKEALL
|— Makefile
|— mkconfig
|— mmc_spl
|— nand_spl
|— nand_sunxi
|— net
|— onenand_ipl
|— post
|— README
|— rules.mk
|— snapshot.commit
|— spl
|— tools
```

除了添加我们自己的 sunxi 平台设置，目录结构与官方网站上的没有区别，有关 u-boot 的详情请参阅 u-boot 的官方文档。

3.2.2.4. tools 目录结构

tools 目录为打包工具目录，与打包相关的脚本和工具都放在该目录中。它的结构如下：

```
$ tree -L 1
```

```
.
|— daily_build
|— pack
|— tools_win
```

3.2.2.5. boot 目录结构

boot 目录中存放的是 A20 平台的 bootloader，该目录为 A20 启动代码，该目录的结构如下：

```
$ tree -L 1
```

```
.
|— boot0
|— boot1
|— config
|— Makefile
```



└─ pack
└─ workspace

4. 编译和打包

本节讲解源码编译和打包的方法

4.1. 源码编译

下面将分别介绍 lichee 和 android 的从源码进行编译的方式方法

4.1.1. lichee 源码编译

使当前目录为 lichee 的根目录。然后执行下面的命令：

```
./build.sh -p sun7i_android
```

或者：

```
$ . buildroot/scripts/mksetup.sh      #导入环境变量，根据提示选择对应选项
```

```
$ mklichee
```

即完成了一次 lichee 的编译（根据服务器配置，耗时至少 10 分钟），编译成功，屏幕上会出现

```
INFO:build u-boot OK.
```

```
...
```

```
INFO:build rootfs OK.
```

```
INFO:build lichee OK.
```

4.1.2. android 源码编译

在确保 lichee 已经编译，并且使当前目录为 android 的根目录。然后执行下面的命令

```
$ . build/envsetup.sh      #导入环境变量
```

```
$ lunch                    #根据自己的开发平台，选择方案
```

```
$ extract-bsp              #拷贝内核和模块到 android 中
```

半
\$ make -j8 #-j 开启多核编译，服务器开发一般为服务器 cpu 数量的一

编译成功，会在 out/target/product/wing-xxx/ 目录下面会生成 boot.img, recovery.img, system.img 3 个包。

4.2. 打包固件

本节涉及两种打包方式，一种是完全打包，一种是局部打包

4.2.1. 完全打包

在保证 lichee 和 android 都编译完成的基础上，相关环境变量已经导入，只需要在 android 的根目录下执行下列命令即可

```
$ pack
```

打包成功后，将会在 lichee/tools/pack 目录下生成 sun7i_android_xxx.img 文件，即生成我们所需的固件

4.2.2. 局部打包

boot.img 镜像中包含 linux kernel 和内存盘 ramdisk，如果内核有修改，就需要重新编译内核，然后再 android 目录下执行下列命令,即可打包生成 boot.img

```
$ . build/envsetup.sh
```

```
$ lunch #选择 wing-xxx 产品
```

```
$ extract-bsp
```

```
$ make bootimage
```

这样就生成了 boot.img，类似的方法就可以重新打包生成 system.img

```
$ . build/envsetup.sh
```

```
$ lunch #选择 wing-xxx 产品
```

```
$ make systemimage-nodps
```

重新生成的镜像在 android 目录下的 out/target/product/wing-xxx/目录下

5. 固件烧写

本小节介绍 A20 下的固件的烧写

5.1. 使用 PhoenixSuit 烧写固件

安装好 PhoenixSuit 后，双击运行，单机一键刷机，选择我们生成的固件，如果我们的 A20 主动设备没有烧如任何系统，可以按住设备上的 recovery 键后插入 usb 然后按几下 power 键系统就会进入自动升级状态。如果已经烧入 android 系统，开机后与 windows 主机连接好，单机立即升级即可自动烧入新的固件。

5.2. 使用 fastboot 更新系统

fastboot 是一种线刷，就是使用 USB 数据线连接手机的一种刷机模式，在 A20 主控中，可以使用 fastboot 的功能来实现局部系统的更新。

5.2.1. 进入 fastboot 模式

启动开发板，在串口界面敲任意按键，可以进入 u-boot；如果进不了 fastboot，将 `lichee/tools\pack\chips\sun7i\configs\android\default\env.cfg` 中的 `bootdelay=0` 改成 `bootdelay=2` 重新打包固件即可（需要安装 google-usb_driver 驱动）。

在串口命令行输入 `fastboot` 命令，进入 fastboot 模式；

通过 pc 端的 fastboot 工具烧录各个固件包（fastboot 是 windows 下的一个工具（android sdk 中有），上网自己下载一个，解压到本地，然后将 fastboot.exe 添加到 windows 环境变量）

进入在 windows 命令行：cmd 进行命令行模式，于是可以在命令行执行 fastboot 指令

退出 fastboot 模式：`ctl+c`

5.2.2. fastboot 命令使用

在 windows 命令行使用 fastboot 命令。

擦除分区命令：

`$ fastboot erase boot` #擦除 boot 分区

`$ fastboot erase system` #擦除 system 分区

`$ fastboot erase data` #擦除 data 分区

烧写分区命令：

`$ fastboot flash boot boot.img` #把 boot.img 烧写到 boot 分区

`$ fastboot flash system system.img` #把 system.img 烧写到 system 分区

`$ fastboot flash data userdata.img` #把 userdata.img 烧写到 data 分区

6. recovery 功能使用

Recovery 是 Android 的专用升级模式，用于对 android 自身进行更新；进入 recovery 模式的方法是，在 android 系统开机时，按住一个特定按键，则会自动进入 android 的 recovery 模式。

6.1. 键值的查看

按键是通过 AD 转换的原理制成。当用户按下某个按键的时候，会得到这个按键对应的 AD 转换的值。同时，所有的按键的键值都不相同，并且，键值之间都有一定的间隔，没有相邻。比如，键值可能是 5,10,15,20，但是不可能是 5,11,12,13。

为了方便用户查看不同按键的键值，这种方法要求连接上串口使用，因此适合于开发阶段使用。具体步骤是：

把小机和 PC 通过串口线连接起来，设置屏幕焦点在串口调试软件上；

用户开机之前，按住 PC 键盘上的数字键“3”；

开机，等待，1 秒后可以松开电脑键盘；

经过这样的步骤，用户会看到屏幕上有如下的打印信息出现：

welcome to key value test

press any key, and the value would be printed

press power key to exit

这表示系统已经进入了按键的键值测试模式，这种模式下将一直等待用户按下按键，并在串口屏幕上把按键的键值打印出。这样，用户可以很方便地查看不同按键的键值。比如，当按下某一个按键，用户可以看到如下的打印信息。

key value = 8

key value = 8

key value = 8

key value = 63

由于 AD 采用的速度非常快，所以同一个按键按下，屏幕上会出现多个值。用户可以看出，这个按键的键值是 8。最后出现的 63 是松开按键的时候的采用，是需要去掉的干扰数据。因此，用户查看按键键值的时候只要关注前面打印出的数值，后面出现的应该忽略不计。

6.2. 按键选择

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。因此，系统中提供了一种方法用于选择进入 recovery 模式的按键：

在 efex/sys_config.fex 配置脚本中，提供了一项配置，用于选择按键的键值，如下所示：

[recovery_key]

key_max = 4

key_min = 6

它表示，所选择用于作为 recovery 功能的按键的键值范围落在 key_min 到 key_max 之间，即 4 到 6 之间。由于所有按键的选择都可以通过前面介绍的方法查看，因此，假设用户要选的按键是 a，用户这里选择配置的方法是：

按照前面介绍的方法，读出所有按键的键值；

读出 a 的键值 a1，同时取出两个相邻于 a 的键值，记为 b1 和 c1，b1 大于 c1；

计算出 $(a1 + b1)/2$ ， $(a1 + c1)/2$ ，分别填写到 key_max 和 key_min 处；

如果 a1 刚好是所有按键的最小值，则取 key_min 为 0；如果 a1 刚好是所有按键的最大值，则取 key_max 为 63；

经过以上的步骤，就可以选择一个特定的按键进入 recovery 模式。取了一个平均值的原因是考虑到长时间的使用，电阻的阻值可能会略有变化导致键值变化，取范围值就可以兼容这种阻值变化带来的键值变化。

6.3. 功能使用

在 android 编译完毕之后，使用如下命令

```
$ get_uboot
```

```
$ make otapackage
```

就可以在 android/out/target/product/wing-xxx/ 目录下生成一个 wing_xxx-ota-buildtime.zip 文件。

在系统启动时，按住设定的特定按键进入 recovery 模式，进入该模式后，可以选择升级文件升级。

7. 调试

本节介绍如何生成调试应用和固件的方式方法

7.1. 调试 apk

修改应用程序 Gallery2，编译修改推送到小机

```
$ . build/envsetup.sh
```

```
$ lunch #选择 wing-evb 产品
```

```
$ cd packages/apps/Gallery2
```

```
$ mm
```

执行“mm”命令局部编译 Gallery2 应用程序，生成 Gallery2Tests.apk。如下所示。

```
Install: out/target/product/wing-xxx/data/app/Gallery2Tests.apk
```

然后在 windows 命令行下将生成的 Gallery2Tests.apk 推送到小机的相应目录 system/app 下即可（注：需要预先安装 adb）。如下所示：

在 windows 命令行：cmd 进入命令行模式。

```
>adb push Gallery2Tests.apk /system/app/
```

7.2. 调试 linux 内核

在更改了内核相关文件后，在 lichee 目录下执行以下命令编译内核

```
$ ./build.sh -p sun7i_android
```

在 android 目录下执行以下命令，打包生成 boot.img

```
$ . build/envsetup.sh
```

```
$ lunch #选择 wing-xxx 产品
```

```
$ extract-bsp
```

```
$ make bootimage
```

生成了 boot.img 之后，需要通过 fastboot 工具刷到小机：

重新启动开发板，在串口界面敲任意按键，可以进入 u-boot；

在串口命令行输入 fastboot 命令，进入 fastboot 模式；

进入 windows 命令行：cmd 进行命令行模式，在命令行执行 fastboot(前提是已经安装了 fastboot 工具)，将 boot.img 拷贝到小机上即可。

```
$ fastboot erase boot
```

```
$ fastboot flash boot boot.img
```

7.3. 调试 android 系统

在 android 目录下执行以下指令，在 out/target/product/wing-xxx/目录下生成 android system.img。

```
$ . build/envsetup.sh
```



\$ lunch #选择 wing-xxx 产品

\$ make systemimage-nodeps

通过 fastboot 工具刷到小机:

重新启动开发板, 在串口界面敲任意按键, 可以进入 u-boot;

在串口命令行输入 fastboot 命令, 进入 fastboot 模式;

进入在 windows 命令行: cmd 进行命令行模式, 在命令行执行 fastboot 指令(前提是已经安装了 fastboot 工具), 将 system.img 拷贝到小机上即可。

\$ fastboot erase system

\$ fastboot flash system system.img

8. 系统定制

本小节主要介绍 android 部分系统定制的内容，其中涉及启动 LOGO 的修改，预装 APK 设置，预设桌面的制定，以及充电图标的修改等

8.1. 启动 LOGO 修改

启动 LOGO 为初始引导阶段的 LOGO。

位置：lichee/tools/pack/chips/sun7i/boot-resource/boot-resource/bootlogo.bmp

修改方法：替换 bootlogo.bmp。根据小机屏幕分辨率大小，定制 Logo 图片的大小，替换即可

8.1.1. Android 初始化 LOGO

位置：android/device/softwinner/*wing-xxx*/initlogo.rle

修改方法：替换 initlogo.rle。initlogo.rle 是通过 windows 端工具制作而成的文件，制作过程：将 logo 图片转化成 24bpp 或 32bpp bmp 图片，打开 LogoGen（位于 lichee/tools/tools_win）目录下的 BmpConvert 工具，按照《BMPConvert_UserManual.doc》文档说明来生成所需的 initlogo.rle 文件；

说明：framebuffer 使用 ARGB 的格式。

8.1.2. Android 引导动画

位置：android/device/softwinner/*wing-xxx*/media/bootanimation.zip

bootanimation 格式：bootanimation.zip 包含 part0 part1 文件夹和 desc.txt 文件，part0, part1 文件夹里面放的是动画拆分的图片，格式为 png 或 jpg；

desc.txt 格式：

800 480 15
p 1 0 part0
p 0 0 part1

说明：800 为宽度，480 为高度，15 为帧数，即每秒播放动画 15 帧；第一项 p 为标志符，第二项为循环次数，1 为只播放 1 次，0 为无限循环，第三项为两次循环之间间隔的帧数，第四项为对应的目录名（图片放在 desc.txt 中目录名指定的目录中，目录中按字符顺序播放）

打包格式要求：windows 使用 winrar 打包，选择 ZIP 格式，压缩标准要选“储存”；linux 下，zip -0 -r ../bootanimation.zip /* linux 命令使用 -0 指定压缩等级为最低等级 stored，即只归档不压缩，否则可能由于包格式问题引起动画显示为黑屏。打包完之后发到指定目录(wing-xxx/media)，修改其权限值：chmod 777 bootanimation.zip

8.2. 预装 APK

预装 apk 安装有两种方法，可以安装到 system/app 目录下，也可以安装到 system/preinstall 目录下。

8.2.1. 预装到 system/app 目录

1. 预装的 APK 存放位置在 android/device/softwinner/wing-common/prebuild/apk 目录下，如果该 apk 包含有 lib，则需要将 lib 库拷贝到 device/softwinner/wing-common/prebuild/apklib 目录下。

2. 修改 android/device/softwinner/wing-common/prebuild/apk/Android.mk, 添加

```
#####  
include $(CLEAR_VARS)  
LOCAL_MODULE := com.google.android.apps.maps  
LOCAL_MODULE_TAGS := optional  
LOCAL_CERTIFICATE := PRESIGNED  
LOCAL_MODULE_CLASS := APPS  
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk  
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)  
include $(BUILD_PREBUILT)  
#####
```

注意红色部分改为 apk 的名称

3. 修改 android4.4/device/softwinner/wing-common/wing-common.mk 增加：

```
PRODUCT_PACKAGES +=  
com.google.android.apps.maps
```

注意：安装到 system/app 目录下的 apk 为系统应用，不能卸载。

8.2.2. 预装到 system/preinstall 目录

1. 预装的 APK 存放位置在 android/device/softwinner/wing-common/prebuild/preinstallapk 目录下。

2. 修改 android/device/softwinner/wing-common/prebuild/preinstallapk/Android.mk, 添加

```
#####  
include $(CLEAR_VARS)  
LOCAL_MODULE := com.google.android.apps.maps  
LOCAL_MODULE_TAGS := optional  
LOCAL_CERTIFICATE := PRESIGNED  
LOCAL_MODULE_PATH := $(TARGET_OUT)/preinstall  
LOCAL_MODULE_CLASS := APPS  
LOCAL_SRC_FILES := $(LOCAL_MODULE)  
include $(BUILD_PREBUILT)
```


#####注意红色部分改为 apk 的名称

3.修改 android4.4/device/softwinner/wing-common/wing-common.mk 增加:

PRODUCT_PACKAGES += \

com.google.android.apps.maps

注意:

1) 存放在该目录下的 apk 可以被卸载

2) 安装之前会做标志位, 如安装过程中出现异常, 则其余应用不会自动安装。

注: apk 包的文件名中不能包含中文和空格

8.3. 预设桌面制定

目前桌面使用 google 默认的 Launcher2, 源码位置为 android/packages/apps/Launcher2, 桌面的快捷方式在 android/packages/apps/Launcher2/res/xml/default_workspace.xml 文件中修改:

<appwidget	
launcher:packageName="com.google....genie.geniewidget"	//widget
package name	
launcher:className="com.google....MiniWidgetProvider"	//widget
class name	
launcher:screen="1"	//第几屏
launcher:x="0"	//x\y 坐标
launcher:y="0"	
launcher:spanX="4"	//widget 占用 x\y
的格数	
launcher:spanY="1" />	
<favorite	
launcher:packageName="com.android.camera"	//apk package
name	
launcher:className="com.android.camera.Camera"	//apk activity
class name	
launcher:screen="1"	//第几屏
launcher:x="1"	//x\y 坐标
launcher:y="3" />	

favorite 项为桌面快捷方式, appwidget 项为 widget 的设置, 其各项的意义如上面标注所示;

8.4. boot 电池充电图标修改

电 池 图 标 是 一 个 bmp 图 片 , 是 位 于
lichee/tools/pack/chips/sun7i/boot-resource/boot-resource/bat 目录下的十三张, 名字分别为

A20 Android 快速移植指南

Copyright © 2013 Allwinner Technology. All Rights Reserved.

- 24 -

从 bat0.bmp 到 bat10.bmp, low_pwr.bmp, bempty.bmp 的文件;

如果希望更换这个显示的电池图标, 直接用相应分辨率的同名文件替换掉这个文件即可。

8.5. 设备相关信息修改

本小节主要讲解 A20 主控下 Android4.2 平台下的设备相关信息的修改。

8.5.1. 设备型号及软件版本修改

在编译后生成的 android/out/target/product/*wing-xxx*/system/build.prop 文件中记录了设置中的显示信息, 常用的信息 (设置-->关于设备) 如下:

```
型号 -->ro.product.model
固件版本--> ro.product.firmware
Android 版本-->ro.build.version.release
版本号 -->ro.build.display.id
```

修改方法: 源码中修改 android/device/softwinner/*wing-xxx*/wing_XXX.mk, 添加如下:
修改固件版本为 V1.3

```
PRODUCT_PROPERTY_OVERRIDES += \
ro.product.firmware=V1.3
```

修改设备型号为 SoftwinnerEvb

等效于在 build.prop 文件中修改 ro.product.model=SoftwinnerEvb

```
PRODUCT_MODEL := SoftwinnerEvb
```

注意: 设备型号中应该全部为英文字符, 不能出现中文, 可以有空格;

8.5.2. USB 相关信息修改

第一次开机 flash 盘符的修改: 在 android/device/softwinner/*wing-xxx*/init.sun7i.rc 的 “format_userdata” shell 命令

后后面接格式化用的盘符, 如:

```
format_userdata /dev/block/UDISK WING
```

格式化 flash 盘符为 WING, 当打开 USB 设备后即可在电脑上看到给盘符;

用户格式化 flash 盘符修改: 修改 android/device/softwinner/*wing-xxx*/wing_XXX.mk 文件的 ro.udisk.lable=WING 属性。

连接电脑时, 显示驱动的修改:

如修改设备连接电脑时, 电脑显示驱动为 “WING USB 2.0 Driver”, 则在 lichee/tools/pack/chips/sun7i/configs/android/*wing-xxx*/sys_config.fex 文件中修改如下:

```
[msc_feature]
vendor_name  = "WING"
product_name = "USB 2.0 Driver"
```



设备序列号的修改：默认设备序列号为"20080411"，该序列号主要在豌豆夹手机精灵上有显示，设备的序列号最好为英文大写字母(A~F)及数字的组合，在 sys_config.fex 文件中：

```
[usb_feature]  
serial_number = "20080411"
```

9. 模块配置

本小节主要讲解 A20 主控下的 Android4.2 的一些外围模块的配置

9.1. 自定义按键配置

9.1.1. KEY 的硬件原理

目前 KEY 检测使用了 ADC 转换的原理实现的，由于该原理的限制，所以不能区分组合键（功能键，不包括电源键）；按照目前公版原理图，0.2V 的电压变化可以区分一档，所以最多可以实现 10 个键，硬件原理如下：

KEY

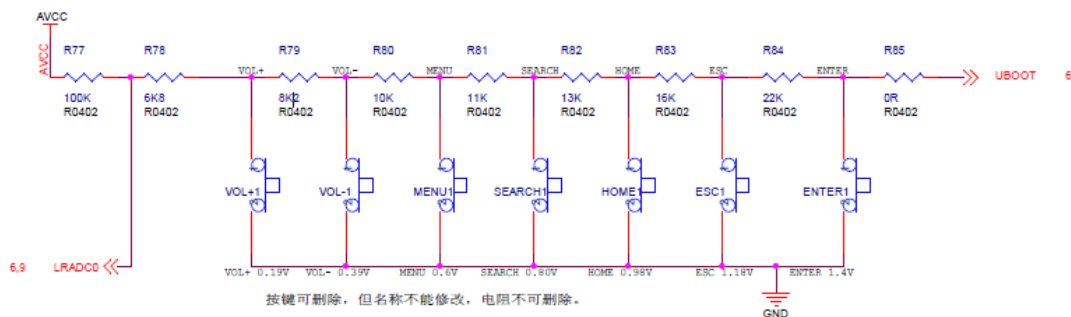


图 9.1.1

9.1.2. 驱动与硬件对应的关系

Key 的驱动实现文件位置：lichee/linux-3.3/drivers/input/keyboard/sw-keyboard.c;
实现原理：通过检测电压值的数字量来区分当前是第几个按键：

```
//0.2V mode
static unsigned char keypad_mapindex[64] =
{
    0,0,0,0,0,0,0,0, //key 1, 8 个, 0-7
    1,1,1,1,1,1,1,1, //key 2, 7 个, 8-14
    2,2,2,2,2,2,2,2, //key 3, 7 个, 15-21
    3,3,3,3,3,3,3,3, //key 4, 6 个, 22-27
    4,4,4,4,4,4,4,4, //key 5, 6 个, 28-33
    5,5,5,5,5,5,5,5, //key 6, 6 个, 34-39
    6,6,6,6,6,6,6,6,6,6, //key 7, 10 个, 40-49
    7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7, //key 8, 17 个, 50-63
};
```

按键对应的键值：



```
static unsigned int sw_scankeycodes[KEY_MAX_CNT] = {  
    [0] = KEY_VOLUMEUP,  
    [1] = KEY_VOLUMEDOWN,  
    [2] = KEY_MENU,  
    [3] = KEY_ENTER,  
    [4] = KEY_HOME,  
    [5] = KEY_RESERVED,  
    [6] = KEY_RESERVED,  
    [7] = KEY_RESERVED,  
    [8] = KEY_RESERVED,  
    [9] = KEY_RESERVED,  
    [10] = KEY_RESERVED,  
    [11] = KEY_RESERVED,  
    [12] = KEY_RESERVED,  
};
```

当有按键事件时，通过以上两次映射将最终的键值上报：

```
scancode = keypad_mapindex[key_val&0x3f];  
input_report_key(swkbd_dev, sw_scankeycodes[scancode], 1);  
input_sync(swkbd_dev);
```

9.1.3. Android 按键功能的映射

映射文件为 android/device/softwinner/~~wing-xxx~~/sw-keyboard.kl，举例如下：

key 102	HOME	WAKE
key 1	BACK	WAKE
key 139	MENU	WAKE_DROPPED
key 115	VOLUME_UP	
key 114	VOLUME_DOWN	

key 后面的数字为驱动中上报的键值，后面为对应的功能，自定义按键时仅需要将前面的映射值和后面的功能对应起来即可；（WAKE_DROPPED：唤醒屏幕，但是这个按键不会发给当前应用程序，WAKE：唤醒屏幕，但是这个按键需要发送给应用程序，后面不加的代表没有唤醒功能）

9.2. WIFI 配置

本小节主要讲解 USB WIFI 模块和 SDIO WIFI 模块的配置

9.2.1. USB WIFI 配置

配置文件：lichee/tools/pack/chips/sun7i/configs/android/~~wing-xxx~~/sys_config.fex 根据



硬件原理图配置‘USB 控制标志’和‘wifi_para’部分，前者标志 usb vcc 电平控制引脚及逻辑，后者标志是 wifi 配置项，举例如下：

wifi_used	= 1	#1 表示使用；0 表示不使用
wifi_sdc_id	= 3	#0: SDC0; 1: SDC1; 2: SDC2; 3SDC3
wifi_usbc_id	= 2	#0: USB0; 1: USB1; 2: USB2
wifi_usbc_type	= 1	#1: ECHI; 2OHCI
wifi_mod_sel	= 7	#0: none; 1: bcm40181; 7: ap6210
wifi_power	= ""	

共有三组 USB，USB0 用于 OTG，USB1/USB2 用于 HOST。wifi 可以使用 USB1 或 USB2。wifi_used 标志平台是否使用 wifi；wifi_usbc_id 标志 wifi 使用哪一组 usb；wifi_usbc_type 标志 usb 接口支持的版本；wifi_mod_sel 标志选用哪一款 wifi 模组；wifi_power 标志 wifi VCC 是哪个 axp pin 控制；可根据自己的硬件原理图情况来配置该部分；

Android 层配置，默认在 android/device/softwinner/*wing-xxx*/BoardConfig.mk 文件中有以下配置：

```

ifeq ($(BOARD_WIFI_VENDOR), realtek)
    WPA_SUPPLICANT_VERSION := VER_0_8_X
    BOARD_WPA_SUPPLICANT_DRIVER := NL80211
    BOARD_WPA_SUPPLICANT_PRIVATE_LIB :=
lib_driver_cmd_rtl
    SW_BOARD_USR_WIFI := rtl8723as
    BOARD_WLAN_DEVICE := rtl8723as
endif

```

SW_BOARD_USR_WIFI 和 BOARD_WLAN_DEVICE 是标志选用哪一款 wifi 模组，这两个宏会直接影响到 hardware/libhardware_legacy/下的 wifi.c 的编译。

9.2.2. SDIO WIFI 模块配置

配置文件：lichee/tools/pack/chips/sun7i/configs/android/*wing-xxx*/sys_config.fex 根据硬件原理图来确定 sdio wifi 使用哪一组 sdio 卡接口，按照如下配置（以 mmc1 为例）：

```

[mmc1_para]
sdc_used          = 0      #1:使用；0: 不使用
sdc_detmode       = 4      #1: 轮训检测；2: 中断检测；3: 引导卡；4: 手动检测
sdc_buswidth      = 4      #1:1bit; 4:4bit; 8:8bit
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      #1: 带有写保护; 0: 无写保护
sdc_wp               =
sdc_isio              = 0      #SDIO 卡
sdc_regulator         = "none" #电源控制
[wifi_para]
wifi_used             = 1
wifi_sdc_id           = 3
wifi_usbc_id          = 2
wifi_usbc_type        = 1
wifi_mod_sel          = 7
wifi_power            = ""
; 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>
```

mmc1_para 部分配置 sdio 通信 data 及 clk 信号, 一般不改动, wifi_para 部分配置 sdio wifi 使用哪个 sdc 卡接口、相关控制 GPIO 的选择, 请根据实际原理图来配置;

Android 层配置: 此部分跟 usb wifi 配置中的 android 层配置一样。

注: 因 wifi 部分和 android framework 配置层关联比较大, 更改 wifi 配置后, 建议 make clean 之后再编译, 否则会出现 wifi 无效的情况;

9.3. LCD Panel 配置

9.3.1. 配置文件的修改

配置文件: lichee/tools/pack/chips/sun7i/configs/android/~~wing-xxx~~/sys_config.fex

调试 lcd pannel 的参数: 如发现屏幕闪动或显示位置有偏差, 可以按照该 panel 的 spec 调整如下参数:

参数调整: 用户需要根据 panel 的 datasheet, 调整如下参数: (仅供参考)

```
[lcd0_para]
lcd_used              = 1      #1: 使能; 0 不使能
lcd_x                 = 800    #x 方向的分辨率
lcd_y                 = 480    #y 方向的分辨率
lcd_dclk_freq         = 33     #in MHZ unit
lcd_pwm_not_used      = 0
lcd_pwm_ch            = 0
lcd_pwm_freq          = 10000 #in HZ unit
```



lcd_pwm_pol	= 0	
lcd_if	= 0	#0: hv; 1:2020; 2: ttl; 3:
lvds		
lcd_hbp	= 46	
lcd_ht	= 1055	
lcd_vbp	= 23	
lcd_vt	= 1050	
lcd_vspw	= 0	
lcd_hspw	= 0	
lcd_hv_if	= 0	
lcd_hv_smode	= 0	
lcd_hv_s888_if	= 0	
lcd_hv_syuv_if	= 0	
lcd_lvds_ch	= 0	
lcd_lvds_mode	= 0	
lcd_lvds_bitwidth	= 0	
lcd_lvds_io_cross	= 0	
lcd_cpu_if	= 0	
lcd_frm	= 0	
lcd_io_cfg0	= 0x10000000	
lcd_gamma_correction_en	= 0	
lcd_gamma_tbl_0	= 0x00000000	
lcd_gamma_tbl_1	= 0x00010101	
lcd_gamma_tbl_255	= 0x00ffffff	
lcd_bl_en_used	= 1	
lcd_bl_en	= port:PH07<1><0><default><1>	
lcd_power_used	= 1	
lcd_power	= port:PH08<1><0><default><1>	
lcd_pwm_used	= 1	
lcd_pwm	=	
port:PB02<2><0><default><default>		
...		

9.3.2. 配置系统 UI 方向属性

在 android4.4/device/softwinner/wing-xxx\wing_XXX.mk 中增加
PRODUCT_PROPERTY_OVERRIDES += \
 property ro.sf.hwrotation=270
支持 0,90,180,270。默认为 0

9.4. Touch Panel 配置

发布的 SDK 中，默认有对电阻屏（两点）、FT5406/FT5506/FT5606（敦泰）、GT813/GT827/GT828（汇顶）、GSL1680/GSL2680/GSL3680(思立微)、zet622x 的支持；

9.4.1. 配置文件的修改

配置文件目录：lichee/tools/pack/chips/sun7i/configs/android/~~wing-xxx~~/sys_config.fex

RTP（电阻屏）：

```
[rtp_para]
rtp_used           = 1
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
```

建议只修改 rtp_screen_size 和 rtp_regidity_level，其他参数请暂不要修改；

CTP（电容屏）：

```
[ctp_para]
ctp_used           = 1
ctp_name           = "gt813_evb"
ctp_twi_id         = 2
ctp_screen_max_x   = 1280
ctp_screen_max_y   = 800
ctp_revert_x_flag  = 0
ctp_revert_y_flag  = 0
ctp_exchange_x_y_flag = 0

ctp_int_port       = port:PH 21 <6> <default> <default> <default>
ctp_wakeup         = port:PB 13 <1> <default> <default> <1>
```

名称	作用
ctp_used	标识是否启动 ctp
ctp_name	gslX680,GT 系类（汇定）等用于区别参数的匹配名称，其他驱动无作用
ctp_twi_id	ctp 驱动位于那一组 i2c 总线上
ctp_screen_max_x	X 轴最大分辨率
ctp_screen_max_y	Y 轴最大分辨率
ctp_revert_x_flag	X 轴反向标志
ctp_revert_y_flag	Y 轴反向标志



ctp_exchange_x_y_flag	X, Y 轴互换标志
ctp_int_port	中断引脚, 根据硬件设置进行相对应的配置
ctp_wakeup	复位引脚, 根据硬件设置进行相对应的配置

需要反置 x 方向时, 若 ctp_revert_x_flag 原值为 0 则将其设置为 1; 若 ctp_revert_x_flag 原值为 1 则将其设置为 0。

需要反置 y 方向时, 若 ctp_revert_y_flag 原值为 0 则将其设置为 1; 若 ctp_revert_y_flag 原值为 1 则将其设置为 0。

需要互换 x 轴跟 y 轴时, 若 ctp_exchange_x_y_flag 原值为 0 则将其设置为 1; 若 ctp_exchange_x_y_flag 原值为 1 则将其设置为 0。

9.4.2. Android 层的配置修改

1) 驱动的加载

在 android/device/softwinner/*wing-xxx*/init.sun7i.rc 文件中加入装载驱动模块的语句:

```
insmod /system/vendor/modules/gslX680.ko
```

电阻屏驱动的加载语句:

```
insmod /system/vendor/modules/sunxi-ts.ko
```

2) IDC 文件修改

Android4.0 之后, 配置文件中需要一个 idc 文件来识别输入设备为触摸屏还是鼠标, 如果没有该文件, 则默认为鼠标, 因此需要添加该文件。

使用 adb shell getevent 命令, 或者设备的名称为 “gslX680”, “gt82x”, “ft5x_ts”, “sunxi-ts”, “gt818_ts”, “gt811”, “gt9xx”, “sw-ts”, “zet622x” 时, 使用的 idc 名字均为 tp.idc。

idc 文件放置的目录为: system/usr/idc, 则在配置文件为 wing_xx.mk 拷贝语句如下所示:

```
PRODUCT_COPY_FILES += \
device/softwinner/wing-xxx/sw-keyboard.kl:system/usr/keylayout/sw-keyboard.kl \
device/softwinner/wing-xxx/tp.idc:system/usr/idc/tp.idc \
```

当使用 adb shell getevent 命令得到的设备名称与以上的设备名称不符合, 则需要增加该名称的 idc 文件进行相应的匹配。如使用 getevent 命令后, 获得的名称为 ctp_name, 如下:



```
root@android:/ # getevent
getevent
add device 1: /dev/input/event3
name: "ctp_name"
add device 2: /dev/input/event2
name: "bma250"
add device 3: /dev/input/event0
name: "sw-keyboard"
could not get driver version for /dev/input/mice, Not a typewriter
add device 4: /dev/input/event1
name: "axp20-supplyer"
```

图 9.4.1

则相应的 idc 文件就应该为 ctp_name.idc,则在配置文件为 wing_xx.mk 拷贝语句如下所示:

```
#input device config
PRODUCT_COPY_FILES += \
    device/softwinner/wing-xxx/sw-keyboard.kl:system /usr/keylayout/sw-keyboard.kl \
    device/softwinner/wing-xxx/ctp_name.idc:system /usr/idc/ctp_name.idc \
    device/softwinner/wing-xxx/gsensor.cfg:system /usr/gsensor.cfg
```

9.4.3. touch panel 驱动使用说明

1) gslX680 使用说明

gslX680 驱动兼容 gsl1680, gsl2680, gsl3680。为了区分下载的参数,在 sys_config.fex 的配置文件中,需要增加 ctp_name 进行区别,目前, gslX680 系列的参数设置方式为每一种分辨率或者是一组参数设置为一个.h 文件,使用 ctp_name 进行区分,使用时请注意项目中使用的头文件。如使用的参数为“gsl168.h”,则 sysconfig.fex 中的参数如下所示:

```
[ctp_para]
ctp_used                = 1
ctp_twi_id              = 2
ctp_name                 = "gsl1680"
ctp_screen_max_x        = 1024
ctp_screen_max_y        = 600
```

使用时注意驱动中已经支持的参数是否跟当前使用的 tp 匹配,若不匹配将无法正常使用,如何增加一组新增参数,请仔细阅读“A20 平台 CTP 模块开发说明文档.doc”中的 6.4 节。

当更换参数时,需要替换 sysconfig.fex 中的 ctp_name 找到相对应的参数。找不到时驱动将退出加载。

2) GT 系列(汇顶)使用说明

Gt 系列的驱动包括 gt811, gt82x (gt813, gt827, gt828), gt9 系列。gt 系列的产品在驱动端初始化时需要根据具体的 tp 屏下载相应的参数之后才可以正常的工作,在掉电之后也需要通过驱动端重新下载相关的参数。



为了兼容多款 tp 面板，多种分辨率，目前将 gt 系列的参数抽取出来放置单独的头文件中，通过名字进行匹配的方法进行参数的选择。如果驱动中没有找到相对应的匹配名字，将使用第 0 组参数。

gt82x.ko 驱动为兼容 gt813，gt827，gt828 这三颗 IC 的驱动。gt82x 对应的参数头文件：lichee/linux-3.x/drivers/input/sw_touchscreen/gt82x.h

gt811.ko 驱动为 gt811 这颗 IC 的驱动。头文件中放置了两组参数，gt811 对应的头文件：lichee/linux-3.x/drivers/input/sw_touchscreen/gt811_info.h

gt9xx_ts.ko 驱动为 gt9 系列对应的驱动。头文件中放置了两组 gt911 使用的参数。gt9xx 对应的头文件：lichee/linux-3.x/drivers/input/sw_touchscreen/gt9xx_info.h

通过 ctp_name 进行区别使用的参数，请先查看驱动中已经支持的参数跟目前使用的 tp 是否符合。

如 evb 中使用 gt813，相对应的参数在 gt82x.h 中的“gt813_evb”数组中。则 sysconfig.fex 中的参数如下所示：

```
[ctp_para]
ctp_used           = 1
ctp_twi_id         = 2
ctp_name           = "gt813_evb"
ctp_screen_max_x   = 1024
ctp_screen_max_y   = 800
```

使用时注意驱动中已经支持的参数是否跟当前使用的 tp 匹配，若不匹配将可能造成无法正常使用等异常情况，如何增加一组新增参数，请仔细阅读“A20 平台 CTP 模块开发说明文档.doc”中的 6.5 节。

当更换参数时，需要替换 sysconfig.fex 中的 ctp_name 找到相对应的参数，如果没有找到匹配的参数，将默认下载第 0 组参数。

3) ft 系类驱动使用说明

Ft5x02 系列的相关说明：

ft5x02 使用时需要 tp 相关的头文件信息即使用 tp 的 ft5x02_config.h 文件。如果该文件为拷贝过来，则请注意头文件中定义的名称是否与原来的文件一致，特别是文件中定义的变量名称的大小写。

驱动中通过读取 a3 寄存器，通过对其值的判断确定是否为 0x02，如果为 0x02，则说明为 02 系列。此时将通过驱动下载 ft5x02_config.h 相关的参数。当 ic 掉电之后重新上电也需要下载参数。

当发现 tp 无法正常读取数据时，请确认相关的参数是否已经正确的下载。

Ft5x06 的相关说明：

当需要更新固件时，可以打开驱动的 CONFIG_SUPPORT_FTS_CTP_UPG 的定义，通过下载 i 文件去下载固件。正常情况下该定义被屏蔽掉，当确认需要下载时，请打开该宏定义且请更换正确的点 i 文件，否则将造成 tp 无法正常使用的情况。

9.5. G Sensor 配置

发布的 SDK 中已添加了对 MMA7660、MMA8452、MMA8652、MMA8653、LIS3DH、LIS3DE、AFA750、及 BMA250 、BMA250e 等 G-Sensor 的支持，需要客户根据需要做如下配置：

9.5.1. 打包配置文件修改

配置文件目录：lichee/tools/pack/chips/sun7i/configs/android/**wing-xxx**/sys_config.fex
G sensor 的配置文件事例如下。

```
[gsensor_para]
gsensor_used      = 1
gsensor_twi_id    = 1
gsensor_int1      =
gsensor_int2      =
```

只需要配置 gsensor_used 与 gsensor_twi_id 即可，gsensor_used 代表是否支持 gsensor， gsensor_twi_id 代表 I2C 总线号。

9.5.2. Android 层配置修改

以 MMA8452 为例：

1) Android 中，在 android/device/softwinner/**wing-xxx**/init.sun7i.rc 文件中加入装载驱动模块的语句：

```
insmod /system/vendor/modules/mma8452.ko
```

2) 方向的调整：

在 android/device/softwinner/**wing-xxx**/gsensor.cfg 中，以 mma8452 方向为例进行说明。

```
gsensor_name = mma8452    //标示用 mma8452 gsensor
gsensor_direct_x = true//如果 x 轴反向，则置 false
gsensor_direct_y = true//如果 y 轴反向，则置 false
gsensor_direct_z = false//如果 z 轴反向，则置 false
gsensor_xy_revert = true//如果 x 轴当 y 轴用，y 轴当 x 轴，则置 true
```

MMA8452 实际方向如下（参见 8452 Datasheet）：



Gsensor 方向调试说明:

假定机器的长轴为 X 轴，短轴为 Y 轴，垂直方向为 Z 轴。

首先调试 Z 轴:

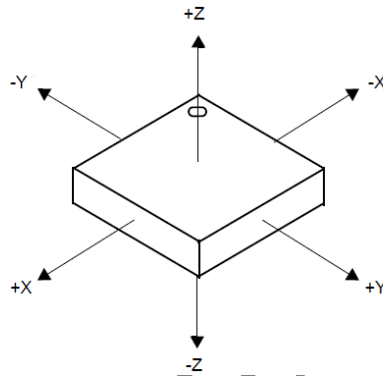
第一步观察现象:

旋转机器，发现当只有垂直 90° 时或者是在旋转后需要抖动一下，方向才会发生变化，则说明 Z 轴反了。若当机器大概 45° 拿着的时候也可以旋转，说明 Z 轴方向正确。无需修改 Z 轴方向。

第二步修改 Z 轴为正确方向。

此时需要找到当前使用模组的方向向量（根据模组的名称）。如果此时该方向 Z 轴向量（gsnesor_direct_z）的值为 false，则需要修改为 true；当为 true，则需要修改为 false。通过 adb shell 将修改后的 gsnesor.cfg 文件 push 到 system/usr 下，重启机器，按第一步观察现象。

其次查看 X，Y 轴是否互换:



第一步观察现象:

首先假定长轴为 X 轴，短轴为 Y 轴，以 X 轴为底边将机器立起来。查看机器的 X，Y 方向是否正好互换，若此时机器的 X，Y 方向正好互换，在说明需要将 X，Y 方向交换。若此时 X，Y 方向没有反置，则进入 X，Y 方向的调试。

第二步 交换 X，Y 方向

当需要 X，Y 方向交换时，此时需要找到当前使用模组的方向向量（根据模组的名称）。如果此时该 X，Y 轴互换向量（gsensor_xy_revert）的值为 false，则需要修改为 true，当为 true，则需要修改为 false。通过 adb shell 将修改后的 gsnesor.cfg 文件 push 到 system/usr 下，重启机器，按第一步观察现象。

再次调试 X，Y 轴方向:

第一步观察现象:

首先假定长轴为 X 轴，短轴为 Y 轴，以 X 轴为底边将机器立起来，查看机器的方向是否正确，如果正确，说明长轴配置正确，如果方向正好相反，说明长轴配置错误。将机器旋转到短轴，查看机器方向是否正确，如果正确，说明短轴配置正确，如果方向正好相反，说明短轴配置错误。

第二步修改 X，Y 轴方向:

当需要修改 X，Y 轴方向时，当只有长轴方向相反或者是只有短轴方向相反时，则只修改方向不正确的一个轴，当两个方向都相反时，则同时修改 X 与 Y 轴方向向量。找到当前使用模组的方向向量（根据模组的名称）。



若长轴方向相反，如果此时该方向 X 轴向量（gsnesor_direct_x）的值为 false，则需要修改为 true，当为 true，则需要修改为 false。

若短轴方向相反，如果此时该方向 Y 轴向量（gsnesor_direct_y）的值为 false，则需要修改为 true，当为 true，则需要修改为 false。

通过 adb shell 将修改后的 gsnesor.cfg 文件 push 到 system/usr 下，重启机器，按第一步观察现象。若发现还是反向 X 轴或者 Y 轴的方向仍然相反，则说明 X 轴为短轴，Y 轴为长轴。此时：

若长轴方向相反，如果此时该方向 Y 轴向量（gsnesor_direct_y）的值为 false，则需要修改为 true，当为 true，则需要修改为 false。

若短轴方向相反，如果此时该方向 X 轴向量（gsnesor_direct_x）的值为 false，则需要修改为 true，当为 true，则需要修改为 false。

9.6. Camera 配置

发布的 SDK 中已添加了对 gc0307, gc0308, gc2035, gt2005, hi253, ov5640, s5k4ec 的支持；

9.6.1. 打包配置文件修改

配置文件位置：lichee/tools/pack/chips/sun7i/configs/android/~~wing-xxx~~/sys_config.fex

从 V1.3 及以后版本，CSI 驱动中添加了 SENSOR 自检测功能。

```
[[camera_list_para]
camera_list_para_used = 1
ov7670                = 0
gc0308                = 1
gt2005                = 0
hi704                 = 0
sp0838               = 0
mt9m112              = 0
mt9m113              = 0
ov2655               = 0
hi253                 = 0
gc0307               = 0
mt9d112              = 0
ov5640               = 1
gc2015               = 0
ov2643               = 0
gc0329               = 0
gc0309               = 0
tvp5150              = 0
```



s5k4ec	= 0
ov5650_mv9335	= 0
siv121d	= 0

配置项	配置项含义
camera_list_para_used	Camera 自适应功能: 1 打开 0: 关闭
xxx(比如 gc0308) = 1	选择需要自检测的 sensor

A20 上面使用了 CSI0 和 CSI1，驱动里面需要配置[csi0_para]，[csi1_para]段落。

[csi0_para]		
csi_used	= 1	
csi_dev_qty	= 1	
csi_stby_mode	= 0	
csi_mname	= "gc0308"	
csi_if	= 0	
csi_iovdd	= ""	
csi_avdd	= ""	
csi_dvdd	= ""	
csi_vol_iovdd	=	
csi_vol_dvdd	=	
csi_vol_avdd	=	
csi_vflip	= 0	
csi_hflip	= 0	
csi_flash_pol	= 0	
csi_facing	= 0	
csi_twi_id	= 1	
csi_twi_addr	= 0x42	
csi_pck	=	
port:PE00<3><default><default><default>		
csi_ck	=	
port:PE01<3><default><default><default>		
csi_hsync	=	
port:PE02<3><default><default><default>		
csi_vsync	=	



port:PE03<3><default><default><default>	
csi_d0	=
port:PE04<3><default><default><default>	
csi_d1	=
port:PE05<3><default><default><default>	
csi_d2	=
port:PE06<3><default><default><default>	
csi_d3	=
port:PE07<3><default><default><default>	
csi_d4	=
port:PE08<3><default><default><default>	
csi_d5	=
port:PE09<3><default><default><default>	
csi_d6	=
port:PE10<3><default><default><default>	
csi_d7	=
port:PE11<3><default><default><default>	
csi_reset	= port:PH13<1><default><default><0>
csi_power_en	=
csi_stby	= port:PH16<1><default><default><0>

根据使用的 Camera 型号来设置如上子项,根据原理图来设置 reset、power 及 standby 引脚的 gpio 和控制逻辑;

配置项	配置项含义
csi_used =xx	是否使用 csi0
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



	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）

9.6.2. Android 层的配置修改

Android 中，在 android/device/softwinner/*wing-xxx*/init.sun7i.rc 文件中加入装载驱动模块的语句：

```
单摄像头：
insmod /system/vendor/modules/videobuf-core.ko
insmod /system/vendor/modules/videobuf-dma-contig.ko
insmod /system/vendor/modules/gc0308.ko
insmod /system/vendor/modules/sunxi_csi0.ko
```

在 android/device/softwinner/*wing-xxx*/ueventd.sun7i.rc 文件中改变相关设备节点的权限：

```
/dev/video0          0666    media    media
```

9.6.3 Camera 参数配置

配置文件路径：android/device/softwinner/*wing-xxx*/camera.cfg

事例内容简介：

```
number_of_camera = 1      #camer 模块的数量(1/2)
camera_id = 0
camera_facing = 0         #1：前置摄像头；0 后置摄像头
```



```
camera_orientation = 0    #camer 模块的方向(0/90/180/270)
camera_device = /dev/video0    #设备文件接口
device_id = 0    #设备 id, 两个 camera 使用一个 CSI
used_preview_size = 1
key_support_preview_size = 640x480
key_default_preview_size = 640x480
used_picture_size = 1
key_support_picture_size = 640x480
key_default_picture_size = 640x480
used_flash_mode = 0
key_support_flash_mode = on,off,auto
key_default_flash_mode = on
used_color_effect=1
key_support_color_effect = none,mono,negative,sepia,aqua
key_default_color_effect = none
used_frame_rate = 1
key_support_frame_rate = 25
key_default_frame_rate = 25
used_focus_mode = 0
key_support_focus_mode = auto,infinity,macro,fixed
key_default_focus_mode = auto
used_scene_mode = 0
key_support_scene_mode = auto,auto,portrait,landscape,night,night-portrait,theatre,beach,snow,
sunset,steadyphoto,fireworks,sports,party,candlelight,barcode
key_default_scene_mode = auto
used_white_balance = 1
key_support_white_balance = auto,incandescent,fluorescent,warm-fluorescent,daylight,cloudy-day
light
key_default_white_balance = auto
used_exposure_compensation = 1
key_max_exposure_compensation = 3
key_min_exposure_compensation = -3
key_step_exposure_compensation = 1
key_default_exposure_compensation = 0
; only for facing back camera in android2.3, should be set in
android4.0
used_zoom = 1
key_zoom_supported = true
key_smooth_zoom_supported = false
key_zoom_ratios = 100,120,150,200,230,250,300
```



```
key_max_zoom = 30
key_default_zoom = 0
camera_orientation = 0
...
```

media_profiles.xml 的路径: android/device/softwinner/~~wing-xxx~~/media_profiles.xml

内容简介: 该文件主要保存 Camera 支持的摄像相关参数, 包括摄像质量, 音视频编码格式、帧率、比特率等等, 该参数主要有摄像头厂商提供: (以下 Demo 对应两个摄像头的情况, 如果只有一个 camera 则只需要一份参数)

```
<MediaSettings>
  <CamcorderProfiles>
    <EncoderProfile    quality="480p"    fileFormat="mp4"
duration="60">
      <Video codec="h264"
        bitRate="1000000"
        width="640"
        height="480"
        frameRate="30" />
      <Audio codec="aac"
        bitRate="12200"
        sampleRate="44100"
        channels="1" />
    </EncoderProfile>
    <EncoderProfile    quality="timelapse480p"
fileFormat="mp4" duration="60">
      <Video codec="h264"
        bitRate="1000000"
        width="640"
        height="480"
        frameRate="30" />
      <Audio codec="aac"
        bitRate="12200"
        sampleRate="44100"
        channels="1" />
    </EncoderProfile>
    <ImageEncoding quality="90" />
    <ImageEncoding quality="80" />
    <ImageEncoding quality="70" />
    <ImageDecoding memCap="20000000" />
  </CamcorderProfiles>
</MediaSettings>
```



```

    <Camera previewFrameRate="0" />
</CamcorderProfiles>
<EncoderOutputFileFormat name="mp4" />
<VideoEncoderCap name="h264" enabled="true"
    minBitRate="64000" maxBitRate="3000000"
    minFrameWidth="320" maxFrameWidth="640"
    minFrameHeight="240" maxFrameHeight="480"
    minFrameRate="1" maxFrameRate="30" />

<AudioEncoderCap name="aac" enabled="true"
    minBitRate="5525" maxBitRate="12200"
    minSampleRate="8000" maxSampleRate="44100"
    minChannels="1" maxChannels="1" />

<AudioEncoderCap name="amrwb" enabled="true"
    minBitRate="6600" maxBitRate="23050"
    minSampleRate="16000" maxSampleRate="16000"
    minChannels="1" maxChannels="1" />

<AudioEncoderCap name="amrnb" enabled="true"
    minBitRate="5525" maxBitRate="12200"
    minSampleRate="8000" maxSampleRate="8000"
    minChannels="1" maxChannels="1" />
<VideoDecoderCap name="wmv" enabled="true"/>
<AudioDecoderCap name="wma" enabled="true"/>
<VideoEditorCap maxInputFrameWidth="1920"
    maxInputFrameHeight="1080"
maxOutputFrameWidth="1920"
    maxOutputFrameHeight="1080"
maxPrefetchYUVFrames="10"/>
    <ExportVideoProfile name="m4v" profile="1" level="128"/>
</MediaSettings>

```

9.6.4 Camera 预览界面自适用配置

配置文件路径: \device\softwinner\wing-xxx\cfg-Gallery2.xml:

文件内容如下:

```

<?xml version="1.0" encoding="utf-8"?>
<!-- used by AWGallery.apk -->
<cfgOverrides>
    <!-- reverse 1:enabled 0:disabled -->

```

```
<cfgOverride
    id="0"
    name="fullscreen"
    value="1"/>
</cfgOverrides>
```

value=1 预览全屏显示。

valve=0 根据预览尺寸自适应显示。

9.7. 震动马达配置

震动马达部分的电路比较单一，软件端只用控制 CHGLED 引脚的电平就可以打开和关闭马达震动；

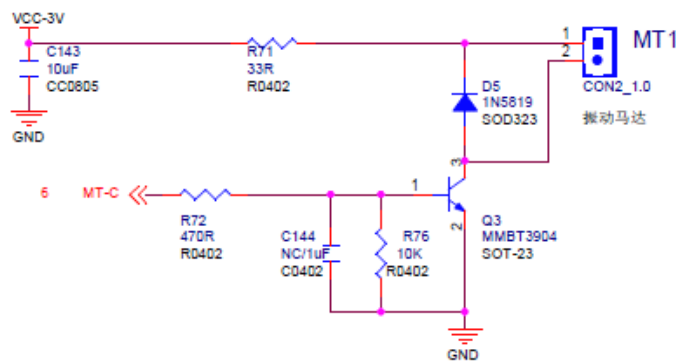


图 9.7.1

9.7.1. 配置文件修改

配置文件路径: `lichee/tools/pack/chips/sun7i/configs/android/wing-xxx/sys_config.fex`
根据硬件原理图进行相关参数的配置

```
[motor_para]
motor_used          = 1  #是否启用马达，启用置 1，反之置 0
motor_shake         = port:power3<1><default><default><1>
#马达震动的 GPIO 选择及控制逻辑：0 代表低电平关闭，高电平打开；1 代表高电平关闭，低电平打开，请根据实际电路图来配置；
```

9.7.2. Android 层配置修改

在 android/device/softwinner/*wing-xxx*/init.sun7i.rc 文件中加入改变相关设备节点及装载驱动模块的语句：

```
# insmod vibrator
insmod /system/vendor/modules/sun7i-vibrator.ko
chmod 777 /sys/class/timed_output/sun7i-vibrator/enable
```

9.8. SD 卡配置

发布的 SDK 中支持 SD 卡和 Mirco SD（TF）卡及其兼容性卡，A20 中支持四组通用的 mmc/sd 卡接口，其基本电路为：

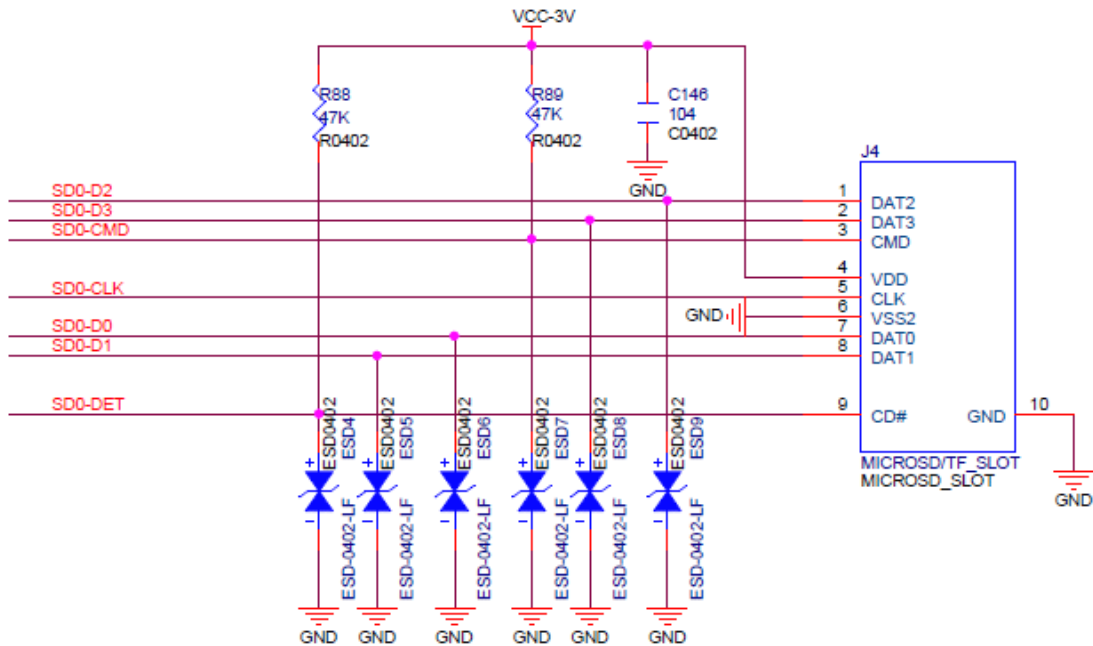


图 9.8.1

9.8.1. 配置文件的修改

配置文件路径：

lichee/tools/pack/chips/sun7i/configs/android/*wing-xxx*/sys_config.fex

根据原理图进行相关配置参数的修改

```
[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:PH1<0><1><default><default>
sdc_use_wp	= 0
sdc_wp	=
sdc_isio	= 0
sdc_regulator	= "none"

参数的意义:

sdc_used, 表示 mmc 是否使用

sdc_detmode, 表示检测模式, 1: 使用 GPIO 轮询检测, 2: GPIO 中断检测

3: 不检测 (作为引导卡), 4: 手动插入和删除, 5: data3 检测

sdc_buswidth, MMC 总线宽度, 1: 1-bit, 4: 4-bit, 8: 8-bit

接下来七个选项用来配置 SDIO 的时钟线、命令线、d0-d3 数据线和 det 线的引脚配置根据实际硬件原理进行配置。

sdc_use_wp, 写保护, 1: 有写保护, 0: 没有写保护

sdc_isio, SDIO Card

sdc_regulator, 电源控制

注意: 如果 sdc_detmode 的值为 5 (选择 data3 检测), 在硬件上, 需要在 DAT3 上加一个约 1MΩ 下拉电阻, 同时 sdc_d3 需要保持高阻态。

9.9. CTP 与 SENSOR 自动检测使用说明

ctp 与 sensor 的自动检测驱动源文件为: \lichee\linux-3.3\drivers\input\sw_device.c, 自动检测的驱动为 sw_device.ko, 该驱动中目前已经支持 ctp 与 gsensor 的自动检测。

ctp 与 sensor 自动检测说明以及如何添加新的模组进去, 请仔细阅读文档

“ctp 与 sensor 自动检测使用文档.doc”。以下列出使用的关键步骤。

9.9.1. CTP 自适应使用说明

1) sysconfig.fex 文件的配置

第一步: 若 sysconfig.fex 文件中不存在 ctp_list_para 配置项, 请增加该配置项。

该配置项的内容为 sw_device.c 文件中 ctps 数组中支持的检测模组。Sysconfig.fex 文件中模组的名称与 sw_device.c 中的名称一一对应。ctp_list_para 配置项如下所示:

```
[ctp_list_para]
ctp_det_used          = 1
ft5x_ts               = 1
```



```
gt82x          = 1
gs1X680        = 1
gt9xx_ts       = 1
gt811          = 1
zet622x        = 1
```

模组后写 1 表示支持该模组的自动检测，模组后写 0 表示不支持该模组的自动检测。当确认方案中不使用该模组或者存在地址冲突无法区别的模组时将该模组写 0。

第二步：ctp_para 下的 ctp_used 以及 ctp_list_para 下的 ctp_det_used 必须写 1，否则将退出自动检测。

2) 自动检测驱动的加载

在 android4.X\device\softwinner\wing-XXX\init.XXX.rc 中添加驱动加载的模块，为了快速的检测到设备，此语句应该放置在模块加载的最前面，如下所示：

```
.....
on boot
#use automatic detecttion insmod ctp & gsensor driver
    insmod /system/vendor/modules/sw_device.ko
#insmod video driver
    insmod /system/vendor/modules/cedarx.ko
.....
```

注意：使用自动检测时，加载了 sw_device.ko 后，在 init.XXX.rc 文件中需要删除掉之前已经加载的 ctp 驱动的句子。

9.9.2. GSENSOR 自适应使用说明

1) sysconfig.fex 文件的配置

第一步：若 sysconfig.fex 文件中不存在 gsensor_list_para 配置项，请增加该配置项。该配置项的内容为 sw_device.c 文件中 gsenosrs 数组中支持的检测模组。sysconfig.fex 文件中模组的名称与 sw_device.c 中的名称一一对应。

gsneosr_list_para 配置项如下所示：

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

模组后写 1 表示支持该模组的自动检测, 模组后写 0 表示不支持该模组的自动检测。
当确认方案中不使用该模组或者存在地址冲突无法区别的模组时将该模组写 0。

第二步: gsneoser_para 下的 gsensor_used 以及 gsensor_list_para 下的 gsensor_det_used 必须写 1, 否则将退出自动检测。

2) 自动检测驱动的加载

在 android4.X\device\softwinner\wing-XXX\init.XXX.rc 中添加驱动加载的模块, 为了快速的检测到设备, 此语句应该放置在模块加载的最前面, 如下所示:

```
.....
on boot
#use automatic detecttion insmod ctp & gsensor driver
    insmod /system/vendor/modules/sw_device.ko
#insmod video driver
    insmod /system/vendor/modules/cedarx.ko
.....
```

注意: 使用自动检测时, 加载了 sw_device.ko 后, 在 init.XXX.rc 文件中需要删除掉之前已经加载的 gsneoser 驱动的句子。

9.9.3. Recovery 功能 tp 的自适应使用说明

1) 默认情况下自动检测加载 ctp 驱动

Recovery 功能时, ctp 同样可以使用自动检测驱动进行相关的检测, 默认情况下 recovery 功能使用自动检测的功能进行相应的 ctp 驱动的添加。ctp 的自适应主要的工作为将驱动拷贝到 recover 的 root 下。

在在 android4.X\device\softwinner\wing-xxx\wing_xxx.mk 文件中。XXX 为相对应的配置文件目录。如下:

```
.....
# for recovery
```



```
PRODUCT_COPY_FILES += \
```

```
device/softwinner/wing-xxx/recovery.fstab:recovery.fstab \
device/softwinner/wing-xxx/modules/modules/disp.ko:disp.ko \
device/softwinner/wing-xxx/modules/modules/lcd.ko:lcd.ko \
device/softwinner/wing-xxx/modules/modules/hdmi.ko:hdmi.ko \
device/softwinner/wing-xxx/modules/modules/gt82x.ko:gt82x.ko \
device/softwinner/wing-xxx/modules/modules/gt811.ko:gt811.ko \
device/softwinner/wing-xxx/modules/modules/ft5x_ts.ko:ft5x_ts.ko \
device/softwinner/wing-xxx/modules/modules/zet622x.ko:zet622x.ko \
device/softwinner/wing-xxx/modules/modules/gslX680.ko:gslX680.ko \
device/softwinner/wing-xxx/modules/modules/gt9xx_ts.ko:gt9xx_ts.ko \
device/softwinner/wing-xxx/modules/modules/sw_device.ko:sw_device.ko
```

.....

其中 recovery.fstab 为 recovery 功能相关文件。

disp.ko, lcd.ko, hdmi.ko, 为显示相关驱动。

gt82x.ko, gt811.ko, ft5x_ts.ko, zet622x.ko, gslX680.ko, gt9xx_ts.ko 等为目前已经支持的 ctp 驱动。sw_device.ko 为自动检测驱动。

自动检测驱动 (sw_device.c) 文件中增加了新的 ctp 模组, 在将相关的 ctp 驱动拷贝到 recovery 的 root 目录下, 当 recovery 功能时使用。

如增加新的 ctp 驱动名称为 screen.ko, 则增加的语句如下所示:

.....

```
# for recovery
```

```
PRODUCT_COPY_FILES += \
```

```
device/softwinner/wing-xxx/recovery.fstab:recovery.fstab \
device/softwinner/wing-xxx/modules/modules/disp.ko:disp.ko \
device/softwinner/wing-xxx/modules/modules/lcd.ko:lcd.ko \
device/softwinner/wing-xxx/modules/modules/hdmi.ko:hdmi.ko \
device/softwinner/wing-xxx/modules/modules/gt82x.ko:gt82x.ko \
device/softwinner/wing-xxx/modules/modules/gt811.ko:gt811.ko \
device/softwinner/wing-xxx/modules/modules/ft5x_ts.ko:ft5x_ts.ko \
device/softwinner/wing-xxx/modules/modules/zet622x.ko:zet622x.ko \
device/softwinner/wing-xxx/modules/modules/gslX680.ko:gslX680.ko \
device/softwinner/wing-xxx/modules/modules/gt9xx_ts.ko:gt9xx_ts.ko \
device/softwinner/wing-xxx/modules/modules/screen.ko:screen.ko \
```

```
device/softwinner/wing-xxx/modules/modules/sw_device.ko:sw_device.ko
```

.....

2) 不使用自动检测功能时加载相关的 tp 驱动修改方法

第一步：驱动拷贝到 recovery 的 root，按照自动检测方法进行相关驱动的拷贝。

第二步：修改 android4.2.1\bootable\recovery\etc\init.rc 中的相关语句，加载使用的 tp 驱动。如需要加载的驱动为 sunxi-ts.ko，相关的修改如下：

.....

```
on init
```

```
export PATH /sbin
```

```
export ANDROID_ROOT /system
```

```
export ANDROID_DATA /data
```

```
export EXTERNAL_STORAGE /sdcard
```

```
insmod /nand.ko
```

```
//sw_device.ko 自动检测驱动的加载，可将其去掉
```

```
insmod /sw_device.ko debug_mask=0xff ctp_mask=1
```

```
insmod /sunxi-ts.ko //加载相对应的 tp 驱动
```

```
insmod /disp.ko
```

```
insmod /lcd.ko
```

```
insmod /hdmi.ko
```

.....

9.10. 安全控制配置

9.10.1. ADB 安全控制

在编译后生成的 android/out/target/product/**wing-xxx**/system/build.prop 文件中记录了 adb secure 如下：

```
ro.adb.secure=1
```

添加 adb 安全控制方法：源码中修改 android/device/softwinner/wing-xxx/wing_xxx.mk，添加如下：

```
PRODUCT_PROPERTY_OVERRIDES += \
ro.adb.secure=1
```

打开 adb 安全控制，需要用户点击确认才能连接 adb。

等效于在 build.prop 文件中修改

```
ro.adb.secure=1
```

去掉该项配置 ro.adb.secure 即可去掉 adb 安全控制

10. Settings 与 Launcher 设置

10.1. 默认 LCD 关闭时间设置

修改

android/device/softwinner/wing-common/overlay/frameworks/base/packages/SettingsProvider/res/values/defaults.xml 中如下设置（单位为秒）：

```
<integer name="def_screen_off_timeout">60000</integer>
```

10.2. 默认亮度设置

修改

android/device/softwinner/wing-common/overlay/frameworks/base/packages/SettingsProvider/res/values/defaults.xml 中如下设置，亮度值从 0~255 表示 0%~100%，如设置 102 则默认亮度为 40%，示例如下：

```
<integer name="def_screen_brightness">102</integer>
```

10.3. 默认字体大小设置

系统字体由 fontScale 来控制缩放，设置菜单中的小，普通，大，超大分别对应的 fontScale 为 0.85、1.0、1.15、1.3，修改默认字体大小可在 android/device/softwinner/wing-xxx.mk 中设置 ro.font.scale 的值来设置默认的字体的大小。如设置为大字体为：

```
PRODUCT_PROPERTY_OVERRIDES +=  
    ro.font.scale=1.15 \
```

注：建议采用默认设置，即 ro.font.scale=1.0，过 CTS 默认必须是 1.0。

10.4. 电池电量百分比显示

电池电量默认是开启的，如果方案不需要可以将其默认 disable 掉，修改文件：

android/device/softwinner/wing-xxx/overlay/frameworks/base/packages/SystemUI/res/values/config.xml，添加此项的 overlay 配置，如下：

```
<!-- Whether or not we show battery text in the bar -->  
<bool name="config_statusBarShowBatteryText">false</bool>
```

10.5. 默认墙纸设置

更换默认墙纸替换方案 overlay 文件：

android/device/softwinner/wing-xxx/overlay/frameworks/base/core/core/res/drawable-swxxx-dp-nodpi/default_wallpaper.jpg

10.6. 添加薄纸

- 准备壁纸及壁纸的缩略图放进壁纸存放目录
android/packages/apps/Launcher2/res/drawable-swxxxdp-nodpi，并按照文件夹内文件命名，分别为 wallpaper_xxx.jpg 与 wallpaper_xxx_small.jpg
- 在 android/packages/apps/Launcher2/res/values-swxxxdp/wallpapers.xml 中添加该壁纸索引，如下：

```
<resources>
    <string-array name="wallpapers" translatable="false">
        <item>wallpaper_00</item>
        <item>wallpaper_01</item>
        <item>wallpaper_02</item>
        <item>wallpaper_03</item>
        <item>wallpaper_04</item>
        <item>wallpaper_05</item>
        <item>wallpaper_06</item>
        <item>wallpaper_07</item>
        <item>wallpaper_08</item>
        <item>wallpaper_09</item>
        <item>wallpaper_10</item>
        <item>wallpaper_xxx</item>
    </string-array>
</resources>
```

10.7. Launcher 桌面默认图标和快捷栏设置

快捷栏图标设置修改 overlay 文件：

android/device/softwinner/wing-xxx/overlay/packages/apps/Launcher2/res/xml-swxxxdp/default_workspace.xml

其中桌面默认设置分为 Left screen，Middle screen，Right screen 等 3 屏的设置，可自行增加，在设置中需要设置需要注意一下几个参数：

设置名	意义
packageName	所运行的 APP 的 package 名，可到具体路径去查找，如在 Setting.java 中第一行有效代码会显示包名 package com.android.settings;
className	点击需要启动的该 APP 的 Activity 的 class 名，其表示方式如下 packageName.ActivityName
screen	表示在第几个，根据显示的个数决定
x	放在该屏的第几行
y	放在该屏的第几列

快捷栏 Hotseat 设置，在同样的文件中，同样修改 Hotseat 部分配置 packageName, className, screen, x 即可，其中保持 screen 和 x 值一致，系统默认最多放置 9 个图标，4 为中心图标即 Launcher 的应用界面，往左 x 设置依次为 3,2,1,0，往右 x 设置依次为 5,6,7,8。如果修改了默认的图标个数，此方法依次类推，从中心值往两边递减。

10.8. 下拉菜单 QuickSetting 中的开关显示

下拉菜单中机主信息，移动信号，蓝牙，飞行模式可以通过配置控制其是否显示，可配置文件

android/device/softwinner/wing-xxx/overlay/frameworks/base/packages/SystemUI/res/values/config.xml 中如下设置（true 显示，false 不显示）：

```
<bool name="quick_settings_show_userinfo">true</bool>
<bool name="quick_settings_show_airplane_switch">true</bool>
<bool name="quick_settings_show_bluetooth_setting">true</bool>
<bool name="quick_settings_show_rssiinfo">true</bool>
```

10.9. Settings 里面的蓝牙选项

要在 Settings.apk 中设置菜单中显示“蓝牙”选项，需要修改 android/device/softwinner/wing-xxx/wing_xxx.mk，添加以下信息：

```
PRODUCT_COPY_FILES += \
frameworks/native/data/etc/android.hardware.bluetooth.xml:system/etc/permissions/android.hardware.bluetooth.xml
```

10.10. Miracast 功能打开和关闭

修改

android/device/softwinner/wing-xxx/overlay/frameworks/base/core/res/res/values/config.xml 中如下设置（true 使能，false 不使能）。

```
<bool name="config_enableWifiDisplay">false</bool>
```

注：Miracast 只能在 1GB DDR 的方案上实现，并保证 BoardConfig.mk 中的 ion_reserve 为 100MB 以上。

10.11. 快速开关机功能使能和关闭

修改 android/device/softwinner/wing-xxx/wing_xxx.mk 中增加如下设置（true 为使能，false 为不使能）：

```
PRODUCT_PROPERTY_OVERRIDES += \
ro.sys.bootfast=true
```

注：使能情况下在设置-辅助功能中有快速开关机使能选项，选择后使能下次选用快速开关机模式。

11. Declaration

This(A20 Android 开发手册) is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.