全志科技
Allwinner Technology

A64

快速移植指南（Android M）

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2014-10-15		初始版本
V2.0	2015-02-02		Android L
V3.0	2015-12-22		Android M
V3.1	2016-2-14		A64 Android M

目录

1. 概述

1.1. 名词解释

A64 平台快速移植文档，本文基于 A64 tulip-p1 方案。

1. vendor-name
   softwinner
2. device-name
   tulip-p1
3. product-name
   tulip_p1

1.2. 编写目的

为了快速移植及配置 A64 平台方案，可以参照文档中的一些配置项进行简单的定制化修改。实现快速移植方案。

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/value/config.xml 中的某一项，则在 overlay 中创建对应的路径：overlay/framework/base/core/res/res/value/config.xml 并添加要修改的一向配置，如：

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
    <integer name="config_multiuserMaximumUsers">4</integer>
</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/preinstal 目录下。

注：apk 名字不能含有中文、空格等特殊字符。

2.2.1. 默认预装 APK

APK	说明
ES 文件管理器	文件管理器
VideoPlayer	视频播放器
DragonAging	工厂测试应用
DragonFire	工厂测试应用
DragonPhone	工厂测试应用

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。

2.2.2. 预装到 system/app 目录

1. 在目录 device/softwinner/common/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_PRIVILEGED_MODULE :=
```

```
LOCAL_CERTIFICATE := PRESIGNED
#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）
#LOCAL_REQUIRED_MODULES :=
LOCAL_PREBUILT_JNI_LIBS := @lib/$(TARGET_ARCH)/libjni.so（使用@直接引用 apk 内部的 so）
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）
```

2.2.3. 预装到 system/preinstall 目录

1.这些 apk 将在系统第一次启动时安装到用户 data 目录，用户可自行卸载。

2.将 apk 放入 device/vendor-name/common/prebuild/preinstallapk/ 目录中，并且在同目录的 Android.mk 文件中加入：

```
include $(CLEAR_VARS)
LOCAL_MODULE := APK_MODULE_NAME（模块的唯一名字）
LOCAL_MODULE_TAGS := optional
LOCAL_CERTIFICATE := PRESIGNED
LOCAL_MODULE_PATH := $(TARGET_OUT)/preinstall
LOCAL_MODULE_CLASS := APPS
LOCAL_SRC_FILES := name.apk（同目录下 apk 的文件名）
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
include $(BUILD_PREBUILT)
```

3.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）
```

2.3. 配置 data 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。

注：多用户方案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。

2.4. 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：lichee/tools/pack/chips/sunxi/configs/device-name/bootlogo.bmp

2.5. 修改启动动画

将动画放入：device/vendor-name/device-name/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 文件内容如下：

```
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。

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，内容如下：

```
card=0
device=0
period_size=2048
period_count=2
mixer "Speaker Function"=spk
```

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。打包完之后修改其权限值：chmod 777 bootanimation.zip

2.6. 修改充电图标

在 android 目录下执行：

python bootable/recovery/interlace-frames.py battery1.png battery2.png ... batteryn.png battery_scale.png

然后将生成的 battery_scale.png 替换 system/core/healthd/images/目录下 battery_scale.png

其中[battery1.png battery2.png ... batteryn.png]为充电动画的图标。

如图片数量有变化则需修改代码 system/core/healthd/healthd_mode_charger.cpp

2.7. 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。

2.7.1. 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。

2.7.2. 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>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。

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。

2.8. 内存自适应

2.8.1. 预留内存设置

一般不需要改，如果确实是预留内存不足，将导致 ion 分配失败，可以修改这里加大预留：

lichee/tools/pack/chips/sunxi/configs/default/env.cfg

```
ion_cma_list="120m,176m,512m"
```

即：512M 内存平台需要预留内存为 120M，1G 内存平台需要预留内存为 176M，2G 内存平台需要预留 512M。

如果需要调整预留内存大小，可以直接修改上述点。注意：修改时 setargs_nand 和 setargs_mmc 两行都要修改。

注：预留内存，指的是 CMA 预留内存，这部分内存空闲时，可以被系统用。

2.8.2. 内存参数调节

一般不需要改. 这里为使客户理解, 对内存参数进行说明:

文件: device/softwinner/common/config/config_mem.ini

(1) 虚拟机参数:

[dalvik_512m], [dalvik_1024m], [dalvik_2048m]分别表示 512M, 1G, 2G 内存方案上的虚拟机参数. 具体包括 dalvik.vm.heapsize, dalvik.vm.heapstartsize, dalvik.vm.heapgrowthlimit, dalvik.vm.heapminfree, dalvik.vm.heapmaxfree. 这些参数含义可以在网上找到.

```
[dalvik_512m]     (512M 方案虚拟机堆参数)
dalvik.vm.heapsize=128m
dalvik.vm.heapstartsize=8m
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=512k
dalvik.vm.heapmaxfree=2m
dalvik.vm.dex2oat-Xmx=128m
[dalvik_1024m]      (1024M 方案虚拟机堆参数)
dalvik.vm.heapsize=384m
dalvik.vm.heapstartsize=8m
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=512k
dalvik.vm.heapmaxfree=8m
[dalvik_2048m]      (2048M 方案虚拟机堆参数)
dalvik.vm.heapsize=512m
dalvik.vm.heapstartsize=8m
dalvik.vm.heapgrowthlimit=192m
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=2m
dalvik.vm.heapmaxfree=8m
```

(2) hwui 参数:

[hwui_800], [hwui_1024], [hwui_1280], [hwui_1920], [hwui_2048], [hwui_2560]. 数字表示 lcd 分辨率的长边像素. 比如 1280*800, 1280*720 的屏, 其对应 hwui 参数是[hwui_1280]. 更详细的参数介绍可参考：https://source.android.com/devices/tech/debug/tuning.html.

```
[hwui_800]
ro.hwui.texture_cache_size=9
ro.hwui.layer_cache_size=6
ro.hwui.r_buffer_cache_size=1
ro.hwui.path_cache_size=4
ro.hwui.drop_shadow_cache_size=1

[hwui_1024]
ro.hwui.texture_cache_size=16
ro.hwui.layer_cache_size=10
```

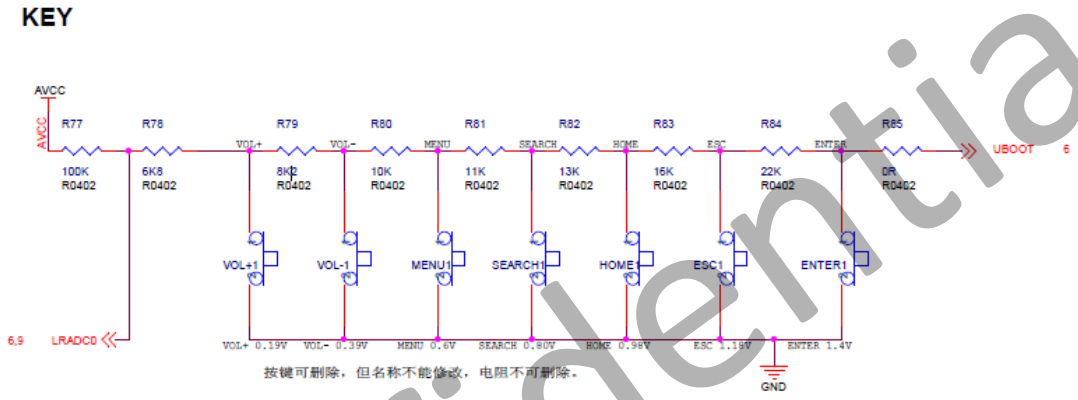
ro.hwui.r_buffer_cache_size=1.5
ro.hwui.path_cache_size=6
ro.hwui.drop_shadow_cache_size=1.5
[hwui_1280] (默认设置是针对 1280*800，所以大部分采用默认设置）
[hwui_1920]
ro.hwui.texture_cache_size=54
ro.hwui.layer_cache_size=36
ro.hwui.r_buffer_cache_size=5
ro.hwui.path_cache_size=24
ro.hwui.drop_shadow_cache_size=5
ro.hwui.gradient_cache_size=1
ro.hwui.texture_cache_flushrate=0.5
ro.hwui.text_small_cache_width=1024
ro.hwui.text_small_cache_height=1024
ro.hwui.text_large_cache_width=2048
ro.hwui.text_large_cache_height=1024
[hwui_2048]
ro.hwui.texture_cache_size=72
ro.hwui.layer_cache_size=48
ro.hwui.r_buffer_cache_size=7
ro.hwui.path_cache_size=32
ro.hwui.drop_shadow_cache_size=6
ro.hwui.gradient_cache_size=1
ro.hwui.texture_cache_flushrate=0.5
ro.hwui.text_small_cache_width=1024
ro.hwui.text_small_cache_height=1024
ro.hwui.text_large_cache_width=2048
ro.hwui.text_large_cache_height=1024
[hwui_2560]
ro.hwui.texture_cache_size=72 (texture cache，MB，至少 5 于 width * height * 32bit)
ro.hwui.layer_cache_size=48 (layer cache，MB，至少 4 倍于 width * height * 32bit)
ro.hwui.r_buffer_cache_size=8 (renderbuffer cache，MB，至少 2 倍于 width * height * 8bit)
ro.hwui.path_cache_size=32 (path cache，MB，至少 1 倍于 width * height * 32bit)
ro.hwui.drop_shadow_cache_size=6 (text drop shadow cache，MB，至少 2 倍于 width * height * 8bit)
ro.hwui.gradient_cache_size=1 (gradient cache，MB)
ro.hwui.texture_cache_flushrate=0.4 (flush 后保留的 texture cache 的比例）
ro.hwui.text_small_cache_width=1024 (pixels)
ro.hwui.text_small_cache_height=1024 (pixels)
ro.hwui.text_large_cache_width=2048 (pixels)
ro.hwui.text_large_cache_height=1024 (pixels)

3. 模块配置

3.1. 自定义按键配置

3.1.1. 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：



3.1.2. 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,                      //key 2, 7 个，  8-14
    2,2,2,2,2,2,2,                      //key 3, 7 个，  15-21
    3,3,3,3,3,3,                        //key 4, 6 个，  22-27
    4,4,4,4,4,4,                        //key 5, 6 个，  28-33
    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     //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);
```

3.1.3. Android 按键功能的映射

映射文件为 device/vendor-name/device-name/configs/sunxi-keyboard.kl：

```
key 114     VOLUME_UP
key 139     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。（附件列，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）

3.2. WiFi/BT 配置

请参考文档 《A64 WiFi-BT-GPS 配置说明书》。

3.3. LCD Panel 配置

配置文件的修改

配置文件 ： lichee/tools/pack/chips/sunxi/configs/device-name/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>
...
```

请参考文档 《A64 LCD 使用说明书》。

3.4. Touch Panel 配置

发布的 SDK 中，默认有对 FT5x 系列（敦泰）、GT81x/GT82x/GT9xx/GT9xxf（汇顶）、GSLx680(思立微)等 ctp 的支持。

3.4.1. 配置文件的修改

配置文件目录：lichee/tools/pack/chips/sunxi/configs/device-name/sys_ config.fex
ctp 的配置文件示例如下。

```
[ctp_para]
ctp_used                = 1
ctp_name                = "ft5x_ts"
ctp_twi_id              = 0
ctp_screen_max_x        = 768
ctp_screen_max_y        = 1024
ctp_revert_x_flag       = 0
ctp_revert_y_flag       = 0
ctp_exchange_x_y_flag = 0

ctp_int_port            = port:PB05<4><default><default><default>
ctp_wakeup              = port:PH01<1><default><default><1>
ctp_power_ldo           = "axp22_eldo1"
ctp_power_ldo_vol       = 3000
ctp_power_io            =
```

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
ctp_power_ldo	Tp ic 使用的 axp ldo 名称
ctp_power_ldo_vol	Tp ic 使用电源的电压
ctp_power_io	使用 axp 的 gpio0 或 gpio1 供电

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。

Tp ic 采用独立供电的方式，需配置供电的 ldo 并设置电压。

3.4.2. Android 层的配置修改

1）驱动的加载

如果使用了自动检测功能，只需在 device/vendor-name/device-name/init.sunxi.rc 中添加以下命令：

```
init_dev_detect
```

该命令会调用自动检测模块 sw_devices.ko，加载此模块后会自动检测使用的 tp，并根据检测结果加载相应的 tp 驱动，并把相关的驱动的信息保存到设备列表中，生成 devlist.info 文件，下次启动时，将直接读取 devlist.info 中的信息，不再加载 sw_devices.ko 模块。

如果采用手动加载方式，则在 device/vendor-name/device-name/init.sunxi.rc 文件中加入装载驱动模块的语句：

```
insmod /system/vendor/modules/gslX680.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，获取设备的名称为"gslX680", "gt82x", "ft5x_ts", "sunxi-ts", "gt818_ts", "tu_ts", "sw-ts", "gt9xx", "gt9xxf_ts", "aw5306_ts"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 product-name.mk 拷贝语句如下所示：

```
PRODUCT_COPY_FILES += \
device/vendor-name/device-name/sw-keyboard.kl:system/usr/keylayout/sw-keyboard.kl \
device/vendor-name/device-name/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 product-name.mk 拷贝语句如下所示：

```
# input device config
PRODUCT_COPY_FILES += \
device/vendor-name/device-name/sw-keyboard.kl:system/usr/keylayout/sw-keyboard.kl \
device/vendor-name/device-name/tp.idc:system/usr/idc/tp.idc \
device/vendor-name/device-name/gsensor.cfg:system/usr/gsensor.cfg
```

3.4.3. 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。

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），gt9xx，gt9xxf 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 gt9xx 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。

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。

3.5. G-Sensor 配置

发布的 SDK 中已添加了对 MMA7660、MMA8452、MMA8652、MMA8653、BMA250 、BMA250e、LSM9ds0 等 G-Sensor 的支持，需要客户根据需要做如下配置。

3.5.1. 打包配置文件修改

配置文件目录：lichee/tools/pack/chips/sunxi/configs/device-name/sys_config.fex

G sensor 的配置文件示例如下。

```
[gsensor_para]
gsensor_used            = 1
gsensor_twi_id          = 1
gsensor_twi_addr        = 0x18
gsensor_int1            = port:PB06<4><1><default><default>
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。

3.5.2. Android 层配置修改

以 LSM9ds0 为例：

Sensor 驱动是自动加载的，自动加载部分配置好后，不需要手动 insmod。

1）sensor HAL 层对 sensor 打开、关闭等操作需要相关文件的操作权限，因为 sensor 驱动中创建的文件节点只有 644 权限，HAL 无法实现打开、关闭 sensor 等操作，需要更改 sensor 的用户组。用户组更改由 device/vendor-name/device-name/sensor.sh 完成。用户在调试过程中，可根据具体 sensor 文件节点的实际路径更改 sensor.sh 文件。

```
#!/sbin/busybox sh

#light
chown system:system /sys/class/input/event7/device/data
chown system:system /sys/class/input/event7/device/delay
chown system:system /sys/class/input/event7/device/enable
chown system:system /sys/class/input/event7/device/resolution
chown system:system /sys/class/input/event7/device/compensate
```

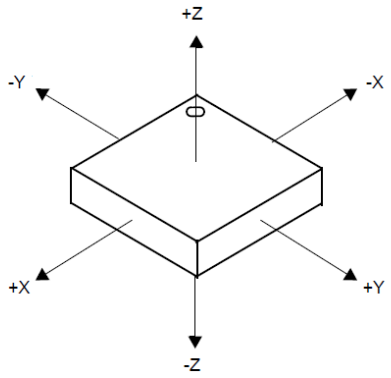
```
#acc
Chown                                                    system:system
/sys/bus/i2c/devices/1-001d/accelerometer/enable_device
chown system:system /sys/bus/i2c/devices/1-001d/accelerometer/pollrate_ms
chown system:system /sys/bus/i2c/devices/1-001d/accelerometer/full_scale

#mag
chown                                                    system:system
/sys/bus/i2c/devices/1-001d/magnetometer/enable_device
chown system:system /sys/bus/i2c/devices/1-001d/magnetometer/pollrate_ms
chown system:system /sys/bus/i2c/devices/1-001d/magnetometer/full_scale

#gyro
chown system:system /sys/bus/i2c/devices/1-006a/enable_device
chown system:system /sys/bus/i2c/devices/1-006a/enable_polling
chown system:system /sys/bus/i2c/devices/1-006a/pollrate_ms
chown system:system /sys/bus/i2c/devices/1-006a/range
chown system:system /sys/bus/i2c/devices/1-006a/fifo_samples
chown system:system /sys/bus/i2c/devices/1-006a/fifo_mode
chown system:system /sys/bus/i2c/devices/1-006a/reg_addr
chown system:system /sys/bus/i2c/devices/1-006a/reg_value
```

2）方向的调整：

在 device/vendor-name/device-name/gsensor.cfg 中，以 lsm9ds0 方向为例进行说明。

```
;name:lsm9ds0_acc
gsensor_name = lsm9ds0_acc    //标示用 lsm9ds0_acc gsensor
gsensor_direct_x = false      //如果 x 轴反向，则置 false
gsensor_direct_y = false      //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
```

Lsm9ds0 实际方向如下（参见 lsm9ds0 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。

3.6. Camera 配置

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

3.6.1. 打包配置文件修改

配置文件位置：lichee/tools/pack/chips/sunxi/configs/device-name/sys_config.fex

A80 上面只有 CSI0，驱动里面需要配置[csi0_para]段落。

```
[csi0]

vip_used                    = 1
vip_mode                    = 0
vip_dev_qty                 = 2
vip_define_sensor_list      = 0
vip_csi_pck                 = port:PE00<2><default><default><default>
vip_csi_mck                 = port:PE01<2><default><default><default>
vip_csi_hsync               = port:PE02<2><default><default><default>
vip_csi_vsync               = port:PE03<2><default><default><default>
vip_csi_d0                  = port:PE04<2><default><default><default>
vip_csi_d1                  = port:PE05<2><default><default><default>
vip_csi_d2                  = port:PE06<2><default><default><default>
vip_csi_d3                  = port:PE07<2><default><default><default>
vip_csi_d4                  = port:PE08<2><default><default><default>
vip_csi_d5                  = port:PE09<2><default><default><default>
vip_csi_d6                  = port:PE10<2><default><default><default>
vip_csi_d7                  = port:PE11<2><default><default><default>

vip_dev0_mname              = "gc2035"
vip_dev0_pos                = "rear"
vip_dev0_lane               = 1
vip_dev0_twi_id             = 2
vip_dev0_twi_addr           = 0x78
vip_dev0_isp_used           = 0
vip_dev0_fmt                = 0
vip_dev0_stby_mode          = 0
vip_dev0_vflip              = 0
vip_dev0_hflip              = 0
vip_dev0_iovdd              = "axp22_dldo3"
vip_dev0_iovdd_vol          = 2800000
vip_dev0_avdd               = "axp22_dldo3"
vip_dev0_avdd_vol           = 2800000
vip_dev0_dvdd               = "axp22_eldo2"
vip_dev0_dvdd_vol           = 1800000
vip_dev0_afvdd              = "axp22_dldo3"
vip_dev0_afvdd_vol          = 2800000
vip_dev0_power_en           =
vip_dev0_reset              = port:PE14<1><default><default><0>
```

各项配置请参考 sys_config 中的注释。

其中重要的几项是电源和 gpio，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、控制逻辑和供电 ldo 和电压。

3.6.2. Android 层的配置修改

Android 中，在 device/vendor-name/device-name/init.sunxi.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 device/vendor-name/device-name/ueventd.sun8i.rc 文件中改变相关设备节点的权限：

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

3.6.3. 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：

```
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,ste
adyphoto,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 = 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：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）

```
<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" />
        <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>
```

3.7. 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；

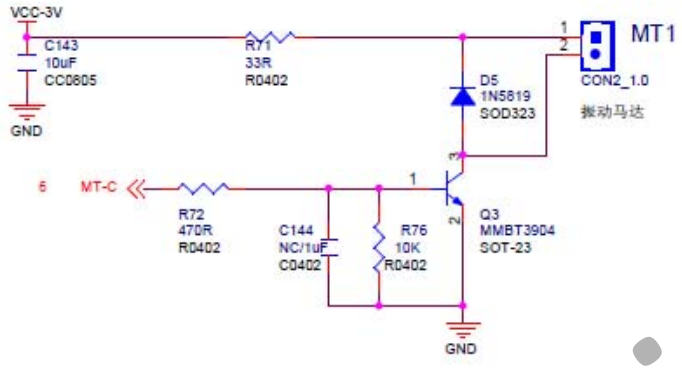


图 9.7.1

3.7.1. 配置文件修改

配置文件路径：lichee/tools/pack/chips/sunxi/configs/device-name/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；
```

3.7.2. Android 层配置修改

在 device/vendor-name/device-name/init.sunxi.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/sunxi-vibrator.ko
```

3.8. SD 卡配置

发布的 SDK 中支持 SD 卡和 Mirco SD（TF）卡及其兼容性卡，A64 中支持四组通用的 mmc/sd 卡接口。

配置文件的修改

配置文件路径：lichee/tools/pack/chips/sunxi/configs/device-name/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:PF06<0><1><2><default>
sdc_use_wp              = 0
sdc_wp                  =
sdc_isio                = 0
sdc_regulator           = "none"
sdc_power_supply    = "vcc-card"
```

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。

3.9. CTP 与 Sensor 自动检测使用说明

3.9.1. CTP 自适应使用说明

1）sys_config.fex 文件的配置

第一步：若 sys_config.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。sys_config.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
gslX680                 = 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 device/vendor-name/device-name/init.sunxi.rc 中添加以下命令：

```
init_dev_detect
```

该命令会调用自动检测模块 sw_devices.ko，加载此模块后会自动检测使用的 tp，并根据检测结果加载相应的 tp 驱动，并把相关的驱动的信息保存到设备列表中，生成 devlist.info 文件，下次启动时，将直接读取 devlist.info 中的信息，不再加载 sw_devices.ko 模块。

为了快速的检测到设备，此命令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
    init_dev_detect
#insmod video driver
  insmod /system/vendor/modules/cedarx.ko
......
```

注意：使用自动检测时，使用 init_dev_detect 命令后，在 init.sunxi.rc 文件中需要删除掉之前已经加载的 ctp 驱动的语句。

3.9.2. 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：gsneosr_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 device/vendor-name/device-name/init.sunxi.rc 中添加以下命令：

```
init_dev_detect
```

该命令会调用自动检测模块 sw_devices.ko，加载此模块后会自动检测使用的 gsensor，并根据检测结果加载相应的 gsensor 驱动，并把相关的驱动的信息保存到设备列表中，生成 devlist.info 文件，下次启动时，将直接读取 devlist.info 中的信息，不再加载 sw_devices.ko 模块。

为了快速的检测到设备，此命令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
    init_dev_detect
#insmod video driver
  insmod /system/vendor/modules/cedarx.ko
......
```

注意：使用自动检测时，使用 init_dev_detect 命令后，在 init.sunxi.rc 文件中需要删除掉之前已经加载的 gsensor 驱动的语句。

3.9.3. 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 device/vendor-name/device-name/product-name.mk 文件中。如下：

```
......
# for recovery
PRODUCT_COPY_FILES += \
    device/vendor-name/device-name/modules/modules/nand.ko:root/nand.ko \
    device/vendor-name/device-name/modules/modules/sunxi_tr.ko:root/sunxi_tr.ko \
```

```
    device/vendor-name/device-name/modules/modules/disp.ko:root/disp.ko \
    device/vendor-name/device-name/modules/modules/gt9xxnew_ts.ko:recovery/root/gt9xxnew_ts.ko \
    device/vendor-name/device-name/modules/modules/sw-device.ko:recovery/root/sw-device.ko \
......
```

其中 recovery.fstab 为 recovery 功能相关文件。

sunxi_tr.ko，disp.ko，为显示相关驱动。

gt9xxnew_ts.ko 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/vendor-name/device-name/modules/modules/gt9xxnew_ts.ko:gt9xxnew_ts.ko \
    device/vendor-name/device-name/modules/modules/screen.ko:screen.ko \
    device/vendor-name/device-name/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 device/vendor-name/device-name/init.recovery.sunxi.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
    insmod /nand.ko
    insmod /sunxi-ts.ko //加载相对应的 tp 驱动
    insmod /sw-device.ko debug_mask=0xff ctp_mask=1
    insmod /sunxi_tr.ko
    insmod /disp.ko
......
```

4. Settings 设置

4.1. 原生设置

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. Allwinner 平台设置

配置文件 platform/frameworks/base/packages/SettingsProvider/res/values/custom_config.xml 中各项为平台自定义属性，可通过 overlay 修改进行配置。

name	value	description
def_time_12_24	12/24	默认显示 12 小时还是 24 小时
def_gesture_screenshot_enable	1/0	默认打开手势截屏功能
def_gesture_screenrecord_enable	1/0	默认打开手势录屏功能
def_low_power_mode_trigger_level	0~100	进入节能模式电量

4.3. 系统配置

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

5. Launcher 及界面设置

5.1. 默认壁纸设置

替换文件 platform/frameworks/base/core/res/res/drawable-sw xxxdp-nodpi/default_wallpaper.jpg，可通过 overlay 方式将文件放在 device/vendor-name/device-name/overlay/frameworks/base/core/res/res/drawable-swxxxdp-nodpi/default_wallpaper.jpg。

5.2. 添加壁纸

准备壁纸及壁纸的缩略图放进壁纸存放目录 platform/packages/apps/Launcher3/WallpaperPicker/res/drawable-nodpi，并按照文件夹内文件命名，分别为 wallpaper_xxx.jpg 与 wallpaper_xxx_small.jpg

在 platform/packages/apps/Launcher3/WallpaperPicker/res/values-nodpi/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_xxx</item>
    </string-array>
</resources>
```

注：可通过 overlay 方式修改，具体请参照 2.1overlay 说明。

5.3. Launcher 默认图标和快捷栏设置

修改文件 platform/packages/apps/Launcher3/res/xml/default_workspace_5x6.xml，文件中各配置项含义如下：

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

注：可通过 overlay 方式修改，具体请参照 2.1overlay 说明。

6. 系统调试

6.1. 生成 debug 固件

Android 编译时使用 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。

6.2. 使用 fastboot

6.2.1. 进入 fastboot 模式

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。
注意：使用-i 0x1f3a 参数指定 ID。
安全固件使用 fastboot 无法烧写 boot、system、recovery 等系统分区。

6.2.2. 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
```

7. Declaration