



PET_RK3568_P01 开发板 系统开发手册

一、安卓系统开发

1、Uboot 研发

详见 芯片原厂文档\common\UBOOT 目录下相关文件

Uboot 源码位于 u-boot 目录下

2、内核研发

内核设备树文件位置：

kernel-4.19/arch/arm64/boot/dts/rockchip/rk3568.dtsi

kernel-4.19/arch/arm64/boot/dts/rockchip/rk3568-gzpeite.dtsi

默认内核配置 kernel-4.19/arch/arm64/configs/rk3568_gzpeite_defconfig

```
cd kernel-4.19
make menuconfig
修改相关配置并保存，同时要手动将新的配置文件复制到 config 目录
cp -rf .config arch/arm64/configs/rk3568_gzpeite_defconfig
```

其他内核相关研发请参考 芯片原厂文档 目录下的相关文档

3、修改启动 logo

用新的 bmp 文件替换 kernel 目录下的 logo.bmp 和 logo_kernel.bmp 文件，图片分辨率不要超过屏幕分辨率。

4、修改开机动画

将动画文件 bootanimation.zip 复制到 device/rockchip/common/bootshutdown 目录下

修改 device\rockchip\rk356x\rk3568_s\rk3568_s.mk 文件，添加

```
include device/rockchip/common/bootshutdown/bootshutdown.mk
```

动画制作需要注意以下几个问题：

- 1、图片分辨率不要超过屏幕分辨率
- 2、压缩 bootanimation.zip 文件是需要选择“存储”方式
- 3、压缩后用 winrar 打开看一下，不能有 bootanimation 这个目录

5、修改默认桌面背景

用新的桌面背景文件替换 device/rockchip/rk356x/overlay/frameworks/base/core/res/res 所有子目录内的 default_wallpaper.png 文件。

6、内置其他应用

将应用程序 APK 放到下面对应目录即可

device/rockchip/rk356x/rk3568_s/preinstall	不可卸载
device/rockchip/rk356x/rk3568_s/preinstall_del	可卸载，恢复出厂设置时会自动再次自动安装
device/rockchip/rk356x/rk3568_s/preinstall_del_forever	可卸载，不可恢复

7、开机自启动 Launcher(不显示系统桌面)

首先在开发应用 APK 时，需要在应用程序 AndroidManifest.xml 的 Intent-filter 里添加下面几行

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
```

```

        <category android:name="android.intent.category.HOME" />
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
    
```

可以参考源代码目录下的 OnlyLauncher.7z

将编译好的 Launcher APK 文件放到 device\rockchip\rk3288\preinstall 目录下

8、修改系统默认参数配置

系统参数配置文件位置 [device/rockchip/rk356x/rk3568_s/rk3568_s.mk](#)

主显示屏默认显示方向

默认旋转 0 度，其他方向注意需要同时修改下面几个参数值

参数名	旋转 0 度	旋转 90 度	旋转 180 度	旋转 270 度
ro.surface_flinger.primary_display_orientation	ORIENTATION_0	ORIENTATION_90	ORIENTATION_180	ORIENTATION_270
ro.minui.default_rotation	ROTATION_NONE	ROTATION_RIGHT	ROTATION_DOWN	ROTATION_LEFT
ro.input_flinger.primary_touch.rotation	0	90	180	270

禁止屏幕旋转

persist.sys.forced_orient

当选择禁止屏幕旋转后，如果系统默认是横屏显示，即使启动竖屏应用，屏幕显示方向也不会改变
默认值 0

可选值：0、1

第二路以太网 IP 获取方式

persist.net.eth1.mode

默认值：0

可选值：0（DHCP）、1（静态）

第二路以太网静态 IP 地址

persist.net.eth1.ip

默认值：192.168.1.159/24

格式为<IP>/24

persist.net.eth1.gateway

默认值：192.168.1.2

格式为<IP>

persist.net.eth1.dns

默认值：192.168.1.2

格式为<IP>

默认是否全屏显示（隐藏状态栏）

persist.sys.def_hidenavigation

persist.sys.def_hidestatusbar

默认值：0

可选值：0、1

显示 LCD DPI 值调整

ro.sf.lcd_density

默认值 160

可选值: 120、160、240、320

是否关闭北斗/GPS 功能

config.disable_gps

默认值: false

可选值: false、true

是否关闭蓝牙功能

config.disable_bluetooth

默认值: false

可选值: false、true

是否打开开发者选项

sys.def_develop_enable

默认值: 1

可选值: 0、1

长按电源键功能

sys.def_powerkey_long

默认值: 1

可选值: 0 (无效)、1 (显示关机菜单)、2 (直接关机需确认)、3 (直接关机无需确认)

默认 WIFI 自动连接 SSID 和密码

sys.def_wifi_ssid

默认值: PEITE-WIFI-WORK

sys.def_wifi_pass

默认值: peite-13579

默认是否打开 WIFI

sys.def_wifi_on

默认值: 1

可选值: 0、1

默认是否打开蓝牙

sys.def_bluetooth_on

默认值: 0

可选值: 0、1

自动休眠时间

sys.def_screen_off_timeout

默认值: 0

可选值: 0	永不休眠
1800000	30 分钟
600000	10 分钟
300000	5 分钟
120000	2 分钟

60000	1 分钟
30000	30 秒
15000	15 秒

默认背光亮度

sys.def_screen_brightness

默认值: 255

可选值: 0 ~ 255

系统默认音量

sys.def_volume_music=15	范围 0 ~ 15
sys.def_volume_ring=7	范围 0 ~ 7
sys.def_volume_system=7	范围 0 ~ 7
sys.def_volume_voicecall=5	范围 0 ~ 5
sys.def_volume_alarm=7	范围 0 ~ 7
sys.def_volume_notification=7	范围 0 ~ 7
sys.def_volume_bluetoothsoc=15	范围 0 ~ 15

是否禁用深度休眠

persis.sys.def_no_deepsleep

默认值: 1

可选值: 0 (启用深度休眠)、1 (禁用深度休眠)

默认 NTP 服务器地址

sys.def_ntp_server

默认值: ntp.aliyun.com

默认 NTP 超时时间

sys.def_ntp_timeout

默认值: 10000

默认界面模式

sys.def_nightmode

默认值: 2

可选值: 0 (自动模式)、1 (普通模式)、2 (黑夜暗黑模式)

二、安卓应用开发

1、GPIO 编程参考

通过 sysfs 方式控制 GPIO，GPIO 的操作接口包括 direction 和 value 等，direction 控制 GPIO 输入和输入模式，而 value 可控制 GPIO 输出或获得 GPIO 输入。

例如控制调试灯 GPIO 操作如下（串口终端命令行方式）：

```

调试灯 GPIO 设置为输出    echo out > /sys/class/gpio/gpio23/direction
调试灯 GPIO 输出高电平    echo 1 > /sys/class/gpio/gpio23/value
调试灯 GPIO 输出高低平    echo 0 > /sys/class/gpio/gpio23/value
调试灯 GPIO 设置为输入    echo in > /sys/class/gpio/gpio23/direction
    
```

读取调试灯 GPIO 输出输入电平 `cat /sys/class/gpio/gpio23/value`

当 GPIO 处于输出和输入模式时都可以读取，当设置为输入模式时读取的是 GPIO 实际电平，当设置为输出模式时读取的是设置的值（如果设置为高电平输出，外部将引脚电平拉低后，读取的值依然是 1）。应用程序控制请参考源码下的 demo 程序源码

GPIO 对应控制目录列表			
丝印	接口	脚位	目录
LED9			/sys/class/gpio/gpio23
J28		2 脚	/sys/class/gpio/gpio33
J27		1 脚 2 脚 3 脚 4 脚 5 脚 6 脚 7 脚 8 脚 9 脚 10 脚 11 脚 12 脚	/sys/class/gpio/gpio113 /sys/class/gpio/gpio58 /sys/class/gpio/gpio97 /sys/class/gpio/gpio59 /sys/class/gpio/gpio88 /sys/class/gpio/gpio60 /sys/class/gpio/gpio115 /sys/class/gpio/gpio148 /sys/class/gpio/gpio114 /sys/class/gpio/gpio95 /sys/class/gpio/gpio57 /sys/class/gpio/gpio103
4G_RST			/sys/class/gpio/gpio41
4G_PWR			/sys/class/gpio/gpio42

2、串口 UART 编程参考

J35	串口/dev/ttyS8	PH2.0 4Pin	标配	默认为 TTL 串口，与蓝牙功能复用
J37	串口/dev/ttyS3	PH2.0 4Pin	标配	默认为 RS232，可修改电阻配置为 TTL 串口
J38	串口/dev/ttyS4	PH2.0 4Pin	标配	默认为 RS232，可修改电阻配置为 TTL 串口
J34	串口/dev/ttyS7	PH2.0 4Pin	标配	TTL 串口
J36	RS485 /dev/ttyS9	PH2.0 4Pin	标配	默认为 TTL 串口，可补焊器件配置为 RS485

安卓系统串口编程请参考源码下的 demo 程序源码或以下链接：

<https://github.com/Geek8ug/Android-SerialPort>

3、WatchDog 看门狗编程参考

进入内核后默认会启动看门狗，内核崩溃等情况出现，会在 15 秒内自动复位主板。

上层应用程序打开看门狗后，内核将看门狗控制权交由上层应用程序控制，上层应用程序的喂狗间隔建议不大于 3 秒。

看门狗的使用流程为 打开看门狗→循环喂狗→停止喂狗→关闭看门狗
 喂狗之前必须先打开看门狗，关闭看门狗之前需停止喂狗操作。
 打开看门狗后如果 15 秒内没有喂狗或关闭看门狗，系统会自动复位。

命令行测试：

打开看门狗：echo 1 >/sys/class/gzpeite/user/watch_dog

喂狗：echo 2 >/sys/class/gzpeite/user/watch_dog

关闭看门狗：echo 0 >/sys/class/gzpeite/user/watch_dog

应用程序控制请参考源码下的 demo 程序源码

4、获取 root 权限

系统默认已开启 root 权限，上层应用 app 可直接获取 root 权限并进行相关操作，可以参考源码目录下的 demo 程序

5、系统签名

系统签名文件位于源代码目录下，使用对应的文件对 APK 进行签名即可。

6、动态隐藏/显示系统状态栏和导航栏

隐藏状态栏和导航栏在应用 app 里面向系统发送广播

gzpeite.intent.systemui.hidennavigation 和 gzpeite.intent.systemui.hidestatusbar

显示状态栏和导航栏在应用 app 里面向系统发送广播

gzpeite.intent.systemui.shownavigation 和 gzpeite.intent.systemui.showstatusbar

测试命令如下：

```
am broadcast -a "gzpeite.intent.systemui.hidennavigation"
am broadcast -a "gzpeite.intent.systemui.hidestatusbar"

am broadcast -a "gzpeite.intent.systemui.shownavigation"
am broadcast -a "gzpeite.intent.systemui.showstatusbar"
```

请参考源码下的 demo 程序源码

7、静默安装/卸载应用

安装 APK 时，向系统发送 gzpeite.intent.action.install_apk 广播

卸载 APK 时，向系统发送 gzpeite.intent.action.uninstall_apk 广播

测试命令如下：

```
am broadcast -a "gzpeite.intent.action.install_apk" --es apk_path "/mnt/media_rw/0000-4823/GPSTest.apk"
am broadcast -a "gzpeite.intent.action.uninstall_apk" --es pkg_name "com.android.gptest"
```

8、重启、关机操作

重启：向系统发送 gzpeite.intent.action.reboot 广播

关机向系统发送 gzpeite.intent.action.shutdown 广播

测试命令如下：

```
重启(有确认提示): am broadcast -a "gzpeite.intent.action.reboot" --ez confirm true
重启(无确认提示): am broadcast -a "gzpeite.intent.action.reboot" --ez confirm false
关机(有确认提示): am broadcast -a "gzpeite.intent.action.shutdown" --ez confirm true
关机(无确认提示): am broadcast -a "gzpeite.intent.action.shutdown" --ez confirm false
```

应用程序控制请参考源码下的 demo 程序源码

9、获取 MAC 地址

原生 Android12 系统默认禁止应用获取 MAC 地址，为了兼容更早期的应用程序，我司已对系统代码进行优化允许应用 app 获取 WIFI 及以太网的 MAC 地址，详见源代码目录下的 demo 程序源码。

三、Linux 系统开发

1、Uboot 研发

详见 芯片原厂文档\common\UBOOT 目录下相关文件

Uboot 源码位于 u-boot 目录下

2、内核研发

内核设备树文件位置：

kernel/arch/arm64/boot/dts/rockchip/rk3568.dtsi

kernel/arch/arm64/boot/dts/rockchip/rk3568-gzpeite.dtsi

默认内核配置 kernel/arch/arm64/configs/rockchip_gzpeite_linux_defconfig

```
cd kernel
make menuconfig
修改相关配置并保存，同时要手动将新的配置文件复制到 config 目录
cp -rf .config arch/arm64/configs/rockchip_gzpeite_linux_defconfig
```

其他内核相关研发请参考 芯片原厂文档 目录下的相关文档

3、修改启动 logo

用新的 bmp 文件替换 kernel 目录下的 logo.bmp 和 logo_kernel.bmp 文件，图片分辨率不要超过屏幕分辨率。

4、buildroot 及 debian11 系统修改

buildroot 及 debian 系统由瑞芯微原厂制作，相关文档已经比较完善，可以参考 芯片原厂文档\Linux 目录下的相关文档进行相关修改。

对于 buildroot 跟文件系统修改：

将需要修改的文件放到 buildroot\board\rockchip\rk3566_rk3568\fs-overlay 目录下的相应目录里面，在编译 buildroot 系统的时候会自动将文件复制到根文件系统中。

对于 debian 根文件系统的修改，可以参考下面的方式：

将需要修改的文件放到 debian\overlay 目录下的相应目录里面，在编译 debian 系统的时候会自动将文件复制到根文件系统中。

5、Ubuntu 22.04 系统修改

修改根文件系统

将需要修改的文件放到 ubuntu\overlay 目录下的相应目录里面，在编译 ubuntu 系统的时候会自动复制到根文件系统中。

修改 WIFI 连接的 SSID 及密码

ubuntu\overlay\usr\local\sbin\boot_run.sh (SSID 是 GZPEITE-WIFI, 密码是 1357924680)

```
nmcli connection add type wifi con-name "wlan0" ifname wlan0 ssid "GZPEITE-WIFI"
nmcli connection modify "wlan0" wifi-sec.key-mgmt wpa-psk
nmcli connection modify "wlan0" wifi-sec.psk "1357924680"
```


修改以太网连接参数

ubuntu\overlay\usr\local\sbin\boot_run.sh (默认是 DHCP 方式)

```
nmcli connection delete "Wired connection 1"
nmcli connection add type ethernet con-name "eth0" ifname eth0 (DHCP 方式)
# nmcli connection add type ethernet con-name "eth0" ifname eth0 ipv4.method manual
ipv4.address 192.168.1.68/24 gw4 192.168.1.1 ipv4.dns "223.5.5.5,8.8.8.8" (静态方式)
```

修改 4G 连接参数

ubuntu\overlay\usr\local\sbin\boot_run.sh, 根据需要修改相关参数即可, 可以参考下面的配置

```
# nmcli connection add type gsm con-name "China-Telecom-01" ifname ttyUSB2 gsm.apn ctnet
gsm.user ctlte@mycdma.cn gsm.password vnet.mobi gsm.number *777 gsm.network-id 46011
# nmcli connection add type gsm con-name "China-Telecom-02" ifname ttyUSB2 gsm.apn ctnet
gsm.user ctlte@mycdma.cn gsm.password vnet.mobi gsm.number *777 gsm.network-id 46012
# nmcli connection add type gsm con-name "China-Telecom-03" ifname ttyUSB2 gsm.apn ctnet
gsm.user ctlte@mycdma.cn gsm.password vnet.mobi gsm.number *777 gsm.network-id 46013
# nmcli connection add type gsm con-name "China-Telecom-04" ifname ttyUSB2 gsm.apn ctnet
gsm.user card gsm.password card gsm.number *777 gsm.network-id 46005

# nmcli connection add type gsm con-name "China-Mobile-01" ifname ttyUSB2 gsm.apn cmnet
gsm.user cmnet gsm.password cmnet gsm.number *98*1# gsm.network-id 46004

nmcli connection add type gsm con-name "China-Unicom-01" ifname ttyUSB2 gsm.apn 3gnet
gsm.user 3gnet gsm.password 3gnet gsm.number *99# gsm.network-id 46006
# nmcli connection add type gsm con-name "China-Unicom-01" ifname ttyUSB2 gsm.apn
unim2m.njm2mapn gsm.user 3gnet gsm.password 3gnet gsm.number *99# gsm.network-id 46006
# nmcli connection add type gsm con-name "China-Unicom-02" ifname ttyUSB2 gsm.apn 3gnet
gsm.user 3gnet gsm.password 3gnet gsm.number *99# gsm.network-id 46009
```

四、Ubuntu 系统 OpenGL 测试

在调试串口终端输入以下命令

```
systemctl stop gdm
Xorg &
glmark2-es2 --off-screen
```

```

root@gzpeite:~# glmark2-es2 --off-screen
arm_release_ver: g13p0-01eac0, rk_so_ver: 3
=====
glmark2 2021.12
=====
OpenGL Information
GL_VENDOR: ARM
GL_RENDERER: Mali-G52
GL_VERSION: OpenGL ES 3.2 v1.g13p0-01eac0.0fd2effaec483a5f4c440d2ffa25eb7a
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0
Surface Size: 800x600 windowed
=====
[build] use-vbo=false: FPS: 208 FrameTime: 4.808 ms
[build] use-vbo=true: FPS: 261 FrameTime: 3.831 ms
[texture] texture-filter=nearest: FPS: 265 FrameTime: 3.774 ms
[texture] texture-filter=linear: FPS: 261 FrameTime: 3.831 ms
[texture] texture-filter=mipmap: FPS: 261 FrameTime: 3.831 ms
[shading] shading=gouraud: FPS: 254 FrameTime: 3.937 ms
[shading] shading=blinn-phong-inf: FPS: 252 FrameTime: 3.968 ms
[shading] shading=phong: FPS: 238 FrameTime: 4.202 ms
[shading] shading=cel: FPS: 231 FrameTime: 4.329 ms
[bump] bump-render=high-poly: FPS: 158 FrameTime: 6.329 ms
[bump] bump-render=normals: FPS: 260 FrameTime: 3.846 ms
[bump] bump-render=height: FPS: 216 FrameTime: 4.630 ms
[effect2d] kernel=0,1,0,1,-4,1,0,1,0;: FPS: 189 FrameTime: 5.291 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 116 FrameTime: 8.621 ms
[pulsar] light=false:quads=5:texture=false: FPS: 199 FrameTime: 5.025 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 185 FrameTime: 5.405 ms
[desktop] effect-shadow-windows=4: FPS: 292 FrameTime: 3.425 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 58 FrameTime: 17.241 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=subdata: FPS: 59 FrameTime: 16.949 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 69 FrameTime: 14.493 ms
[ideas] speed=duration: FPS: 80 FrameTime: 12.500 ms
[jellyfish] <default>: FPS: 187 FrameTime: 5.348 ms
[terrain] <default>: FPS: 31 FrameTime: 32.258 ms
[shadow] <default>: FPS: 189 FrameTime: 5.291 ms
[refract] <default>: FPS: 60 FrameTime: 16.667 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 289 FrameTime: 3.460 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 257 FrameTime: 3.891 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 232 FrameTime: 4.310 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 212 FrameTime: 4.717 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 232 FrameTime: 4.310 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 273 FrameTime: 3.663 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 272 FrameTime: 3.676 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 250 FrameTime: 4.000 ms
=====
glmark2 Score: 199
=====

```

五、Ubuntu 系统 OpenCL 测试

在调试串口终端输入以下命令

```

clinfo -a
root@gzpeite:~# clinfo -a
arm_release_ver: g13p0-01eac0, rk_so_ver: 3
Number of pPlatforms 1
Platform Name ARM Platform
Platform Vendor ARM
Platform Version OpenCL 3.0 v1.g13p0-01eac0.0fd2effaec483a5f4c440d2ffa25eb7a
Platform Profile FULL_PROFILE
Platform Extensions cl_khr_global_int32_base_atomics cl_khr_global_int32_exten
ed_atomics cl_khr_local_int32_base_atomics cl_khr_local_int32_extended_atomics cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_icd cl_khr_
egl_image cl_khr_image2d_from_buffer cl_khr_depth_images cl_khr_subgroups cl_khr_subgroup_extended_types cl_k
hr_subgroup_non_uniform_vote cl_khr_subgroup_ballot cl_khr_subgroup_non_uniform_arithmetic cl_khr_subgroup_sh
uffle cl_khr_subgroup_shuffle_relative cl_khr_subgroup_clustered_reduce cl_khr_il_program cl_khr_priority_hin
ts cl_khr_create_command_queue cl_khr_spirv_no_integer_wrap_decoration cl_khr_extended_versioning cl_khr_devi
ce_uuid cl_khr_suggested_local_work_size cl_khr_extended_bit_ops cl_khr_integer_dot_product cl_khr_semaphore
cl_khr_external_semaphore cl_khr_external_semaphore_sync_fd cl_khr_command_buffer cl_arm_core_id cl_arm_print
f cl_arm_non_uniform_work_group_size cl_arm_import_memory cl_arm_import_memory_dma_buf cl_arm_import_memory_h
ost cl_arm_integer_dot_product_int8 cl_arm_job_slot_selection cl_arm_scheduling_controls cl_arm_controlled_ke
rnel_termination cl_ext_cxx_for_opengl cl_ext_image_tiling_control cl_ext_image_requirements_info cl_ext_imag
e_from_buffer
Platform Extensions with Version cl_khr_global_int32_base_atomics
0x400000 (1.0.0) cl_khr_global_int32_extended_atomics
0x400000 (1.0.0) cl_khr_local_int32_base_atomics
0x400000 (1.0.0) cl_khr_local_int32_extended_atomics
0x400000 (1.0.0)

```

六、Ubuntu 系统 OpenCV 测试

在调试串口终端输入以下命令

```

python3 -c "import cv2; print(cv2.__version__)"
root@gzpeite:~# python3 -c "import cv2; print(cv2.__version__)"
4.5.4
root@gzpeite:~# █

```

七、千兆以太网配置

开发板硬件支持千兆以太网，需要使用六类以上网线，实际使用过程中，在识别为千兆模式后，如果因网线或交换机品质实际导致通讯过程中可能会遇到丢包等情况，为了避免这种情况，开发板源码默认配置以太网为百兆模式，如果需要测试千兆模式则修改内核设备树文件：

```

&gmac0 {
    .....
    // max-speed = <1000>;
    max-speed = <100>;
    .....
};
&gmac1 {
    .....
    // max-speed = <1000>;
    max-speed = <100>;
    .....
};
    
```

八、显示屏配置

在源代码目录下有 U 盘或 TF 卡更新显示参数.7z 压缩包，里面有常见显示屏的参考配置，在调试显示屏时根据显示屏的规格书修改内核设备树文件，重新编译源码、烧写测试。

为了提高调试效率，可以先用 U 盘或 TF 更新显示屏参数的方式把显示屏调试好，然后再修改源码编译。

SDK 内的系统源代码默认配置为 HDMI 输出，购买 7 寸或 10.1 寸贴合屏的用户如果重新烧写系统固件后显示屏无显示，用压缩包内下面文件更新显示屏参数后就正常了。

rk3568-gzpeite_LVDS0_1024x600_7 寸贴合屏.dts

rk3568-gzpeite_MIPiO_800x1280_10.1 寸贴合屏.dts

九、动态修改开机 logo 和动画

将 logo.bmp、logo_kernel.bmp、bootanimation.zip，复制到系统 /mnt/logo 目录下即可

```

adb push logo.bmp /mnt/logo/
adb push logo_kernel.bmp /mnt/logo/
adb push bootanimation.zip /mnt/logo/
    
```

两个 logo 文件必须为 bmp 文件格式，一个是在 uboot 阶段加载显示，一个是在内核阶段加载显示，文件名不可修改，文件内容可以完全一样。

开机动画 bootanimation.zip 仅支持安卓系统，制作方式可以通过搜索引擎查询相关教程。

十、联系方式

总公司：广州佩特电子科技有限公司

总公司地址：广州市天河区大观中路新塘大街鑫盛工业园 A1 栋 201

总公司网站：<http://www.gzpeite.net>

SMT 子公司：广州佩特精密电子科技有限公司（全资子公司）

子公司地址：广州市白云区人和镇大巷村顺景路 11 号

SMT 网站：<http://www.gzptjm.com>

官方淘宝店：<https://shop149045251.taobao.com>

微信扫描维码联系支持人员：



广州佩特电子科技有限公司

2023 年 2 月