全志科技
Allwinner Technology

H6 Android N

定制化文档

0.8

2017.04.07

文档履历

版本号	日期	制/修订人	内容描述
0.8	2017.04.07		

目录

1. 前言

1.1 编写目的

本文档介绍 H6 方案中 AndroidN 系统的常见的定制开发问题，以帮助客户快速熟悉开发环境，加快产品上市。

1.2 适用范围

系统定制化开发，系统调试

1.3 相关人员

Android 系统工程师、应用工程师

1.4 相关术语

- homlet：面向客厅设备的产品线的名称。
- SDK：全志定制环境总称
- petrel：基于全志 H6 芯片的家庭娱乐产品代号
- AOSP：Android 系统开源代码项目
- CTS：Android 设备兼容性认证

2. SDK 概述

2.1 硬件资源

- petrel 样机验证板一块
- 串口线一根（TTL 电平）
- 双头 USB Type-A 连接线一根
- HDMI 连接线一根
- 12V 电源适配器一个

  硬件上电连接方式如下, 三个按键依次是：reset 按键、休眠唤醒按键、uboot 按键,TTL 串口线可以只接 4 针，从左到右依次是 TXD、RXD、VCC、GND, 开发主机建议使用 I7/E5 以上配置, 硬盘容量》500G、内存容量》8G

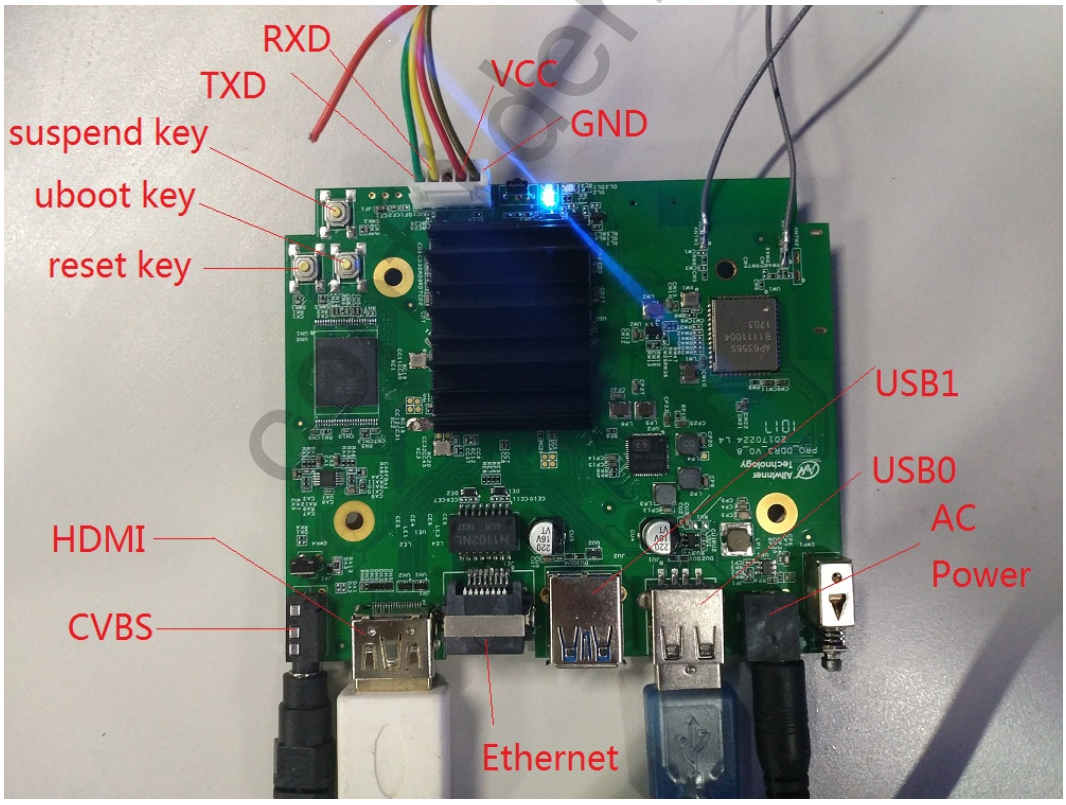

图 1

2.2 软件资源

开发环境强烈建议使用 Ubuntu12.04 以上的版本。避免不必要的编译错误

2.2.1 安装 JDK

AndroidN 开发只能使用 openjdk8 的版本，高于或低于此版本以及 oracle 的 JDK 都会导致编译错误失败 openjdk-8 的安装方法有两种，一是通过命令

```
sudo apt-get install openjdk-8-jdk
```

二是下载已经打包好的 java-8-openjdk-amd64 archives 然后自行设置环境变量。这种方法可以实现多套 JDK 共存对于第一种方法，需要注意如果已经安装了其他的 JDK 版本，需要彻底删除，通过命令切换版本的方法并不能完全保证工具一致，推荐使用第二种方法安装，建议安装到如下路径：

```
/usr/lib/jvm/java-8-openjdk-amd64
```

在此路径下编译前，编译脚本会自动设置编译环境的 JDK 为 OpenJDK8

2.2.2 安装平台支持软件

Ubuntu 12.04 平台使用如下命令安装

```
sudo apt-get install git gnupg flex bison gperf build-essential \
  zip curl libc6-dev libncurses5-dev:i386 x11proto-core-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
```

Ubuntu 14.04 及以上版本使用如下命令安装

```
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 x11proto-core-dev libx11-dev lib32z-dev ccache \
 libgl1-mesa-dev libxml2-utils xsltproc unzip
```

安装 uboot 工具软件

```
sudo apt-get install u-boot-tools
```

2.2.3 安装 phoenixSuit

phoenixSuit 软件可以在 SDK 代码以下路径找到 lichee/tools/tools_win/USB_update_and_produce/, 将 PhoenixSuit.msi 复制到 XP 主机上，按照安装向导提示安装，此软件支持 Windows XP SP3 及以上系统，在 Ubuntu 环境下可以通过 Virtual Box 虚拟机使用之

2.2.4 其他软件

建议安装 Winrar 便于制作开机动画，在 Windows 环境下可以安装 Windows Media Player 以使用 MTP 的功能

2.3 代码下载说明

请参考 SDK 发布文档的下载说明，须向全志申请下载 sdk 的权限和账号。

2.4 代码与 AOSP 差异比较

SDK android 部分的代码是在 AOSP 的基础上开发的，和 AOSP 的代码相比，增加了以下的仓库

```
android/device/softwinner/common #系统公共代码，框架层扩展的API
android/device/softwinner/petrel-common #H6公共配置文件
android/device/softwinner/petrel-p1 #H6 P1方案板配置仓库
android/external/multi_ir #多遥控器支持代码
android/frameworks/opt/net/pppoe #PPPOE支持代码
android/hardware/aw #全志平台HAL层代码仓库
android/hardware/realtek
android/vendor/fvd #全志应用仓库
android/vendor/operator #运营商应用仓库
```

与 android 平级的 lichee 是内核、uboot、硬件配置的目录，其主要内容如下：

```
lichee/brandy      #uboot boot0代码目录
lichee/buildroot   #内核编译工具链目录
lichee/linux-3.10  #内核代码
lichee/tools       #方案硬件配置，打包工具等
```

对于客户自己建立的方案配置，建议放在 android/device/softwinner 目录下，硬件配置放在 lichee/tools/pack/chips/sun50iw6p1/configs，如果需要过 CTS 认证安装 GMS 包，建议将 GMS 包解压在 android/vendor 目录下，对于预编译需要集成的应用，可以放在 package/apps 目录下，需要集成的二进制源码，可以放在 external 目录下

2.5 海外（CTS）方案与普通方案差异对比

方案中 petrel-p1 一个方案可以 launch 出两个配置，分别是 petrel_fvd_p1 和 petrel_cts_p1, 两者的区别在于前者为普通 OTT 方案，后者为 CTS 方案 22. petrel_fvd_p1-eng 24. petrel_cts_p1-eng

CTS 方案较普通方案相比，为了能够按照谷歌的规格标准，添加了 adb secure 支持，同时打开了窗口旋转，移除了摄像头属性，这些对于国内 OTT 用户的习惯不一定适用，但对于海外 CTS 认证又是必须的，所以务必明确项目的终端客户是哪个市场，选择正确的配置文件参考

2.6 代码编译流程

2.6.1 内核编译流程

在 lichee 目录下输入以下命令：

```
H6SDK/lichee$ ./build.sh config

Welcome to mkscript setup progress
All available chips:
  0. sun50iw1p1
  1. sun50iw2p1
  2. sun50iw6p1
  3. sun8iw11p1
  4. sun8iw12p1
  5. sun8iw6p1
  6. sun8iw7p1
  7. sun8iw8p1
  8. sun9iw1p1
Choice: 2
All available platforms:
  0. android
  1. dragonboard
  2. linux
  3. eyeseelinux
Choice: 0
All available business:
```



```
  0. 5.1
  1. 4.4
  2. 7.x
Choice: 2
```

编译成功后输出内容如下：

```
regenerate rootfs cpio
15757 块
17093 块
build_ramfs
Copy boot.img to output directory ...
Copy modules to target ...

sun50iw6p1 compile Kernel successful


INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: build rootfs OK.
----------------------------------------
build sun50iw6p1 android 7.x lichee OK
----------------------------------------
```


内核的代码在 lichee/linux-3.10 目录，执行上述命令编译前会将配置文件从 lichee/linux-3.10/arch/arm64/configs/sun50iw6p1smp_android_7.x_defconfig 拷贝到 lichee/linux-3.10/.config 作为默认配置，下次编译时可以直接在 liche 下运行./build.sh，将继续采用上一次的.config 配置

2.6.2 uboot/boot0 编译流程（可选）

通常情况下不需要重新编译 uboot，但如果对 uboot 有定制修改可以编译，编译的方法如下：


```
cd lichee/brandy/u-boot-2014.07
make distclean && make sun50iw6p1_config && make -j32   #编译uboot

cd lichee/brandy/u-boot-2014.07
make distclean && make sun50iw6p1_config && make spl    #编译boot0
```

如果没有编译 uboot/boot0 的话，默认是采用 lichee/tools/pack/chips/sun50iw6p1/bin 已经预编译好的结果，采用上述命令重编译后，将会自动拷贝替换掉上述文件

2.6.3 Android 代码编译流程

```
cd ~/workspace/AndroidN/android
source ./build/envsetup.sh    ----导入环境变量(执行者一句之后会默认选择openjdk8)
lunch    petrel_fvd_p1-eng    ----根据自己的开发平台选择方案
extract-bsp                   ----拷贝lichee下的内核和模块到android中
make -j8                      ----j开启多核编译，服务器开发一般为服务器cpu数量的一半
pack                          ----打包生成固件
```

2.6.4 jack 编译问题

Android7.0 使用了 Jack 编译工具，首次编译时容易有 Jack 编译失败的问题出现，如果对于单用户模式的开发的话，则 Jack 服务器的端口号不需要修改，但如果对应多用户服务器的开发模式的话，则会出现有用户占用了 jack 服务器的端口，导致无法编译通过的问题出现，需要修改默认的 jack 服务器端口。

编译到一半后，编译系统会自动在/目录下生成/.jack-settings 文件，用 vi 打开此文件，修改端口号

```
SERVER_PORT_SERVICE=9076 改成没有被占用的端口号
SERVER_PORT_ADMIN=9077 改成没有被占用的端口号
```


接着修改 ~/.jack-server/config.properties 文件，如果没有这个文件，就继续编译 Android 代码，系统会自动生成这个文件，等到报错之后再打开修改

```
jack.server.service.port=9076 改成与上面的端口号一致
jack.server.admin.port=9077 改成与上面的端口号一致
```

然后运行以下命令启动 jack 服务器，如果配置没有问题的话，下面命令

```
jack-admin start-server
```

注意 ~/.jack-server/目录下的文件，要确保文件的属性不变，建议使用 vi 而不要使用其他可能改变文件属性的编辑器进行修改，否则也会导致编译错误，正常情况下的文件属性如下：

```
-rw------- 1 yuguoxu yuguoxu    2082  8月 26  2016 client.jks
-rw------- 1 yuguoxu yuguoxu    2794 12月  6 10:18 client.pem
-rw------- 1 yuguoxu yuguoxu     282  8月 26  2016 config.properties
drwxrwxr-x 2 yuguoxu yuguoxu    4096  8月 26  2016 jack
-rw------- 1 yuguoxu yuguoxu 4378061  8月 26  2016 launcher.jar
drwx------ 2 yuguoxu yuguoxu    4096  3月 30 19:53 logs
-rw------- 1 yuguoxu yuguoxu 4758810  8月 26  2016 server-1.jar
-rw------- 1 yuguoxu yuguoxu    2067  8月 26  2016 server.jks
-rw------- 1 yuguoxu yuguoxu    1042 12月  6 10:18 server.pem
```


2.6.5 刷机方法

将生产的 img 固件拷贝出来，用 PhoenixSuite 选择固件后，双口 USB 线连接 PC 和样机，按下 uboot 按键，再重启上电，即可进行刷机

3. 新增方案定制步骤

3.1 添加定制的方案板配置

我们将在 device 目录下建立自己的方案的过程称为配置过程，为了方便配置方案，我们通常在已有的方案上进行修改达到配置方案的目的，在配置之前，我们必须明确我们的方案是否需要通过 CTS 认证，以确定我们的方案配置的基底，下面我们以添加一个名为 petrel-xxx 的方案为例，介绍如何配置新的方案将 android/device/softwinner/petrel-p1 拷贝一份为 android/device/softwinner/petrel-xxx, 然后利用代码搜索工具或者 shell 脚本查找相应的字符串进行重命名使用 shell 命令操作如下：

```
$ cd android/device/softwinner/
$ cp -r petrel-p1 petrel-xxx
$ cd petrel-xxx/
$ rm -rf .git modules kernel              #删除原有的git modules目录和内核
$ mv petrel_cts_p1.mk   petrel-xxx.mk     #重命名makefile文件，此步需要考虑方案是CTS还是OTT
$ find . -type f | xargs sed -i 's/petrel_p1/petrel_xxx/g' #替换掉petrel_p1的字符串
$ find . -type f | xargs sed -i 's/petrel-p1/petrel-xxx/g' #替换掉petrel-p1的字符串
$ git init                          #重新初始化git管理
$ git add * -f
$ git commit -m "init first versioion for petrel-xxx"          #第一次提交
```

3.1.1 添加海外方案配置

如果我们的方案是需要过 CTS 认证，那么我们的配置文件应当基于 petrel_cts_p1.mk 来进行配置，也就是执行

```
mv petrel_cts_p1.mk  petrel-xxx.mk
```


3.1.2 添加国内方案配置

对于国内 OTT 市场的方案，可以采用

```
mv petrel_fvd_p1.mk  petrel-xxx.mk
```

3.1.3 修改厂商名称

经过上的步骤后打开 petrel-xxx.mk 文件后，可以修改方案的名称和厂商的名字

```
PRODUCT_BRAND := Allwinner
PRODUCT_NAME := petrel_cts_p1
PRODUCT_DEVICE := petrel-xxx
PRODUCT_MODEL := Allwinner
PRODUCT_MANUFACTURER := Allwinner
```

注意 PRODUCT_NAME、PRODUCT_DEVICE 必须和实际情况相符合，否则会 launch 不成功，另外 PRODUCT_BRAND、PRODUCT_MODEL、PRODUCT_MANUFACTURER 的取名也必须符合规范，必须为英文字母或数字或短下划线，否者将无法通过 CTS 测试

3.2 添加定制的方案 lichee 配置

拷贝一份通用的配置，如 lichee/tools/pack/chips/sun50iw6p1/configs/petrel-p1 为 lichee/tools/pack/chips/sun50iw1p1/configs/petrel-xxx，然后按照实际的硬件电路进行配置修改，配置的方法见《H6_sys_config.fex 配置说明.pdf》和《H6_sys_partition.fex 分区表说明.pdf》。通常对于盒子产品，定制化配置主要集中在" 存储介质分区"、" 遥控器地址键码" 和"wifi 和蓝牙" 等功能上。

以 petrel-p1 为例，在 lichee/tools/pack/chips/sun50iw6p1/configs/petrel-p1 中定义了各个方案的硬件参数配置，每个方案都由两个文件：sys_config.fex, sys_partition.fex, test_config.fex, env.cfg 和 bootlogo.bmp 来定义。其中 sys_config.fex, sys_partition.fex 为 android 固件使用，test_config.fex 为板卡测试工具 dragonboard 使用 (复用 android 的 sys_config.fex). 对于每个模块中譬如 XXX_used 这个参数模块是表示该模块是否用到，当设置为 0(不可用) 时其他

参数可以不用配置，如模块 [ps2_0_para] 中的 ps2_used 设置为 0 时，ps2_scl 和 ps2_sda 可以不用配置。注意：如果 env.cfg 未在方案目录下进行配置，将默认使用配置 lichee/tools/pack/chips/sun50iw6p1/default/env.cfg

3.2.1 分区配置说明

以 petrel-p1 为例，盒子系统中常用的分区大小和作用如下：

分区名	大小	用途
bootloader	16M	Bootloader 资源
env	16M	系统启动环境变量
boot	16M	Android Boot 分区，存放内核，根文件系统等
system	1536M	Android System 分区，存放系统服务、应用等
verity_block	16M	Android dm-verify 数据校验机制使用
recovery	32M	Android Recovery 分区，用于 Android Recovery 系统
misc	16M	Misc 分区，用于写入 BCB（Bootloader Cmd Block）进入 recovery
private	16M	存放厂商序列号等私有数据（私有分区）
sysrecovery	1536M	固件备份分区，用于一键恢复功能 (相当于固件备份)
cache	768M	Android Cache 分区，用于 Recovery 系统存放 OTA 固件等
UDISK	剩余大小	作为 Android 的 data 分区
Reserve0	16M	预留分区
Reserve1	32M	预留分区
Reserve2	16M	预留分区
alog	16M	内核 oops 时将 kernel 的 logbuf 打印到此分区

3.2.2 添加新的分区

开发中添加的分区分为两种，一种为普通分区，另一种为私有分区。私有分区的数据在重新擦除量产升级后数据不会丢失，可以用于存放序列号等数据，公版默认有一个 private 分区，厂商可使用此分区来存放私有数据，如果实际使用不够，还可以继续添加。添加分区的方法参考《H6 sys_partition.fex 分区表说明》。注意，修改 System 分区的大小后，同时需要修改 BoardConfig.mk 中的 BOARD_SYSTEMIMAGE_PARTITION_SIZE 的值

3.3 添加定制的方案板 Android 配置

3.3.1 修改方案资源

3.3.1.1 修改 bootlogo

替换 lichee/tools/pack/chips/sun50iw6p1/boot-resource/boot-resource/bootlogo.bmp 文件。
图片要求：
1.Bootlogo 格式必须是 32 位的 bmp 格式的图片。
2.Bootlogo 图片的只支持分辨率为 1280*720 的 bmp 图片。
3. 如果希望开机过程中无黑屏，bootlogo 应该和开机动画第一帧一样。

3.3.1.2 修改开机动画

修改开机动画可以替换掉 android/device/softwinner/petrel-p1/media/bootanimation.zip 文件，开机动画的制作方法如下：

1. 准备 part0、part1 的逐帧图片资源（PNG），通常 part0 只播放一次，part1 循环播放至开机

2. 准备 desc.txt，此文件的内容示例如下

```
400 409 16
p 1 0 part0
p 0 0 part1
```

400 409 16 ---这里的 400 代表图片的像素（大小）宽度，409 代表图片的像素（大小）高度，16 代表帧数；
p 1 0 part0 ---这里的 p 代表标志符，1 代表循环次数为 1 次，0 代表阶段间隔时间为 0，part0 代表对应的文件夹名，为第一阶段动画图片目录；
p 0 0 part1---这里的 p 代表标志符，0 代表本阶段无限循环，0 代表阶段间隔时间为 0，part1 代表对应的文件夹名，为第二阶段动画图片目录；准备好资源后，用选中所有的文件，用 winrar 压缩，并选择存储方式压缩成 zip 格式的文档

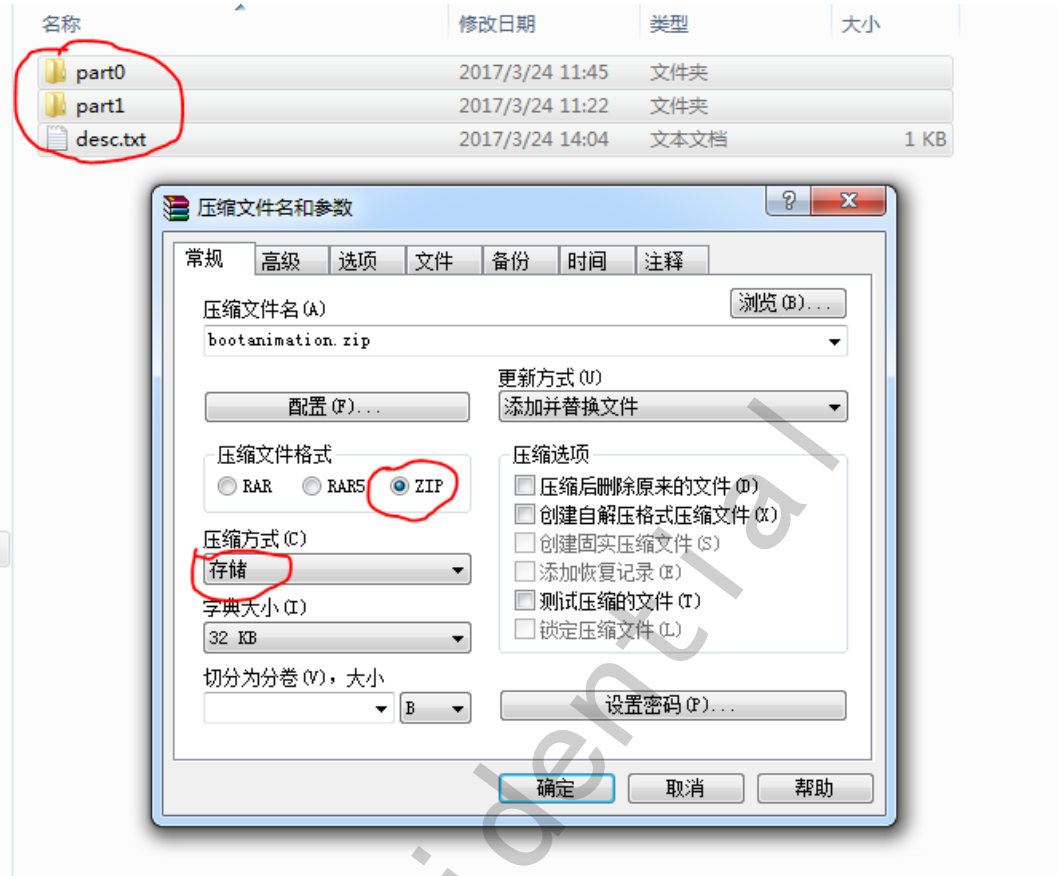

图 2

3.3.2 方案目录内文件说明

3.3.2.1 petrel_fvd_p1.mk

H6 国内 OTT 平台差异化配置文件，文件内部定义了需要定制的信息，如需要预装的 apk 包，需要编译的 apk 源码、产品名字等等。应该把这个文件名改为自己的方案名，如：petrel_fvd_p1.mk，文件中有几个比较重要变量值意义如下：- PRODUCT_PACKAGES

这里定义了需要添加的产品包或库，添加上去后，会编译该源码，打包之后固件里就会有该 apk 或库文件，需要添加生成的 apk 或.so 文件时，应该把它的.mk 文件中定义的 PACKAGENAME 的值加上去。

- PRODUCT_COPY_FILES　　编译时把该环境变量中定的东西拷贝到指定的路径，如在 petrel-p1 方案目录下把 init.rc 拷贝到根目录下：


```
PRODUCT_COPY_FILES += \
device/softwinner/rabbit-p1/init.rc:root/init.rc \
```

PRODUCT_PROPERTY_OVERRIDES(定义 Property 环境参数) 此命令用于向 android 系统中添加系统属性，这些属性在编译时会被收集，最终放到系统的 system/build.prop 文件中，开机时被加载，可以被系统读取到并进行相应的设置。比如下面属性定义了固件的默认时区、国家、语言。

```
PRODUCT_PROPERTY_OVERRIDES += \
      persist.sys.timezone=Asia/Shanghai \
      persist.sys.language=zh \
      persist.sys.country=CN
```

3.3.2.2 petrel_cts_p1.mk

H6 海外 CTS 平台差异化配置文件，语法和和上面的国内 OTT 配置文件一样，注意如果定制化的方案是需要过 CTS 认证的方案的话，一定要基于此方案开展配置，避免不必要的错误

3.3.2.3 AndroidProducts.mk

这里只有一句话：

```
PRODUCT_MAKEFILES := \
   $(LOCAL_DIR)/petrel_fvd_p1.mk \
   $(LOCAL_DIR)/petrel_cts_p1.mk
```

此文件和 vendorsetup.sh 一起定义了系统 launch 的方案名称

3.3.2.4 device.mk

此文件是 CTS 和 OTT 市场的共同的公共配置文件，包括了公共必须的包含的包，链接库，配置等信息，如果定制化过程中有两个平台必须的内容，建议也放到此文件中来

3.3.2.5 BoardConfig.mk

此文件非常重要，除了包括 wifi 的配置信息之外，还包括了文件系统及内核传递的信息，有可能需要改动到的内容如下：

```
INSTALLED_KERNEL_TARGET := kernel
BOARD_KERNEL_CMDLINE := selinux=1 androidboot.selinux=enforcing cma=550M
TARGET_USERIMAGES_USE_EXT4 := true
BOARD_FLASH_BLOCK_SIZE := 4096
BOARD_SYSTEMIMAGE_PARTITION_SIZE := 1610612736
BOARD_USERDATAIMAGE_PARTITION_SIZE := 1610612736
```

修改 androidboot.selinux=permissive 可以将 SELinux 改成 permissive 状态（即只警告而不拦截）修改 BOARD_SYSTEMIMAGE_PARTITION_SIZE 可以修改 system 分区的大小，这个值的单位是字节，必须与 sys_partation.fex 中定义的大小一致

3.3.2.6 init.sun50iw6p1.rc

此文件是系统的初始化启动文件，被 init 进程解析执行，重要的驱动加载、重要目录的创建、权限的修改都是在其中执行的，如果自己开发的驱动 ko 文件需要在系统启动时加载，也可以加入到其中, 注意 system 分区是在 on fs 之后才挂载的，所以如果要加载的驱动的代码，务必添加在 mount_all 过程之后

```
on fs

    mkdir /Reserve0 0777 system system
```



```
mount_all /fstab.sun50iw6p1

swapon_all /fstab.sun50iw6p1

insmod /system/vendor/modules/mali_kbase.ko
insmod /system/vendor/modules/snd-hwdep.ko
insmod /system/vendor/modules/snd-usbmidi-lib.ko
insmod /system/vendor/modules/snd-usb-audio.ko

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

3.3.2.7 ueventd.sun50iw6p1.rc

此文件是定义/dev 文件系统系的设备节点权限和用户组，如果新加入的设备节点需要修改权限，可以在此处加入

```
/dev/disp               0770    system      system
/dev/cedar_dev          0666    media       media
/dev/deinterlace        0666    media       media
/dev/ace_dev            0666    system      system
/dev/ump                0777    system      graphics
/dev/mali0              0666    system      graphics
/dev/video0             0770    media       media
/dev/video1             0770    media       media
/dev/snd/pcmC0D0c       0770    media       media
/dev/snd/pcmC0D0p       0770    media       media
/dev/ttyUSB0            0777    system      system
/dev/ttyUSB1            0777    system      system
/dev/ttyUSB2            0777    system      system
/dev/ttyUSB3            0777    system      system
/dev/ttyACM0            0777    system      system
/dev/ttyACM1            0777    system      system
```



```
/dev/ttyACM2              0777    system      system
```

3.3.2.8 fstab.sun50iw6p1

此文件为系统挂载分区配置，通常内容如下

```
# Android fstab file.
#<src>                          <mnt_point> <type> <mnt_flags and options> <fs_mgr_flags>
# The filesystem that contains the filesystem checker binary (typically /system) cannot
# specify MF_CHECK, and must come before any filesystems that do specify MF_CHECK

/dev/block/by-name/system   /system     ext4    ro  wait
/dev/block/by-name/cache   /cache    ext4   noatime,nosuid,nodev,nomblk_io_submit,barrier=1
   wait,check
/dev/block/by-name/UDISK   /data     ext4   noatime,nosuid,nodev,nomblk_io_submit,barrier=1
   wait,check
/dev/block/by-name/Reserve0 /Reserve0   vfat   rw      wait,check
/dev/block/by-name/misc     /misc       emmc    defaults    defaults
#tf
/devices/soc/sdc0/mmc_host*     auto    vfat    defaults
   wait,check,voldmanaged=extsd:auto,encryptable=userdata
#usbc0->usb0
/devices/soc/5101000.ehci0-controller*  auto    vfat    defaults
   wait,check,voldmanaged=usbhost:auto,encryptable=userdata
/devices/soc/5101000.ohci0-controller*  auto    vfat    defaults
   wait,check,voldmanaged=usbhost:auto,encryptable=userdata
#usbc1->usb1 3.0
/devices/soc/5200000.xhci-controller*   auto    vfat    defaults
   wait,check,voldmanaged=usbhost:auto,encryptable=userdata
#usbc3->usb2(perf)
/devices/soc/5311000.ehci3-controller*  auto    vfat    defaults
   wait,check,voldmanaged=usbhost:auto,encryptable=userdata
```



```
/devices/soc/5311000.ohci3-controller* auto    vfat    defaults
    wait,check,voldmanaged=usbhost:auto,encryptable=userdata
/dev/block/zram0    none    swap    defaults    zramsize=134217728
```

zram 的大小可以在此处进行修改

3.3.2.9 recovery.fstab

此文件在 recovery 及 OTA 升级中被使用到，用于决定 recovery 时挂载的分区，通常情况下不需要进行修改

3.3.2.10 vendorsetup.sh

此文件定义 launch 时的方案

3.3.2.11 package.sh

此文件为打包生成固件的脚本，支持的参数如下：

```
"Usage: pack [-cCHIP] [-pPLATFORM] [-bBOARD] [-d] [-s] [-h]
        -c CHIP (default: $chip)
        -p PLATFORM (default: $platform)
        -b BOARD (default: $board)
        -d pack firmware with debug info output to card0
        -s pack firmware with signature
        -h print this help message
    -m normal or ota boot test"
```

通常使用到的参数为：pack;pack -d;pack -s -f;pack -s -f -d 分别为生成 uart0 打印输出的固件，生成串口重定向到 TF 卡卡槽并打开 JTAG 的固件，生成安全的固件，生成安全并串口重定向到 TF 卡卡槽的固件

3.3.2.12 config.xml

此文件为系统框架层的配置文件，路径为：overlay/frameworks/base/core/res/res/values/config.xml，通常情况下不需要进行修改

3.3.2.13 defaults.xml

此文件为系统 SettingsProvider 的配置文件，路径为：overlay/frameworks/packages/SettingsProvider/res/values/defaults.xml，通常情况下不需要进行修改

4. 遥控器配置

添加遥控器支持需要知道遥控器的地址码以及扫描码，对于新的未适配的遥控器来说，首先要获取其协议为 NEC 制式还是 RC5 制式，遥控器地址码为多少，下面介绍添加新遥控器的方法

4.1 遥控器地址码

遥控器的地址码需要添加到 sys_config.fex 中，否则无法识别,sys_config.fex 默认已经配置了一些按键码

```
;-------------------------------------------------------------------------
;ir --- infra remote configuration
;ir_protocol_used : 0 = NEC / 1 = RC5
;-------------------------------------------------------------------------
[s_cir0]
s_cir0_used         = 1
ir_protocol_used    = 0
ir_power_key_code0  = 0x57
ir_addr_code0       = 0x9f00
ir_power_key_code1  = 0x1a
ir_addr_code1       = 0xfb04
ir_power_key_code2  = 0x14
ir_addr_code2       = 0x7F80
ir_power_key_code3  = 0x15
ir_addr_code3       = 0x7F80
ir_power_key_code4  = 0x0b
ir_addr_code4       = 0xF708
ir_power_key_code5  = 0x03
ir_addr_code5       = 0x00EF
ir_power_key_code6  = 0xdc
ir_addr_code6       = 0x4cb3
```



```
ir_power_key_code7  = 0x0a
ir_addr_code7       = 0x7748
ir_power_key_code8  = 0x45
ir_addr_code8       = 0xbd02
ir_power_key_code9  = 0x4d
ir_addr_code9       = 0xde21
ir_power_key_code10  = 0x18
ir_addr_code10      = 0xfe01
ir_power_key_code11  = 0x18
ir_addr_code11      = 0xff00
ir_power_key_code12  = 0x4d
ir_addr_code12      = 0xff40
ir_power_key_code13  = 0x88
ir_addr_code13      = 0xdd22
ir_power_key_code14  = 0x0d
ir_addr_code14      = 0xbc00
ir_power_key_code15  = 0x0d
ir_addr_code15      = 0xfc00

rc5_ir_power_key_code0 = 0x01
rc5_ir_addr_code0     = 0x04
```


4.2 配置遥控器的按键值

如果已经有一个已经支持遥控器的 Android 设备，可以在 adb 中输入 getevent 命令获取遥控器的信息

```
root@petrel-p1:/ # getevent
getevent
add device 1: /dev/input/event4
  name:     "sunxi-ir-uinput"
add device 2: /dev/input/event3
```

```
  name:    "MCE IR Keyboard/Mouse (sunxi-rc-recv)"
could not get driver version for /dev/input/mouse0, Not a typewriter
add device 3: /dev/input/event2
  name:    "sunxi-ir"
add device 4: /dev/input/event0
  name:    "sunxi-ths"
could not get driver version for /dev/input/mice, Not a typewriter
add device 5: /dev/input/event1
  name:    "axp80-powerkey"
/dev/input/event4: 0001 0014 00000001
/dev/input/event2: 0004 0004 01ff0016
/dev/input/event4: 0000 0000 00000000
/dev/input/event2: 0000 0000 00000000
/dev/input/event4: 0001 0014 00000000
/dev/input/event2: 0004 0004 00ff0016
/dev/input/event4: 0000 0000 00000000
/dev/input/event2: 0000 0000 00000000
/dev/input/event2: 0004 0004 01ff0016
/dev/input/event4: 0001 0014 00000001
```

在 adb 中键入 getevent 后再按下遥控器按键，反应如下，其中 sunxi-ir-uinput 为真实的遥控器设备的值，已 01ff0016 这个数据为例，01 表示按键的状态，1 表示按下，ff00 为地址码，16 为按键值得到遥控器的按键值之后，可以将配置具体的按键为具体的功能，需要修改 kl 文件，常用的 kl 文件放在 device/softwinner/pertel-p1/config 目录下

```
camera.cfg      customer_ir_2992.kl  customer_ir_9f00.kl  customer_ir_dd22.kl
customer_ir_fc00.kl  customer_rc5_ir_04.kl  sunxi-ir.kl        sunxi-keyboard.kl
cameralist.cfg  customer_ir_4cb3.kl  customer_ir_bc00.kl  customer_ir_fb04.kl
customer_ir_ff00.kl  media_profiles.xml     sunxi-ir-uinput.kl  virtual-remote.kl
```

文件的命名务必命名为 customer_ir_xxxx.kl 的方式，其中 xxxx 是地址码，一个 kl 的内容如下


```
key 64   BACK            WAKE_DROPPED
key 4    MENU            WAKE_DROPPED
key 85   DPAD_CENTER      WAKE_DROPPED
key 22   DPAD_DOWN        WAKE_DROPPED
key 70   DPAD_UP          WAKE_DROPPED
key 78   HOME             WAKE
key 71   DPAD_LEFT        WAKE_DROPPED
key 21   DPAD_RIGHT       WAKE_DROPPED
key 20   VOLUME_UP        WAKE
key 16   VOLUME_DOWN      WAKE
key 24   POWER            WAKE
key 29   0                WAKE
key 15   1                WAKE
key 17   2                WAKE
key 18   3                WAKE
key 76   4                WAKE
key 88   5                WAKE
key 27   6                WAKE
key 23   7                WAKE
key 77   8                WAKE
key 10   9                WAKE
key 91   MUTE             WAKE
key 90   PROG_RED         WAKE
key 2    PROG_BLUE        WAKE
key 13   PROG_GREEN       WAKE
key 6    PROG_YELLOW      WAKE
key 80   MEDIA_NEXT       WAKE
key 82   MEDIA_PREVIOUS   WAKE
key 79   SETTINGS         WAKE
key 30   DEL              WAKE
key 28   ENTER            WAKE
key 205  TV               WAKE
key 145  MOVIE            WAKE
```



```
key 131  APPS            WAKE
key 195  FAVOURITE        WAKE
```

注意其中不能有系统不支持的按键，否则系统无法解析此文件将导致配置失败然后在 device.mk 中引入 kl 文件，添加

```
PRODUCT_COPY_FILES += \
    device/softwinner/petrel-p1/configs/virtual-remote.kl:system/usr/keylayout/virtual-remote.kl \
    device/softwinner/petrel-p1/configs/sunxi-keyboard.kl:system/usr/keylayout/sunxi-keyboard.kl \
  device/softwinner/petrel-p1/configs/sunxi-ir.kl:system/usr/keylayout/sunxi-ir.kl \
      device/softwinner/petrel-p1/configs/customer_ir_9f00.kl:system/usr/keylayout/customer_ir_9f00.kl \
      device/softwinner/petrel-p1/configs/customer_ir_dd22.kl:system/usr/keylayout/customer_ir_dd22.kl \
      device/softwinner/petrel-p1/configs/customer_ir_fb04.kl:system/usr/keylayout/customer_ir_fb04.kl \
      device/softwinner/petrel-p1/configs/customer_ir_ff00.kl:system/usr/keylayout/customer_ir_ff00.kl \
      device/softwinner/petrel-p1/configs/customer_ir_xxxx.kl:system/usr/keylayout/customer_ir_xxxx.kl \
      device/softwinner/petrel-p1/configs/customer_ir_4cb3.kl:system/usr/keylayout/customer_ir_4cb3.kl \
      device/softwinner/petrel-p1/configs/customer_ir_bc00.kl:system/usr/keylayout/customer_ir_bc00.kl \
      device/softwinner/petrel-p1/configs/customer_ir_fc00.kl:system/usr/keylayout/customer_ir_fc00.kl \
      device/softwinner/petrel-p1/configs/customer_ir_2992.kl:system/usr/keylayout/customer_ir_2992.kl \
     device/softwinner/petrel-p1/configs/customer_rc5_ir_04.kl:system/usr/keylayout/customer_rc5_ir_04.kl \
   device/softwinner/petrel-p1/configs/sunxi-ir-uinput.kl:system/usr/keylayout/sunxi-ir-
```



```
uinput.kl
```

4.3 多遥控器支持

关于多遥控器的支持，参考《H6 多遥控器使用说明书 V1.0》的内容

4.4 修改" 软鼠标模式" 下使用的键值

软鼠标模式是指使用遥控器的按键去模拟鼠标的动作，需要通过配置属性来关联遥控器按键和鼠标动作，如下所示，在方案配置文件中添加以下属性

```
#define virtual mouse key
    PRODUCT_PROPERTY_OVERRIDES += \
        ro.softmouse.left.code=6 \
        ro.softmouse.right.code=14 \
        ro.softmouse.top.code=67 \
        ro.softmouse.bottom.code=10 \
        ro.softmouse.leftbtn.code=2 \
        ro.softmouse.midbtn.code=-1 \
        ro.softmouse.rightbtn.code=-1
```


4.5 配置开关机及待机

4.6 配置短按和长按遥控器 power 键行为

通过修改方案目录下的 overlay/frameworks/base/core/res/res/value/config.xml 配置短按和长按遥控器的 power 键的行为


```
<!-- longPress behavior
        0 : 什么都不做
        1:  显示GLOBAL_ACTIONS对话框
        2:  弹出关机对话框
        3:  直接关机  -->
      <integer name="config_longPressOnPowerBehavior">2</integer>
      <!-- shortPress behavior
        0 : 休眠
        1:  弹出关机对话框
        2:  直接关机 -->
      <integer name="config_shortPressOnPowerBehavior">0</integer>
```

4.7 替换鼠标图标

替换 3 个图片文件：

```
android\frameworks\base\core\res\res\drawable-mdpi\pointer_arrow.png
android\frameworks\base\core\res\res\drawable-hdpi\pointer_arrow.png
android\frameworks\base\core\res\res\drawable-xhdpi\pointer_arrow.png
```

4.8 游戏手柄配置

游戏手柄实质上也是一个 input 设备，通过配置对应的 kl 文件能够让它正常工作，游戏手柄通常是蓝牙或者 usb2.4G 的设备，通过 dumpsys input 设备可以知道它的名字以及对应的 id，下图为一个游戏手柄的典型的信息

```
6: Gamepad
  Classes: 0x80000141
  Path: /dev/input/event5
  Descriptor: 45a5bcaa66c2d59edad17fa809c30d066eca2431
```



```
Location:
ControllerNumber: 2
UniqueId: AA:B2:21:CA:8F:F8
Identifier: bus=0x0005, vendor=0x18d1, product=0x2c40, version=0x0050
KeyLayoutFile: /system/usr/keylayout/Generic.kl
KeyCharacterMapFile: /system/usr/keychars/Generic.kcm
ConfigurationFile:
HaveKeyboardLayoutOverlay: false
```

添加游戏手柄配置的方法是根据相应的 vid、pid，构造相应的 kl 文件，如 Vendor_18d1_Product_2c40.kl，其内容通常如下：

```
# Copyright (C) 2013 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# ION GO PAD

key 288 BUTTON_A
key 289 BUTTON_B
key 290 BUTTON_X
key 291 BUTTON_Y
key 294 BUTTON_L1
```



```
key 295 BUTTON_R1
key 292 BUTTON_L2
key 293 BUTTON_R2

axis 0x00 HAT_X
axis 0x01 HAT_Y
```

然后放到编译到/system/usr/keylayout 目录下即可

5. 预编译 APK

5.1 源码预装

官方应用程序源码在 android/package/apps 目录下，如果需要在生成固件时包含某个 apk(如音乐播放器, 超清播放器)，需要把其包名添加到需要编译的列表中。比如要添加录音程序，则在 android/device/softwinner/petrel-xxx/petrel_xxx.mk 文件内，给变量 PRODUCT_PACKAGES 添加一个新的值 ``SpeechRecoder"（注意，可能需要添加 `` " 做分隔）。SpeechRecoder 就是该应用的包名 (PACKAGE_NAME)，包名可以在每个应用程序源码的 Android.mk 文件中找到 ``LOCAL_PACKAGE_NAME" 的值。

5.2 system 预装

对于第三方没有源码的 APK 预装，如果希望预装到 system 分区（用户不可卸载），可以按以下步骤操作，以芒果 TV 的预装为例：

1. 查看 APK 有没有 JNI 库：

使用 winrar 等软件打开 apk，检查 apk 是否包含 lib 库，如果包含 lib 库，解压其中的 lib 库，注意必须解压 armeabi 文件夹下的 mips/x86 下的无用。

图 3

2.APK/JNI 库集成到方案目录

以芒果 TV 的 APK 为例，将芒果 TV APK 拷贝 android/vendor/fvd/prebuild/apk/HunanOTT-STD 目录并将解压出来的 APK JNI 库拷贝到 android/vendor/fvd/prebuild/prebuild/apklib 目录，修改 android/vendor/fvd/prebuild/prebuild/apk/Android.mk，添加如下

```
################################################################################
################################################################################
# HunanOTT-STD.apk
my_archs := arm
my_src_arch := $(call get-prebuilt-src-arch, $(my_archs))
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := HunanOTT-STD
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_TAGS := optional
```



```
LOCAL_BUILT_MODULE_STEM := package.apk
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := PRESIGNED
LOCAL_MODULE_TARGET_ARCH := arm
#LOCAL_OVERRIDES_PACKAGES :=
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
#LOCAL_REQUIRED_MODULES :=
LOCAL_PREBUILT_JNI_LIBS := \
    lib/arm/lib_All_imgoTV_bitmaps.so \
    lib/arm/lib_All_imgoTV_nn_tv_air_control.so \
    lib/arm/lib_All_imgoTV_nn_tv_client.so \
    lib/arm/libbspatch.so
include $(BUILD_PREBUILT)
```

然后再在主 makefile（device.mk）中添加对 HunanOTT-STD 的引用

```
PRODUCT_PACKAGES += \
    HunanOTT-STD
```

5.3 Preinstall 预装

preinstall 方式预装 apk 常用于 JNI 库较多、用户可卸载的场景，可以按照以下步骤操作：APK 集成到方案目录
1. 将 Abc.apk 拷贝至 vendor/fvd/prebuild/apk/ 2. 将 APK 内的 jni 库解压到应用的目录下，如 lib/arm/的路径 3. 编写 Android.mk

```
include $(CLEAR_VARS)
LOCAL_MODULE := Abc
LOCAL_MODULE_TAGS := optional
LOCAL_CERTIFICATE := PRESIGNED
LOCAL_MODULE_PATH := $(TARGET_OUT)/preinstall
LOCAL_MODULE_CLASS := APPS
```



```
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
include $(BUILD_PREBUILT)
```


6. GPIO 配置

6.1 定义需要控制的 GPIO

通常这一块不需要太多的修改，但如果需要进行修改的话，可以参考 lichee/tools/pack/chips/sun50iw6p1/configs/petrel-p1/sys_config.fex 文件中，类似如下的配置信息：

```
;----------------------------------------------------------------------------
;userspace gpio interface for android
;----------------------------------------------------------------------------
[gpio_para]
compatible          = "allwinner,sunxi-init-gpio"
gpio_used           = 1
gpio_num            = 3
gpio_pin_1          =  port:PL07<1><default><default><1>
gpio_pin_2          =  port:PL03<1><default><default><0>
gpio_pin_3          =  port:PL04<1><default><default><0>
normal_led          = "gpio_pin_1"
standby_led         = "gpio_pin_2"
network_led         = "gpio_pin_3"
easy_light_used = 1
normal_led_light = 1
standby_led_light = 1
network_led_light = 1
```


在这个范例中，变量 gpio_used 置为"1" 表示此配置将起作用，其他的就是各个 GPIO 的配置信息。这些 GPIO 的编码必须从"1" 开始依次递增。
通常盒子会有两个 gpio 控制两个颜色的灯，为了向 Android 框架提供统一路径控制 Led 灯，所以提供一个指定控制命名 Led 的 GPIO 的配置，包括 normal 和 standby，上面的内容就是将 gpio_pin_1 配置为 normal_led，gpio_pin_2 配置为 standby_led，gpio_pin_3 配置为 network_led。

6.2 配置 boot 阶段初始化的 gpio 功能

系统上电的时候，能快速的初始化用户自定义的 GPIO 口，这里包括：上电亮灯等。

配置在 lichee/tools/pack/chips/sun50iw6p1/configs/petrel-xxx/sys_config.fex 文件中，根据自己方案中要上电初始化 GPIO 来添加类似如下的配置信息：范例：

```
[boot_init_gpio]
boot_init_gpio_used     = 1
gpio0 =  port:PL07<1><default><default><1>
gpio1 =  port:PA03<1><default><default><0>
```

以上配置表示：在 boot 阶段，设置 PL07 输出高电平，PA03 输出低电平。

6.3 控制 GPIO 的接口

添加了 GPIO 的配置后，会在 sys 文件系统下产生节点

```
root@petrel-p1:/sys/class/gpio_sw # ls
ls
PL3
PL4
PL7
network_led
normal_led
standby_led
root@petrel-p1:/sys/class/gpio_sw # pwd
pwd
/sys/class/gpio_sw
root@petrel-p1:/sys/class/gpio_sw/PL3 # echo 1 > data
echo 1 > data
root@petrel-p1:/sys/class/gpio_sw/PL3 # echo 0 > data
```

对于目录下的 data 节点写入 0，将导致输出低电平，写入 1，将导致输出高电平，为

了方便代码中进行操作，有提供 java 以及 C++ 的接口

6.4 java 层的接口

java 控制 GPIO 的接口定义在文件 Gpio.java 中，其路径为：android/device/softwinner/common/addons/framework/gpio/java/Gpio.java 在 java 代码中 import com.softwinner.Gpio; 的 setNormalLedOn(bool) 和 setStandbyLedOn(bool) 接口方便的操作 Led 的亮灭。提供的接口如下：

```
public static int setNormalLedOn(boolean on);
public static int setStandbyLedOn(boolean on);
public static int setNetworkLedOn(boolean on) ;
public static int writeGpio(char group, int num, int value);
public static int readGpio(char group, int num) ;
public static int setPull(char group, int num, int value);
public static int getPull(char group, int num);
public static int setDrvLevel(char group, int num, int value);
public static int getDrvLevel(char group, int num);
public static int setMulSel(char group, int num, int value);
public static int getMulSel(char group, int num);
private static String composePinPath(char group, int num);
```

6.5 c++ 层的接口

C++ 层的操作函数是对内核接口的简单封装，具体的接口如下

```
cfg：设置/读取gpio的功能
0x00：input
0x01：output
pull：设置/读取gpio电阻上拉或者下拉
0x00：关闭上拉/下拉
```


0x01：上拉
0x02：下拉
0x03：保留
drv：设置/读取gpio的驱动等级
0x00：level 0
0x01：level 1
0x02：level 2
0x03：level 3
data：设置/读取gpio的电平状态
0x00：低电平
0x01：高电平

在 C 语言中可以用 read 和 write 函数直接操作这 4 个文件。具体的范例可参考文件 android/device/softwinner/common/addons/framework/gpio/libgpio/GpioService.cpp 中的代码。

7. USB 外设配置

通常情况下 USB 外设不需要太多设置，pertel-p1 有 USB0、USB1 两个接口，其中 USB0 为 OTG 接口，可以配置其状态

7.1 adb 调试配置

```
;-------------------------------
;[usbc0]: usbc0 configuration.
;usb_used: usb controller enable. 0-disable, 1-enable.
;usb_port_type: usb mode. 0-device, 1-host, 2-otg.
;usb_detect_type: usb hotplug detect mode. 0-none, 1-vbus/id detect, 2-id/dpdm detect.
;usb_detect_mode: usb otg switch has two config. 0-thread scan, 1-id gpio interrupt.
;usb_id_gpio: usb id detect IO.
;usb_det_vbus_gpio: USB DET_VBUS has two config. (1)gpio pin; (2)"axp_ctrl", use axp intf.
;usb_drv_vbus_gpio: USB DRY_VBUS has two config. (1)gpio pin; (2)"axp_ctrl", use axp intf.
;-------------------------------
;-------------------------------
;---      USB0 CONFIG
;-------------------------------
[usbc0]
usbc0_used         = 1
usb_port_type      = 1
usb_detect_type    = 1
usb_detect_mode    = 0
usb_id_gpio        =
usb_det_vbus_gpio  =
usb_drv_vbus_gpio  =
usb_host_init_state = 1
usb_regulator_io   = "nocare"
usb_wakeup_suspend  = 0
```



```
;---      USB Device
usb_luns          = 3
usb_serial_unique   = 0
usb_serial_number   = "20080411"
rndis_wceis       = 1
```

将 usb_port_type 改成 0 可以让 USB0 上电后默认为 device 状态，究竟是工作于 adb 还是 mtp 由 android 的配置决定，usb_serial_unique 改为 1 则启用 USB 唯一序列号，默认的序列号来自于 CPU ID

对于 Android 端 usb0 功能的配置，则是通过修改 pertel_xxx.mk 文件实现的，修改下面一行。

```
persist.sys.usb.config=mtp,adb \
```

需要注意的是这个值必须与 init.sun50iw6p1.usb.rc 定义的一致

7.2 支持外置 USB 蓝牙 dongle

具体配置方法和已验证的支持列表请参见《H6 USB 蓝牙配置使用说明书》和《Allwinner H6 WIFI_BLUETOOTH Support list》。

8. 开机 logo 与开机动画设置

传统的开机动画是固定的，但某些情况下，客户可能希望将开机动画作为一种广告来显示，目前我们有提供这样的接口在使用此功能前需要打开 sys_config.fex 中的设置，将 advert_enable 设置为 1

```
;------------------------------------------------------------------------------
; burn_key:          0-do not use dragonKey   1-use dragonKey
; dragonboard_test:  0-build dragonboard for flash boot   1-build dragonboard for TF boot
; power_mode:        0-use axp    1-use dummy axp
; advert_enable      0-close advert logo    1-open advert logo    (只有多核启动下有效)
;------------------------------------------------------------------------------
[target]
boot_clock      = 1440
storage_type    = -1
burn_key        = 0
dragonboard_test = 0
power_mode      = 0
advert_enable   = 0
```

然后可以调用 setBootAnimation、setBootLogo

```
/*
 描述：设置开机动画，传入开机动画文件路径，系统会拷贝一份作为开机动画
 输入：path代表开机动画压缩文件的路径 输出：返回0代表成功，1代表失败
*/
public static int setBootAnimation(String path)

/*
 描述：设置开机logo，传入开机logo文件路径，系统会拷贝一份作为开机logo
 输入：path代表开机logo文件的路径 输出：返回0代表成功，1代表失败
*/
public static int setBootLogo(String path)
```


开机 logo 的要求和开机动画的要求请配置开机动画的章节，如果传入开机 logo 不是按照规定要求的图片，可能会显示默认开机 logo 或者显示花屏，如果传入的开机动画不符合要求也有可能显示默认开机动画或者显示异常。

8.1 开机视频配置

将视频命名为 boot.mp4，放到/system/media/下或者/data/local/下。系统启动优先从/data/local/检测视频文件，如果没有则从/system/media/下获取，如果两个路径件都没有，则默认使用启动动画。

8.2 开机音乐配置

音乐内置于 bootanimation.zip 中。目录结构为

```
├── bootanimation
│   ├── audio_conf.txt
│   ├── desc.txt
│   ├── part0
│   │   ├── Animation_00000.png
│   │   ├── Animation_00001.png
│   │   ├── Animation_00002.png
│   │   ├── Animation_00003.png
│   │   ├── Animation_00004.png
│   │   ├── Animation_00005.png
│   │   ├── Animation_xxxxx.png
│   │   ├── audio.wav
│   └── part1
│       └── Animation_00045.png
└── bootanimation.zip
```

其中 audio.wav 为 riff/wave 格式音频，即常见的 wav 音频，格式必须为 16bit little endian，44.1K 或 48K。audio_conf.txt 为音频配置文件，内容如下


```
# edit output_disable/output_enable only
card=0
device=0
period_size=1024
period_count=4
mixer "Speaker Function"="spk"
mixer "AIF1IN0L Mux"="AIF1_DA0L"
mixer "AIF1IN0R Mux"="AIF1_DA0R"
mixer "DACL Mixer AIF1DA0L Switch"=1
mixer "DACR Mixer AIF1DA0R Switch"=1
mixer "Left Output Mixer DACL Switch"=1
mixer "Right Output Mixer DACR Switch"=1
#mixer "AIF1 AD0R Mixer ADCR Switch"=1
#mixer "AIF1 AD0L Mixer ADCL Switch"=1
mixer "SPK_L Mux"="MIXEL Switch"
mixer "SPK_R Mux"="MIXER Switch"
mixer "External Speaker Switch"=1
#0: cvbs/headset
#1: hdmi
output_disable=0
output_disable=1
```

只需在最后面用 output_disable 将对应输出禁掉就可以，0 为 cvbs 或耳机，1 为 hdmi 输出。如需要 hdmi 输出，cvbs 不输出，请配置 output_disable=0；如需要 cvbs/hdmi 输出，请去掉后面的 output_disable=0 与 output_disable=1；最后用 zip 命令打包 bootanimation.zip，替代方案目录下 media/bootanimation.zip 即可。

```
zip -r -X -Z store ../bootanimation part*/*.png part*/*.wav desc.txt audio_conf.txt
```


9. 一键恢复功能

9.1 配置说明

注意：支持"Usb-Recovery 功能" 及" 一键恢复" 功能。在制定方案时只能二选一。原理：Usb-recovery 走的是 OTA 流程，而一键恢复功能走的是类似量产的流程。
一键恢复：即在系统遭受破坏时，按住机器的"recovery 键"，上电进入系统恢复功能。此次恢复的系统为出厂时的系统，不包括后续用户自己安装的任何 apk。而且采用一键恢复功能，本地存储设备需要有一块专门分区存储，这样会减少用户可用空间。开启一键恢复功能需要做几个修改点：

1. 在方案的 sys_config.fex 文件里，修改配置，如：

```
;----------------------------------------------------------------------------
;  used: 模块使能端    1：开启模块  0：关闭模块
;  mode: 模式选择      1：一键进入OTA升级    2：一键恢复（通过sysrecovery分区来恢复）  其他值：无效
;  recovery_key：  按键配置  （例如：recovery_key= port:PH16<0><default>）
;----------------------------------------------------------------------------
[recovery_para]
used = 1
mode = 2
recovery_key = port:PL02<0><default><default><default>

[boot_init_gpio]
boot_init_gpio_used     = 1
gpio0 =  port:PL07<1><default><default><1>
gpio1 =  port:PA03<1><default><default><0>
```

将 mode 改成 2，即按下 recovery 键后进入一键恢复模式
2. 添加 sysrecovery 分区（这个分区默认有的）在 lichee/tools/pack/chips/sun8iw7p1/configs/rabbit-xx/sys_partition.fex 文件里


```
;----------------------------->nandk, system image backup—添加的分区
        [partition]
            name        = sysrecovery
            size        = 1343488
            downloadfile = "sysrecovery.fex"
            verify      = 0
```

9.2 注意事项

如果硬件上没有用于``一键恢复" 的 GPIO，而在配置文件中配置了 system 下的 recovery_key 项，有可能会导致系统不断进入一键恢复。如果没有实际按键，把 sys_config.fex 的使能配置为 0

```
[recovery_para]
        used = 0
        mode = 2
        recovery_key = port:PH16<0><default><default><default>
```

9.3 一键恢复失败常见原因

失败的常见原因：（不限于以下）

1. 硬件参数配置文件中的``recovery_key" 参数是否配置正确？2. 硬件问题：硬件上，按下按键时，GPIO 是否发生了对应的电平变化？3. 生成的 img 文件过大，超出了 sysrecovery 分区的大小，导致烧录时就没将备份系统烧录进去。

10. 修改屏保界面

允许系统在待机时间进入屏保界面，在屏保界面下可显示时钟或厂商自己的广告页面，能够更好地提高用户体验，修改使能屏保界面的方法如下：

10.1 设置默认屏保应用

android/ device/ softwinner/ rabbit- fvd- p1/ overlay/ frameworks/ base/ core/ res/ res/ values/ config.xml

```
<!-- Is the dreams feature supported? -->
<bool name="config_dreamsSupported">true</bool>
<!-- If supported, are dreams enabled? (by default) -->
<bool name="config_dreamsEnabledByDefault">true</bool>
<!-- If supported and enabled, are dreams activated when docked? (by default) -->
<bool name="config_dreamsActivatedOnDockByDefault">true</bool>
<!-- If supported and enabled, are dreams activated when asleep and charging? (by default) -->
<bool name="config_dreamsActivatedOnSleepByDefault">true</bool>
<!-- ComponentName of the default dream (Settings.Secure.SCREENSAVER_COMPONENT)
-->
<string name="config_dreamsDefaultComponent">com.android.dreams.web/
com.android.dreams.web.Screensaver</string>
```

其中 config_dreamsDefaultComponent 为默认的屏保应用的包名，这个应用位于 android/ vendor/fvd/packages/WebScreensaver 目录下 android/device/softwinner/petrel-p1/overlay/ frameworks/base/packages/SettingsProvider/res/values/defaults.xml

```
<!-- If this is true, the screen will come on when you unplug usb/power/whatever. -->
    <bool name="config_unplugTurnsOnScreen">true</bool>
```


10.2 修改广告界面

WebScreensaver 应用能检测当前是否有网络连接，无网络连接的情况下，将使用应用 asset 文件夹下自带的 HTML5 屏保界面，显示为一数字时钟。有网络连接的情况下，可获取网页显示出来：默认的 URL 在 android/vendor/fvd/packages/WebScreensaver/src/com/android/dreams/web/Screensaver.java 中：

```
final SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
                String  url;
                if(isNetConntected())
                        url = prefs.getString("url", "http://www.allwinnertech.com/#zh");
                else
            url = prefs.getString("url", "file:///android_asset/index.html");
        final boolean interactive = prefs.getBoolean("interactive", false);
```


11. OTA 升级说明

参考《H6 N OTA 使用说明文档》

12. SELinux 配置

需要注意的是，在 Android7.0 上，SELinux 不能关闭，否则将导致无法启动，只能设置为 Enforcing 和 Permissive 两种状态，Permissive 状态下 SELinux 将只打印违规信息而不拦截

12.1 开关 SELinux

修改 android/device/softwinner/petrel-p1/BoardConfig.mk，将 androidboot.selinux 设置成 permissive，编译后重新刷机

```
BOARD_KERNEL_CMDLINE := selinux=1 androidboot.selinux=permissive cma=550M
```

输入 getenforce 命令查看当前 SELinux 状态

```
petrel-p1:/ # getenforce
getenforce
Permissive
petrel-p1:/ #
```


12.2 添加自定义的规则

对于部分内置的应用或者二进制程序，如果违反了 SELinux 规则，需要进行改正，不得已的情况下，可以添加 SELinux 规则进行放行添加 SELinux 规则的方法可以参考 https://source.android.com/security/selinux/validate 先将 SELinux 状态设置成 Permissive 状态

```
petrel-p1:/ # setenforce 0
```

再使用以下命令抓取 avc 报错信息


```
petrel-p1:/ # dmesg | grep "avc" > /sdcard/avc.txt
```

上述命令将 avc 报错信息打印到一个文件中去，将这个文件获取出来，在主机上使用下列命令

```
cat avc.txt | audit2allow
```

上述命令将把 avc 信息转化为 te 文件，将得到如下的输出结果，将输出结果填写到 android/device/softwinner/petrel-common/sepolicy 目录下

```
#============= att_policy ==============
allow att_policy system_data_file:dir read;

#============= init ==============
allow init sunxi_soc_info_device:chr_file { ioctl write };

#============= mediacodec ==============
allow mediacodec mediaserver:dir search;
allow mediacodec mediaserver:file { open read };

#============= mediadrmserver ==============
allow mediadrmserver proc_net:file { getattr open read };
```

然后重新编译生成固件

13. 安全方案配置

请参考《H6 安全方案配置文档》

14. 显示配置

请参考《H6 显示模块配置文档》

15. 系统预留内存配置

请参考《H6 CMA 内存配置文档》

16. 系统及应用检测

16.1 屏蔽特定的按键

可以修改 TvWindowManager.java 的代码 android/frameworks/base/services/core/java/com/android/server/policy/TvWindowManager.java 修改

```
@Override
public int interceptKeyBeforeQueueing(KeyEvent event, int policyFlags) {
    int keyCode = event.getKeyCode();
    else{//key up
            switch(keyCode){
                case KeyEvent.KEYCODE_HOME:
                    String focusedWindowApp = mFocusedWindow.getOwningPackage();
                    Log.d(TAG,"the focus window is "+focusedWindowApp);
                    if(focusedWindowApp.equals(FACTORY_TEST_APP)){
                        return 0;
                    }
                    break;
            }
        }
        return super.interceptKeyBeforeQueueing(event,policyFlags);
    }
```


16.2 检测特定的应用是否在前台

可以获取 mFocusedWindow.getOwningPackage(); 变量的值，这个也是修改 android/frameworks/base/services/core/java/com/android/server/policy/TvWindowManager.java 的代码

16.3 禁止特定的应用联网

可以拿掉此应用的联网权限，修改 android/frameworks/base/services/core/java/com/android/server/pm/PackageManagerService.java

```
private void grantPermissionsLPw(PackageParser.Package pkg, boolean replace)
+          if(pkg.packageName.indexOf("com.baidu.input")>=0){
+             Log.d(TAG,"Oh, it's baidu input not grant it's inet");
+             if(name.indexOf("android.permission.INTERNET")>=0){
+                  Log.d(TAG,"hey we not enable baidu to inet");
+                 continue;
+             }
+
+          }
```

16.4 禁止特定的应用升级

可以拒绝应用的升级，修改 android/frameworks/base/services/core/java/com/android/server/pm/PackageManagerService.java 在 installPackageLI 中添加

```
if(replace && pkgName.equals("com.google.android.youtube.tv")){
   Slog.w(TAG,"don't let youtube tv update on android4.4.2");
   res.returnCode = PackageManager.INSTALL_FAILED_INSUFFICIENT_STORAGE;
   return;
}
```

16.5 自启动触发特定的功能

可以在 preinstall.sh 脚本中添加，此脚本的位置位于 android/device/softwinner/common/addons/binary/preinstall.sh 此脚本每次启动都会运行，可以在代码的最后添加代码


```
#!/sbin/busybox sh

BUSYBOX="/sbin/busybox"

if [ ! -e /data/system.notfirstrun ] ; then
    echo "do preinstall job"

    /system/bin/sh /system/bin/pm preinstall /system/preinstall
    /system/bin/sh /system/bin/pm preinstall /sdcard/preinstall

    $BUSYBOX touch /data/system.notfirstrun

    echo "preinstall ok"
else
    echo "do nothing"
fi
qw --daemon
```


17. CTS 测试及认证

请参考《H6 CTS 认证配置文档》

18. 常用调试命令及方法

18.1 提高内核打印等级

修改 lichee/tools/pack/chips/sun50iw1p1/configs/default/env.cfg 中的 loglevel 等级即可，比如：loglevel=4 改为 8，但注意对外发布的固件一定要把 loglevel 改回 4，否则会影响系统启动速度或者系统性能！

18.2 将 logcat 和 dmesg 信息保存到文件系统

为了调试方便，可以在开发调试阶段将系统的 logcat 和内核 dmesg 自动打印到 data 分区文件系统上保存，这种方法可以方便调试偶发问题，系统默认的日志在/data/anr 目录

18.3 使用 fastboot 烧写

使用 fastboot 的功能来实现局部系统的更新。比如，修改内核 buildin 文件或者 init.rc 文件后，使用 make bootimage 命令可以生成 boot.img，使用 fastboot 可以将 boot.img 烧写到机器中 boot 分区，而不用烧写整个固件，从而大大缩短开发时间，命令如下：

```
# adb reboot-bootloader    #进入fastboot模式
# fastboot flash boot boot.img #只烧写boot分区
# fastboot reboot          #重启
```

使用 fastboot 烧写其他分区请参考 android 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
```

18.4 使用网络 adb 调试

如果需要使用网络 adb 调试，可以在 android/device/softwinner/common/init.debug.rc 添加下面的语句：

```
on boot
setprop service.adb.tcp.port 5555
stop adbd
start adbd
```

18.5 调试 apk

修改应用程序 Gallery2，编译修改推送到小机

```
    $ . build/envsetup.sh
    $ lunch #选择方案
    $ cd packages/apps/Gallery2
    $ mm
# fastboot flash data userdata.img   #把userdata.img烧写到data分区
```

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/rabbit-xxx/system/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 remount
> adb push Gallery2Tests.apk /system/app/
```


19. FAQ

20. Declaration

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

