H616 Android Q

SDK 快速移植指南

1.0

2019.12.5

文档履历

版本号	日期	制/修订人	内容描述
1.0	2019.12.5		Android Q

目录

1. 概述

1.1 编写目的

本文档介绍 H616 Android Q 系统常见的定制开发问题，以帮助客户快速熟悉开发环境，实现快速移植方案。

1.2 SDK 概述

参考《H616_Android_Q_SDK Quick Start Guide》搭建开发环境

1.3 名词解释

H616 平台快速移植文档，本文基于 H616 cupid-p2 方案。

1. vendor-name
   softwinner
2. device-name
   cupid-p2
3. product-name
   cupid_p2
4. chip-name
   h616
5. board-name
   p2

2. 方案定制

方案目录 device/vendor-name/device-name/

2.1 overlay 说明

Android overlay 机制允许在不修改 apk 或者 framework 源代码的情况下，实现资源的定制。以下几类能够通过 overlay 机制定义：

1. Configurations (string, bool, bool-array)
2. Localization (string, string-array)
3. UI Appearance (color, drawable, layout, style, theme, animation)
4. Raw resources (audio, video, xml)
   更详细的资源文件可浏览 android 网站：http:// developer.android.com/ guide/ topics/ resources/ available-resources.html

2.1.1 为产品添加 Overlay 目录

有两种不同的 overlay 目录定义：

1. PRODUCT_PACKAGE_OVERLAYS
   用于指定产品
2. DEVICE_PACKAGE_OVERLAYS
   用于同一设备模型的一系列产品

如果包含同一资源，那么 PRODUCT_PACKAGE_OVERLAYS 将覆盖 DEVICE_PACKAGE_OVERLAYS 。如果要定义多个 overlays 目录，需要用空格隔开，同一资源的定义，将使用先定义的目录中的资源。

在方案目录下创建 overlay 和 product-name/overlay 目录，分别用于 device 通用及 product 使用的 overlay 文件夹。

2.1.2 改变 mk 文件来添加 overlays 的编译项

在文件device/vendor-name/device-name/product-name.mk中添加：

```
PRODUCT_PACKAGE_OVERLAYS := \
    device/vendor-name/device-name/product-name/overlay \
    $(PRODUCT_PACKAGE_OVERLAYS)
DEVICE_PACKAGE_OVERLAYS := \
    device/vendor-name/device-name/overlay \
    $(DEVICE_PACKAGE_OVERLAYS)
```

注：必须加上 $(PRODUCT_PACKAGE_OVERLAYS) 变量否则将找不到默认资源。

2.1.3 在 overlay 目录下创建资源文件

在overlay目录下创建和要替换资源所在文件相同的路径的文件，此路径是相对于android platform目录。如替换framework-res路径为：platform/framework/base/core/res/res/values/config.xml中的某一项，则在overlay中创建对应的路径：overlay/framework/base/core/res/res/values/config.xml并添加要修改的一向配置，如：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="config_showNavigationBar">true</bool>
</resources>
```

2.2 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。

注：apk 名字不能含有中文、空格等特殊字符。

由于涉及版权问题，建议不安装GAPP应用。若通过GMS认证需安装Google提供的正版GAPP应用。

2.2.1 预装到 system/app 目录

1. 在目录 vendor/aw/public/prebuild/apk/中创建一个目录存放对应 APK。
2. 将 apk 放入该目录中。
3. 在该目录中创建 Android.mk 文件，并编辑：

```
# Example
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := APK_MODULE_NAME（模块的唯一名字）
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_TAGS := optional
LOCAL_BUILT_MODULE_STEM := package.apk
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := PRESIGNED（签名方式）
#LOCAL_OVERRIDES_PACKAGES := OVERRIDES_MODULE（要替代的模块）
LOCAL_SRC_FILES := name.apk（apk的文件名，一般与MODULE同名）
include $(BUILD_PREBUILT)
```

4. 在方案 mk 文件（device/vendor-name/device-name/product-name.mk）中 PRODUCT_PACKAGES 项中加入：

```
PRODUCT_PACKAGES += APK_MODULE_NAME（apk模块名字，预装多个apk用空格隔开）
```

2.2.2 预装到 system/preinstall 目录

1. 同预装到 system/app 目录，完成所有步骤。
2. 修改 apk 目录下的 Android.mk，加入一行：

```
LOCAL_MODULE_PATH := $(TARGET_OUT)/preinstall
```

2.2.3 配置分区

data 分区大小可以由 BoardConfig.mk 文件的 BOARD_USERDATAIMAGE_PARTITION_SIZE 指定，单位是字节。system、vendor 则分区分别由 BOARD_SYSTEMIMAGE_PARTITION_SIZE，BOARD_VENDORIMAGE_PARTITION_SIZE 指定。

注：一般将最后一个分区作为 data 分区，该分区大小是 Nand 或者 eMMC 总容量减去其他分区大小。如果需要烧写 data 分区镜像，分区大小需要预留一定预度，防止超出 Nand 或者 eMMC 容量。

2.3 修改启动 LOGO

启动 LOGO 为初始引导阶段的 LOGO。

将启动 logo 放入位置：longan/device/config/chips/chip-name/configs/board-name/bootlogo.bmp

2.4 修改开机动画

将动画放入：device/vendor-name/device-name/media/bootanimation.zip

2.4.1 文件结构

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

```
|- desc.txt
|- audio_conf.txt
|- part0
```

```
        |- part000.png
        |- part001.png
        |- part0…….png
        |- audio.wav(可选)
        |- trim.txt(可选)
    |- part1
        |- part100.png
        |- part101.png
        |- part1…….png
        |- audio.wav(可选)
        |- trim.txt(可选)
    |- part...N
```

2.4.2 desc.txt 配置文件

```
第一行：
WIDTH HEIGHT FPS
后面每行，表示部分part动画：
TYPE: 类型（p：播放直到开机完成，c：播放完整动画）
COUNT: 循环次数，0表示无限循环直到开机结束
PAUSE: part结束后暂停帧数
PATH: 文件加路径（如：part0）
RGBHEX:（可选）背景颜色：#RRGGBB
CLOCK:（可选）画当前时间的y坐标(for watches)
```

例如：

```
800 480 15
p 1 0 part0
p 0 0 part1
```

说明：第一行：800 为宽度，480 为高度，15 为帧数。第二行开始 p 为标志符，接下来第二列为循环次数（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。播放动画时会按照图片文件名顺序自动播放。

2.4.3 开机音乐

如需开机音乐，将开机音乐放入 part0 目录中，命名为 audio.wav。在根目录中加入 audio_conf.txt，复制原有动画配置即可。

2.4.4 优化 png 图片

由于图片占用内存较大，需要做一些优化来减少图片资源占用及加快读取时间，可参考源码中 frameworks/base/cmds/bootanimation/FORMAT.md 文件进行优化，如下命令可以无损压缩图片：

```
for fn in *.png ; do zopflipng -m ${fn} ${fn}.new && mv -f ${fn}.new ${fn}; done
```

2.4.5 打包 bootanimation.zip

windows 使用 winrar 打包，选择 ZIP 格式，压缩标准要选``储存"；linux 系统下使用命令：$ zip -0qry -i *.txt *.png *.wav @ bootanimation.zip .txt part。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。

2.5 定制 recovery 功能

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。

2.5.1 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，小机开机时按住对应按键，此时会串口屏幕上打印对应按键的键值。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。

2.5.2 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 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_mix = 0x3
key_man = 0x5
```

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：

1. 按照前面介绍的方法，读出所有按键的键值；
2. 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>c1；
3. 计算出 (a1 + b1)/2，(a1 + c1)/2，分别填写到 key_max 和 key_min 处；
4. 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。

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。

3. 模块配置

3.1 红外遥控器配置

3.1.1 内核配置

要支持红外遥控器 (多遥控器适配)，需要打开下面的配置：

```
1.Device Drivers -> <*>Multimedia support
2.Device Drivers -> <*>Multimedia support ->[*]Remote controller decoders
3.Device Drivers -> <*>Multimedia support ->[*]Remote controller decoders -> <*> Enable IR raw decoder for the NEC protocol
4.Device Drivers -> <*>Multimedia support ->[*]Remote controller decoders -> <*> Enable IR raw decoder for the RC-5 protocol
5.Device Drivers -> <*>Multimedia support ->[*]Remote Controller devices
6.Device Drivers -> <*>Multimedia support ->[*]Remote Controller devices -> <*> SUNXI IR remote control
7.Device Drivers -> <*>Multimedia support ->[*]Remote Controller devices -> <*> SUNXI IR Legacy feature
8.Device Drivers -> <*>Multimedia support ->[*]Remote Controller devices -> <*> sunxi multi support
```

3.1.2 Device Tree 配置

在 soc 节点下配置 s_cir 节点属性，其中 ir_protocol_used 属性配置红外协议，主要是 NEC(0x0) 和 RC5(0x1) 两种协议，这个属性可以不配置，不配置则默认使用 NEC 协议。

```
s_cir0: s_cir@07040000 {
        compatible = "allwinner,s_cir";
        reg = <0x0 0x07040000 0x0 0x400>;
        interrupts = <GIC_SPI 109 IRQ_TYPE_LEVEL_HIGH>;
        pinctrl-names = "default";
        pinctrl-0 = <&s_cir0_pins_a>;
        clocks = <&clk_hosc>,<&clk_cpurcir>;
        supply = "vcc-pl";
        supply_vol = "3300000";
        status = "okay";
        ir_protocol_used = <0>
};
```

3.1.3 Android multi_ir 配置

multi_ir 是 android 的一个服务，用于适配多遥控器，如果需要添加此功能，需要在方案下添加以下配置：

```
# utils, add multi_ir to recovery
PRODUCT_PACKAGES += \
    multi_ir \
    multi_ir.recovery \
    libmultiir_jni \
    libmultiirservice \
```

3.1.4 Android 按键功能的映射文件

multi_ir 的按键映文件主要放在 vendor/aw/homlet/hardware/input/multi_ir/keylayout 目录下，以 customer_ir_xxxx.kl 命名的文件是不同遥控器的映射文件，xxxx 是底层驱动识别到的遥控器 id，随着事件上报。sunxi-ir.kl 则是 multi_ir 映射底层上报的键值为统一的 scancode。sunxi-ir-uinput.kl 是 inputflinger 所读取的映射文件。使用此功能时需要将这些映射文件放入到机器内部，应如下配置：

```
SUNXI_VENDOR_KL_DIR := vendor/aw/homlet/hardware/input/multi_ir/keylayout
PRODUCT_COPY_FILES += \
    $(SUNXI_VENDOR_KL_DIR)/virtual-remote.kl:system/usr/keylayout/virtual-remote.kl \
    $(SUNXI_VENDOR_KL_DIR)/sunxi-ir.kl:system/usr/keylayout/sunxi-ir.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_9f00.kl:system/usr/keylayout/customer_ir_9f00.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_dd22.kl:system/usr/keylayout/customer_ir_dd22.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_fb04.kl:system/usr/keylayout/customer_ir_fb04.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_ff00.kl:system/usr/keylayout/customer_ir_ff00.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_4cb3.kl:system/usr/keylayout/customer_ir_4cb3.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_bc00.kl:system/usr/keylayout/customer_ir_bc00.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_fc00.kl:system/usr/keylayout/customer_ir_fc00.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_ir_2992.kl:system/usr/keylayout/customer_ir_2992.kl \
    $(SUNXI_VENDOR_KL_DIR)/customer_rc5_ir_04.kl:system/usr/keylayout/customer_rc5_ir_04.kl \
    $(SUNXI_VENDOR_KL_DIR)/sunxi-ir-uinput.kl:system/usr/keylayout/sunxi-ir-uinput.kl \
```

3.1.5 新增遥控器适配

当需要兼容新的遥控器，只要新增一个新的 customer_ir_xxxx.kl 文件，而文件主要内容应如下：

```
key 25 BACK
key 0 MENU
key 19 DPAD_CENTER
key 26 DPAD_DOWN
key 22 DPAD_UP
key 17 HOME
key 81 DPAD_LEFT
key 80 DPAD_RIGHT
key 24 VOLUME_UP
key 16 VOLUME_DOWN
key 15 APPS
key 67 CONTACTS
key 64 POWER
 ……
key 34 ZOOM_OUT
key 35 INFO
```

其中这三列字符串分别表示事件类型 (KeyEvent)、scancode 和事件 lable。multi_ir 是根据 lable 来进行映射的，sunxi-ir.kl 中有所支持的所以事件 lable。新增遥控器主要修改 scancode。scancode 的获取方式可以通过机器执行 getevent -l (sunxi-ir 所对应的设备节点) 来获取，如下：

```
//getevent -l /dev/input/event1
EV_REP REP_DELAY 00000000
EV_REP REP_PERIOD 00000000
EV_MSC MSC_SCAN 01fe0116
EV_SYN SYN_REPORT 00000000
EV_MSC MSC_SCAN 00fe0116
EV_SYN SYN_REPORT 00000000
EV_MSC MSC_SCAN 01fe0150
EV_SYN SYN_REPORT 00000000
EV_MSC MSC_SCAN 00fe0150
EV_SYN SYN_REPORT 00000000
EV_MSC MSC_SCAN 01fe011a
EV_SYN SYN_REPORT 00000000
EV_MSC MSC_SCAN 00fe011a
EV_SYN SYN_REPORT 00000000
```

其中 MSC_SCAN 所上报的就是我们所要的数据，由 8 位 16 进制数据组成 (32bit)，24 ~31bit 表示按下状态，0 表示松开，1 表示按下。8 ~ 23bit 表示设备 id，根据这个 id 生成新的 customer_ir_xxxx.kl，0 ~ 7bit 就是 scancode，对应 kl 文件的第二列数据。

3.2 GPIO 配置

3.2.1 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 longan/device/config/chips/chip-name/configs/board-name/sys_config.fex 文件中，类似如下的配置信息：

```
;-----------------------------------------------------------------------------
;userspace gpio interface for android
;-----------------------------------------------------------------------------
[gpio_para]
gpio_para_used = 1
compatible = "allwinner,sunxi-init-gpio"
gpio_num = 2
gpio_pin_1 = port:PL10<1><default><default><0>
gpio_pin_2 = port:PA15<1><default><default><1>
normal_led = "gpio_pin_2"
standby_led = "gpio_pin_1"
easy_light_used = 1
normal_led_light = 1
standby_led_light = 1
```

在这个范例中，变量 gpio_para_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_2 配置为 normal_led，gpio_pin_1 配置为 standby_led。

3.2.2 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 longan/device/config/chips/chip-name/configs/board-name/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:PL10<1><default><default><0>
gpio1 = port:PA15<1><default><default><1>
```

以上配置表示：在 boot 阶段，设置 PA15 输出高电平，PL10 输出低电平。

3.2.3 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

```
cupid-p2:/sys/class/gpio_sw # ls
PA15 PL10 normal_led standby_led
cupid-p2:/sys/class/gpio_sw # echo 1 > normal_led
cupid-p2:/sys/class/gpio_sw # echo 0 > standby_led
```

对于目录下的 normal_led/standby_led 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

3.2.4 java 层的接口

java 控制 GPIO 的接口定义在文件 Gpio.java 中，其路径为：platform/frameworks/base/services/core/java/com/aw/server/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);
```

3.2.5 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

```
int readData(const char * filePath);
int writeData(const char *data, int count, const char *filePath);
```

```
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 vendor/aw/homlet/framework/gpio/libgpio/GpioService.cpp 中的代码。

3.3 显示配置

- 720 UI 分辨率设置

```
//路径：hardware/aw/hwc2/de2family/DisplayOpr.cpp
#define MAXUIWIDTH 1280
#define MAXUIHEIGHT 720
//路径：方案目录product-name.mk
ro.sf.lcd_density=213
```

- 1080 UI 分辨率设置

```
//路径：hardware/aw/hwc2/de2family/DisplayOpr.cpp
#define MAXUIWIDTH 1920
#define MAXUIHEIGHT 1080
//路径：方案目录product-name.mk
ro.sf.lcd_density=320
```

- 配置单双显

```
//路径：方案目录product-name.mk
persist.display.policy=2
//默认是单显，2是双显；
```

- 其余请参考《H616 显示模块说明书》

3.4 WiFi/BT 配置

详细配置请参考文档《H616_Android_Q_wifi/bt 模块移植说明书》。

3.5 Camera 配置

3.5.1 Camera 参数配置

配置文件路径：device/vendor-name/device-name/configs/camera.cfg。事例内容简介：

```
;------------------------------------------------------------------------------
; 用于camera的配置
;
; 采用格式:
; key = key_value
; 注意: 每个key需要顶格写;
; key_value紧跟着key后面的等号后面, 位于同一行中;
; key_value限制大小为256字节以内;
;
;------------------------------------------------------------------------------

;------------------------------------------------------------------------------
; exif information of "make" and "model"
;------------------------------------------------------------------------------
key_camera_exif_make = MAKE_AllWinner
key_camera_exif_model = PRODUCT_BOARD

;------------------------------------------------------------------------------
; 1 for single camera, 2 for double camera
;------------------------------------------------------------------------------
number_of_camera = 2

;------------------------------------------------------------------------------
; CAMERA_FACING_BACK
; gc2355
;------------------------------------------------------------------------------
camera_id = 0

;------------------------------------------------------------------------------
; 1 for CAMERA_FACING_FRONT
; 0 for CAMERA_FACING_BACK
;------------------------------------------------------------------------------
camera_facing = 0

;------------------------------------------------------------------------------
; 1 for camera without isp(using built-in isp of Axx)
; 0 for camera with isp
;------------------------------------------------------------------------------
use_builtin_isp = 0

;------------------------------------------------------------------------------
; camera orientation (0, 90, 180, 270)
;------------------------------------------------------------------------------
camera_orientation = 90

;------------------------------------------------------------------------------
; driver device name
;------------------------------------------------------------------------------
camera_device = /dev/video0
```

```
;------------------------------------------------------------------------------
; device id
; for two camera devices with one CSI
;------------------------------------------------------------------------------
device_id = 0

used_preview_size = 1
key_support_preview_size = 640x480,320x240,176x144
key_default_preview_size = 640x480

used_picture_size = 1
key_support_picture_size = 1600x1200,1280x720,640x480,320x240
key_default_picture_size = 1600x1200

used_flash_mode = 0
key_support_flash_mode = on,off,auto
key_default_flash_mode = off

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 = 24
key_default_frame_rate = 24
used_focus_mode = 0
key_support_focus_mode = auto,infinity,macro,fixed,continuous-video,continuous-picture
key_default_focus_mode = auto

used_scene_mode = 0
key_support_scene_mode = 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-daylight
key_default_white_balance = auto

used_exposure_compensation = 1
key_max_exposure_compensation = 4
key_min_exposure_compensation = -4
key_step_exposure_compensation = 1
key_default_exposure_compensation = 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
key_horizonal_view_angle = 48.6
key_vertical_view_angle = 37.0
```

```
;------------------------------------------------------------------------
; CAMERA_FACING_FRONT
; gc0310
;------------------------------------------------------------------------
camera_id = 1

;------------------------------------------------------------------------
; 1 for camera without isp(using built-in isp of Axx)
; 0 for camera with isp
;------------------------------------------------------------------------
use_builtin_isp = 0

;------------------------------------------------------------------------
; 1 for CAMERA_FACING_FRONT
; 0 for CAMERA_FACING_BACK
;------------------------------------------------------------------------
camera_facing = 1

;------------------------------------------------------------------------
; camera orientation (0, 90, 180, 270)
;------------------------------------------------------------------------
camera_orientation = 270

;------------------------------------------------------------------------
; driver device name
;------------------------------------------------------------------------
camera_device = /dev/video0

;------------------------------------------------------------------------
; device id
; for two camera devices with one CSI
;------------------------------------------------------------------------
device_id = 1

used_preview_size = 1
key_support_preview_size = 640x480,320x240,176x144
key_default_preview_size = 640x480

used_picture_size = 1
key_support_picture_size = 640x480,320x240
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 = 24
```

```
key_default_frame_rate = 24

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,portrait,landscape,night,night-portrait,theatre,beach,snow,sunset,steadyphoto,fireworks,sports,party,candlelight,barcode
key_default_scene_mode = auto

used_white_balance = 0
key_support_white_balance = auto,incandescent,fluorescent,warm-fluorescent,daylight,cloudy-daylight
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

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
key_horizonal_view_angle = 44.3
key_vertical_view_angle = 33.9
```

media_profiles.xml 的路径：device/vendor-name/device-name/configs/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），需要注意帧率配置，我们配置 frameRate=``24" 为 24 帧，这个是多媒体要求的 camera 帧率最低的要求，这样配置我们可以满足低性能的 sensor，适用的 sensor 范围广一些。

```
<MediaSettings>
<!-- Each camcorder profile defines a set of predefined configuration parameters -->
  <!-- Back Camera -->
  <!-- Front Camera -->

  <CamcorderProfiles cameraId="0" startOffsetMs="700">

      <EncoderProfile quality="480p" fileFormat="mp4" duration="30">
          <Video codec="h264"
                 bitRate="1500000"
```

```
                width="640"
                height="480"
                frameRate="24" />

        <Audio codec="aac"
                bitRate="12200"
                sampleRate="8000"
                channels="1" />
    </EncoderProfile>

        <EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="30">
        <Video codec="h264"
                bitRate="1500000"
                width="640"
                height="480"
                frameRate="24" />

        <Audio codec="aac"
                bitRate="12200"
                sampleRate="8000"
                channels="1" />
    </EncoderProfile>


    <ImageEncoding quality="90" />
    <ImageEncoding quality="80" />
    <ImageEncoding quality="70" />
    <ImageDecoding memCap="20000000" />


  </CamcorderProfiles>

<CamcorderProfiles cameraId="1" startOffsetMs="700">

    <EncoderProfile quality="480p" fileFormat="mp4" duration="30">
        <Video codec="h264"
                bitRate="1500000"
                width="640"
                height="480"
                frameRate="24" />

        <Audio codec="aac"
                bitRate="12200"
                sampleRate="8000"
                channels="1" />
    </EncoderProfile>

    <EncoderProfile quality="timelapse480p" fileFormat="mp4" duration="30">
        <Video codec="h264"
                bitRate="1500000"
                width="640"
                height="480"
```

```
                frameRate="24" />

        <Audio codec="aac"
                bitRate="12200"
                sampleRate="8000"
                channels="1" />
    </EncoderProfile>

    <ImageEncoding quality="90" />
    <ImageEncoding quality="80" />
    <ImageEncoding quality="70" />
    <ImageDecoding memCap="20000000" />

</CamcorderProfiles>

<EncoderOutputFileFormat name="mp4" />

<!--
    If a codec is not enabled, it is invisible to the applications
    In other words, the applications won't be able to use the codec
    or query the capabilities of the codec at all if it is disabled
-->
<VideoEncoderCap name="h264" enabled="true"
    minBitRate="64000" maxBitRate="3000000"
    minFrameWidth="176" maxFrameWidth="2592"
    minFrameHeight="144" maxFrameHeight="1936"
    minFrameRate="1" maxFrameRate="30" />

<AudioEncoderCap name="aac" enabled="true"
    minBitRate="12200" maxBitRate="51200"
    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" />
<!--
    FIXME:
    We do not check decoder capabilities at present
    At present, we only check whether windows media is visible
    for TEST applications. For other applications, we do
    not perform any checks at all.
-->
<VideoDecoderCap name="wmv" enabled="true"/>
<AudioDecoderCap name="wma" enabled="true"/>
```

```
    <!--
        The VideoEditor Capability configuration:
        - maxInputFrameWidth: maximum video width of imported video clip.
        - maxInputFrameHeight: maximum video height of imported video clip.
        - maxOutputFrameWidth: maximum video width of exported video clip.
        - maxOutputFrameHeight: maximum video height of exported video clip.
        - maxPrefetchYUVFrames: maximum prefetch YUV frames for encoder,
        used to limit the amount of memory for prefetched YUV frames.
        For this platform, it allows maximum 30MB(3MB per 1080p frame x 10
        frames) memory.
    -->
    <VideoEditorCap maxInputFrameWidth="1920"
        maxInputFrameHeight="1080" maxOutputFrameWidth="1920"
        maxOutputFrameHeight="1080" maxPrefetchYUVFrames="10"/>
    <!--
        The VideoEditor Export codec profile and level values
        correspond to the values in OMX_Video.h.
        E.g. for h264, profile value 1 means OMX_VIDEO_AVCProfileBaseline
        and level 4096 means OMX_VIDEO_AVCLevel41.
        Please note that the values are in decimal.
        These values are for video encoder.
    -->
    <!--
      Codec = h.264, Baseline profile, level 4.1
    -->
    <ExportVideoProfile name="h264" profile= "2" level="4096"/>
    <!--
      Codec = h.263, Baseline profile, level 0
    -->
    <ExportVideoProfile name="h263" profile= "1" level="1"/>
    <!--
      Codec = mpeg4, Simple profile, level 5
    -->
    <ExportVideoProfile name="m4v" profile= "1" level="128"/>
</MediaSettings>
```

3.6 SD 卡配置

发布的 SDK 中支持 SD 卡和 Mirco SD（TF）卡及其兼容性卡。

3.6.1 配置文件的修改

配置文件路径：longan/device/config/chips/chip-name/configs/board-name/sys_config.fex

根据原理图进行相关配置参数的修改

```
[sdc0]
sdc0_used = 1
bus-width = 4
sdc0_d1 = port:PF00<2><1><2><default>
sdc0_d0 = port:PF01<2><1><2><default>
sdc0_clk = port:PF02<2><1><2><default>
sdc0_cmd = port:PF03<2><1><2><default>
sdc0_d3 = port:PF04<2><1><2><default>
sdc0_d2 = port:PF05<2><1><2><default>
sd-uhs-sdr50 =
sd-uhs-ddr50 =
sd-uhs-sdr104 =
;broken-cd =
;cd-inverted =
cd-gpios = port:PF06<6><1><2><default>
card-pwr-gpios = port:PB02<1><1><2><default>
sunxi-power-save-mode =
;sunxi-dis-signal-vol-sw =
max-frequency = 150000000
vmmc="vcc-card"
vqmmc="vcc-pf"
vdmmc="vcc-pf"
```

参数的意义：

配置项	配置项含义
sdc0_used=xx	SDC 使用控制：1 使用，0 不用
bus-width=xx	位宽：1-1bit，4-4bit
sdc0_d1=xx	SDC DATA1 的 GPIO 配置
sdc0_d0 = xx	SDC DATA0 的 GPIO 配置
sdc0_clk=xx	SDC CLK 的 GPIO 配置
sdc0_cmd=xx	SDC CMD 的 GPIO 配置
sdc0_d3=xx	SDC DATA3 的 GPIO 配置
sdc0_d2=xx	SDC DATA2 的 GPIO 配置
cd-gpios	card detect pin 的 GPIO 配置
vmmc	power for card vdd，从 [pmu0_regu] 中查找对应的 regulator
vqmmc	power for card io，从 [pmu0_regu] 中查找对应的 regulator
vdmmc	power for card detctect pin io，从 [pmu0_regu] 中查找对应的 regulator
sd-uhs-sdr50	UHS sdr50 mode
sd-uhs-ddr50	UHS ddr50 mode

配置项	配置项含义
sd-uhs-sdr104	UHS sdr104 mode,dafault mode is HS 50M
broken-cd	卡检测方式 --定时器定时检测卡
cd-inverted	卡检测引脚低电平有效
card-pwr-gpios	PWR 引脚
sunxi-power-save-mode	For sdio wifi,should not be set when use sdio wifi
sunxi-dis-signal-vol-sw	Not allow to change io voltage
max-frequency	max clk sdc use,unit HZ

4. 系统配置

4.1 SettingsProvider 设置

配置文件 platform/frameworks/base/packages/SettingsProvider/res/values/defaults.xml 中各项为 Android 原生属性，可通过 overlay 修改进行配置，下面列出一些常用修改

name	value	description
def_screen_off_timeout	Int(毫秒)	默认 LCD 关闭时间
def_screen_brightness	0~255	默认亮度设置
def_screen_brightness_automatic_mode	Boolean	是否默认打开自动亮度
def_wifi_on	Boolean	是否默认打开 WIFI
def_bluetooth_on	Boolean	是否默认打开蓝牙

4.2 默认配置

通过系统配置文件 platform/frameworks/base/core/res/res/values/config.xml 中各项修改系统的配置，可通过 overlay 方式进行修改。

name	value	description
config_showNavigationBar	Boolean	默认显示导航栏
config_multiuserMaximumUsers	1~8	最大用户数量
config_enableMultiUserUI	Boolean	是否支持多用户 UI（多用户时不需要设置）
config_unplugTurnsOnScreen	Boolean	拔出 usb 或电源时唤醒屏幕
config_automatic_brightness_available	Boolean	是否支持自动亮度调节
config_enableWifiDisplay	Boolean	是否支持 Miracast
config_allowAllRotations	Boolean	是否支持 4 个方向旋转
config_enableLockScreenRotation	Boolean	锁屏时是否支持旋转

4.3 屏保功能

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：

4.3.1 设置默认屏保应用

device/vendor-name/device-name/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>
```

注意：其中 config_dreamsDefaultComponent 为默认的屏保应用的包名，这个应用位 android/vendor/aw/homlet/package/WebScreensaver 目录下

device/ vendor- name/ device- name/ 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>
```

4.3.2 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/aw/homlet/package/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);
```

4.4 一键恢复功能

4.4.1 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：

4.4.2 修改 sys_config

在 longan/device/config/chips/chip-name/configs/board-name/sys_config.fex 文件里，修改配置，如：

```
;-----------------------------------------------------------------------------------
; ir_boot_recovery_used : 1: used this function 0: not used
; ir_work_mode : 模式选择
; 0: 刷机,
; 1: 一键恢复(uboot阶段),
; 2: 安卓recovery,
; 3: 安卓恢复出厂设置.
; 如果不设置，默认为安卓recovery .
; ir_press_times : ir遥控器连续按几次才生效,如果不设置默认为按1次生效
; ir_detect_time : ir遥控检测时间,单位:ms,如果不设置默认为3000ms
; ir_key_no_duplicate : ir遥控按键是否可重复，0：可重复(默认)，1：不可重复；
```

```
; 不可重复表示一个按键无论被按下几次，都只算ir_press_times的一次； ;
      不可重复的应用场景为组合按键功能，如:交替按下'菜单键'和'音量-键'进入安卓recover y.
; ir_recovery_key_code0 : ir check key code
; ir_addr_code0 : ir key addr
; you can increase ir support num, like:
; ir_recovery_key_code1, or 2, 3, but limit to 16
;----------------------------------------------------------------------------

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

;----------------------------------------------------------------------------
; boot阶段上电初始化GPIO
; used :模块使能端 置1：开启模块 置0：关闭模块
; gpiox ：上电初始化gpio （名称自定，但不能重复，并且GPIO允许可以多个）
; PH06 : 系统显示LED GPIO
;----------------------------------------------------------------------------
[boot_init_gpio]
boot_init_gpio_used = 1
gpio0 = port:PL10<1><default><default><0>
gpio1 = port:PA15<1><default><default><1>

;----------------------------------------------------------------------------
; 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
; trigger: 触发电平 0: 低电平触发(默认) 1: 高电平触发
; recovery_key: 按键配置 （例如：recovery_key= port:PH16<0><default>）
;----------------------------------------------------------------------------
[recovery_para]
used = 0
mode = 2
trigger = 0
recovery_key = port:PL04<0><default><default><default>
```

将 mode 改成 2，即按下 recovery 键后进入一键恢复模式

4.4.3 添加 sysrecovery 分区

在 longan/device/config/chips/chip-name/configs/board-name/android/sys_partition.fex 文件里

```
;----------------------------->nandk, system image backup—添加的分区
    [partition]
     name = sysrecovery
     size = 4194304
     downloadfile = "sysrecovery.fex"
     verify = 0
```

4.4.4 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>
```

4.4.5 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。

5. 打包发布

5.1 安全方案配置

在方案配置文件 device/vendor-name/device-name/device_name.mk 中添加或修改一下宏为 true：

BOARD_HAS_SECURE_OS := true

5.2 编译固件

编译内核及芯片相关：

```
# cd longan
# ./build.sh config（选择对应平台内核）
# ./build.sh
```

编译 android：

```
# cd android
# source ./build/envsetup.sh
# lunch（选择对应平台方案配置）
# extract-bsp
# make -j16
```

5.3 调试 debug

5.3.1 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 logcat 在/data/misc/logd 目录，kmsg 在/data/vendor/kmsgd/目录在文件中加入 PRODUCT_DEBUG := true 即可打开该功能，eng 及 userdebug 固件默

认开启

5.3.2 生成 debug 固件

编译 android 后，pack -d 即可生成 debug 固件，该固件将串口引入卡口打印出来，配合配套的工具即可实时查看 log 信息。

5.3.3 使用 fastboot

获取 fastboot 工具：

1. 建议更新最新版本 Android SDK tools 中的 fastboot 工具
2. 在 android 源代码编译过后的生成文件获得 (platform/out/host/linux-x86/bin/fastboot)

进入 bootloader 状态：

在 adb shell 中，使用命令 reboot bootloader 即可进入 fastboot 模式。

安装驱动后，在 PC 端执行 fastboot 命令即可进行 fastboot 操作。

安全固件使用 fastboot 无法烧写 boot、system、recovery 等系统分区。

fastboot 常用命令：

```
usage: fastboot [ <option> ] <command>
commands:
  update <filename> reflash device from update.zip
  flash <partition> [ <filename> ] write a file to a flash partition
  erase <partition> erase a flash partition
  format <partition> format a flash partition
  help show this help message
```

5.4 发布

5.4.1 发布固件流程

发布固件即可用作量产使用的固件，同时也支持 OTA 升级功能。发布时需要使用该流程进行发布。

使用 6.1 编译固件流程后，使用命令：

```
# pack4dist [-v]
```

即可生成固件及对应版本的 OTA 包。（注：上述 -v 参数用于启用安全系统校验）

品牌签名：如果品牌商有自己的系统签名文件，把相关签名文件放入 platform/vendor/security 目录。签名文件可参考 platform/build/target/product/README 文档

5.4.2 OTA 包

OTA 包包含差分包和完整包，以下是各个名词定义：

- 目标文件包（target-files-package）：包含固件完整的编译后目标文件的包。
- 差分包（incremental-package）：将基础版本与新版本固件之间的差别制作的补丁包。
- 完整包（full-package）：将新版本固件的完整补丁包。

OTA 包生成过程：

使用 pack4dist 后会自动生成目标文件包（target-files-package）路径为：

```
$OUT/obj/PACKAGING/target_files_intermediates/$TARGET_PRODUCT-target_files-$DATE.zip
```

若包含签名目标文件包，则路径为：

```
$OUT/signed_target_files-$DATE.zip
```

注：生成的 target_files.zip 文件需要与固件一同保存，用于后续生成 OTA 包。

完整包路径为：

```
$OUT/$TARGET_PRODUCT-full_ota-$DATE.zip
```

完整包生成命令：

```
# ./build/tools/releasetools/ota_from_target_files [ -k vendor/security/releasekey ] target.zip ota.zip
```

差分包生成命令：

```
# ./build/tools/releasetools/ota_from_target_files [ -k vendor/security/releasekey ] -i origin.zip target.zip inc.zip
```

注：其中 vendor/security 为签名 key 放置路径，origin.zip 为基础版本（即需要升级的版本）的目标文件包，target.zip 为当前版本的目标文件包。

5.5 使用 OTA 包升级

选择 ``设置 > 备份和重置 >Recovery 模式" 重启进入 Recovery。

或 PC 端通过 adb reboot recovery 命令，重启进入 recovery。

5.5.1 Apply update from ADB

1. 将固件放在 PC 端，如：E:/update.zip。
2. 进入 Recovery。
3. 选择 Apply update from ADB。
4. 打开 cmd，并输入 adb sideload E:/update.zip。

5. 等待打印 Install from ADB complete. 升级完成。
6. 选择 reboot system now 重启并进入 android。

5.5.2 Apply update from TFcard or USB

1. 将固件放入 TF 卡或 U 盘中。
2. 进入 recovery
3. 插入 TF 卡或 U 盘
4. 在 Recovery 菜单中选择 Apply update from SD card
5. 找到升级包的路径并选择开始升级。
6. 等待打印 Install from SD card complete. 升级完成。
7. 选择 reboot system now 重启并进入 android。

6. 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.