Security Class: Top-Secret( ) Secret( ) Internal( ) Public(√)

Rockchip Android 11.0 Ebook SDK Developer Guide

Status: [ ] Draft [√] Released [ ] Modifying	File No.:	RK-KF-YF-283
	Current Version:	V1.0.1
	Author:	Zheng Jian
	Finish Date:	2021-03-01
	Auditor:	Liu Yixing, Huang Zufang, Wu Liangqing
	Finish Date:	2021-03-01

Version no.	Author	Revision Date	Revision Description	Remark
V1.0.0	Zheng Jian	2021-01-26	Released RK3566 Android11.0 Ebook initial version	
V1.0.1	Liu Yixing	2021-03-01	Added the instruction of the virtual width and height of panel parameter	

If there is any question about the document, please email to: zj@rock-chips.com

DISCLAIMER

Trademark Statement

All rights reserved. ©2021. Rockchip Electronics Co., Ltd.

Rockchip Electronics Co., Ltd.
No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian, PRC
Website: www.rock-chips.com
Customer service Tel: +86-4007-700-590
Customer service Fax: +86-591-83951833
Customer service e-Mail: fae@rock-chips.com

Rockchip Android 11.0 Ebook SDK Chipset support

Chipset platform	Support or not	SDK version
RK3566	Support	RKR1

Rockchip Android 11.0 Ebook SDK code download and compile

Code download

Download address

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-
release/tools/repo.git -u
ssh://git@www.rockchip.com.cn:2222/Android_R/manifests_ebook.git -m
Android11_ebook.xml
```

Download server mirroring

```
repo init --repo-url=ssh://git@www.rockchip.com.cn:2222/repo-
release/tools/repo.git -u
ssh://git@www.rockchip.com.cn:2222/Android_R/manifests_ebook.git -m
Android11_ebook.xml --mirror
```

Note: repo is a script invoking git developed by Google using Python script, and mainly used to download, manage Android project software lib. The download address is as follows:

```
git clone ssh://git@www.rockchip.com.cn:2222/repo-release/tools/repo
```

Generally, Rockchip FAE contact will provide the initial compressed package of the corresponding version SDK in order to help customers acquire SDK source code quickly. Take `ROCKCHIP_ANDROID11.0_EBOOK_SDK_RELEASE.tar.gz.*` as an example, you can sync the source code through the following command after getting the initial package:

```
mkdir ROCKCHIP_ANDROID11.0_EBOOK_SDK_RELEASE
cat ROCKCHIP_ANDROID11.0_EBOOK_SDK_RELEASE.tar.gz* | tar -zx -C
ROCKCHIP_ANDROID11.0_EBOOK_SDK_RELEASE
cd ROCKCHIP_ANDROID11.0_EBOOK_SDK_RELEASE
.repo/repo/repo sync -l
.repo/repo/repo sync -c
```

Code compiling

One key compiling command

```
./build.sh -UKAup
（ WHERE: -U = build uboot
       -C = build kernel with Clang
       -K = build kernel
       -A = build android
       -p = will build packaging in IMAGE
       -o = build OTA package
       -u = build update.img
       -v = build android with 'user' or 'userdebug'
       -d = huild kernel dts name
       -V = build version
       -J = build jobs
   ------------you can use according to the requirement, no need to record
uboot/kernel compiling commands------------------
 ）

============================================================
Please remember to set the environment variable before using the one key
compiling command, and select the platform to be compiled, for example:
source build/envsetup.sh
lunch rk3566_eink-userdebug
============================================================
```

Compiling command summary

Soc	type	model	Android	one key compiling	kernel compiling
RK3566	10.3" tablet	device	build/envsetup.sh;lunch rk3566_eink-userdebug	./build.sh -AUCKu -d rk3566-rk817-eink-w103	make ARCH=arm64 rockchip_defconfig rk356x.config android-11-go.config rk356x_eink.config && make ARCH=arm64 rk3566-rk817-eink-w103.img
RK3566	6" tablet	device	build/envsetup.sh;lunch rk3566_einkw6-userdebug	./build.sh -AUCKu -d rk3566-rk817-eink-w6	make ARCH=arm64 rockchip_defconfig rk356x.config android-11-go.config rk356x_eink.config && make ARCH=arm64 rk3566-rk817-eink-w6.img

Other compiling instruction

Android11.0 cannot directly flash kernel.img and resource.img

Android11.0 kernel.img and resource.img are included in boot.img. After updating and compiling kernel, need to execute ./mkimage.sh in android root directory to re-package boot.img, and then flash boot.img under rockdev. You can use the following method to compile kernel only.

Only compile kernel to generate boot.img

The compiling principle: in kernel directory, replace the old `boot.img` with the newly compiled `kernel.img` and `resource.img`. So when compiling it requires to use `BOOT_IMG=xxx` parameter to specify the path of boot_sample.img. the command is as follow:

Take `RK3566 w103` device as example, replace the corresponding boot.img and dts when compiling:

```
cd kernel
make ARCH=arm64 rockchip_defconfig rk356x.config android-11-go.config
rk356x_eink.config
make ARCH=arm64 BOOT_IMG=boot_sample.img rk3566-rk817-eink-w103.img -j24
```

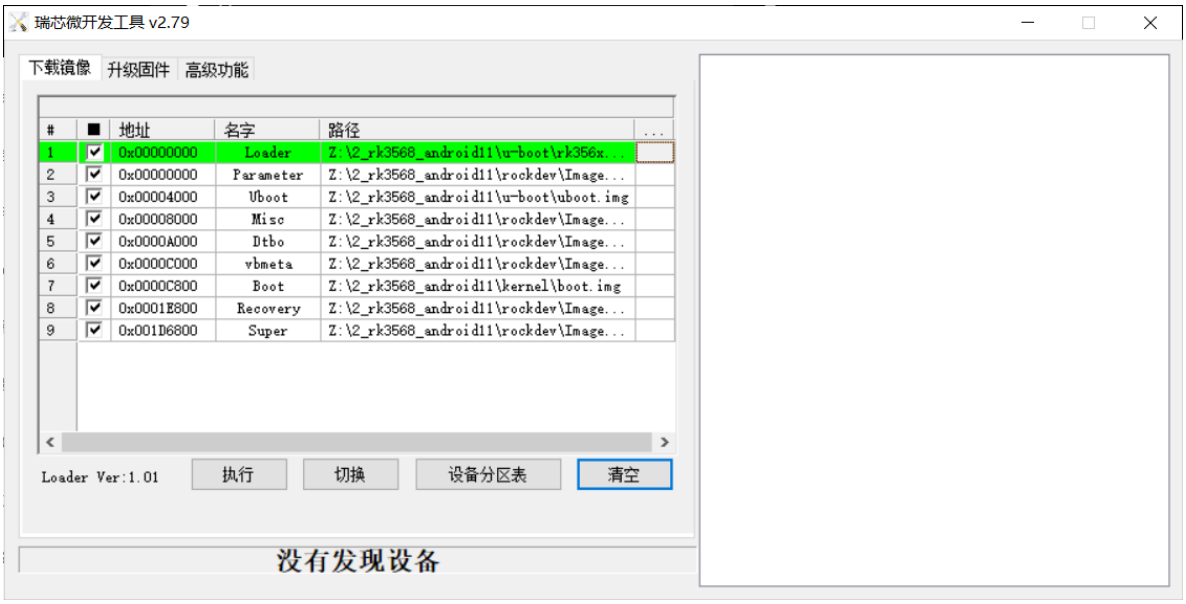
After compiling, you can directly flash the boot.img under kernel directory to the boot location of the device. **Please firstly load the partition table（parameter.txt）** before flashing, to prevent from flashing to the wrong place.

Image flashing

Image flashing tool

Android11 requires to update the USB driver DriverAssitant to V5.1.1 version. You can refer to the tool instruction chapter to do the upgrade.
Windows flashing tool:

```
RKTools/windows/AndroidTool/AndroidTool_Release_v2.79
```




```
RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.56
```

There are more details in the tool instruction chapter.

Image instruction

After complete compiling, it will generate the following files:（take RK3566 as example, here lunch rk3566_eink-userdebug)

```
rockdev/Image-rk3566_eink/
```

Just use the tool to flash the following files:(no need to flash trust.img for RK3566/RK3568)

```
rockdev/Image-rk3566_eink
├── boot-debug.img
├── boot.img
├── config.cfg
├── dtbo.img
├── logo.img
├── MiniLoaderAll.bin
├── misc.img
├── parameter.txt
├── pcba_small_misc.img
├── pcba_whole_misc.img
├── recovery.img
├── resource.img
├── super.img
├── uboot.img
├── vbmeta.img
└── waveform.img
```

or you can directly flash `update.img`

Image instruction

Image	Instruction
boot.img	including ramdis、kernel、dtb
boot-debug.img	the difference between boot.img and boot-debug.img is that user image can flash this boot.img to do root operation
dtbo.img	Device Tree Overlays refer to dtbo chapter instruction later
logo.img	including ebook platform uboot bootup logo, kernel bootup logo, uboot charging logo and so on
config.cfg	the configuration file of the flash tool, you can directly load the options required to be flashed for the flash tool
MiniLoaderAll.bin	including first level loader
misc.img	including recovery-wipe boot symbol information, after flashing it will enter recovery
parameter.txt	including partition information
pcba_small_misc.img	including pcba boot symbol information, after flashing it will enter the simple pcba mode
pcba_whole_misc.img	including pcba boot symbol information, after flashing it will enter the complete pcba mode
recovery.img	including recovery-ramdis、kernel、dtb
super.img	including the contents of odm, product, vendor, system, system_ext partitions
trust.img	including BL31、BL32 which are not generated for RK3566/RK3568, no need to flash
uboot.img	including uboot image
vbmeta.img	including avb verification information, used for AVB verification
update.img	including the above img files to be flashed, can be used for the tool to directly flash the whole image package
waveform.img	waveform file

Use fastboot to flash dynamic partition

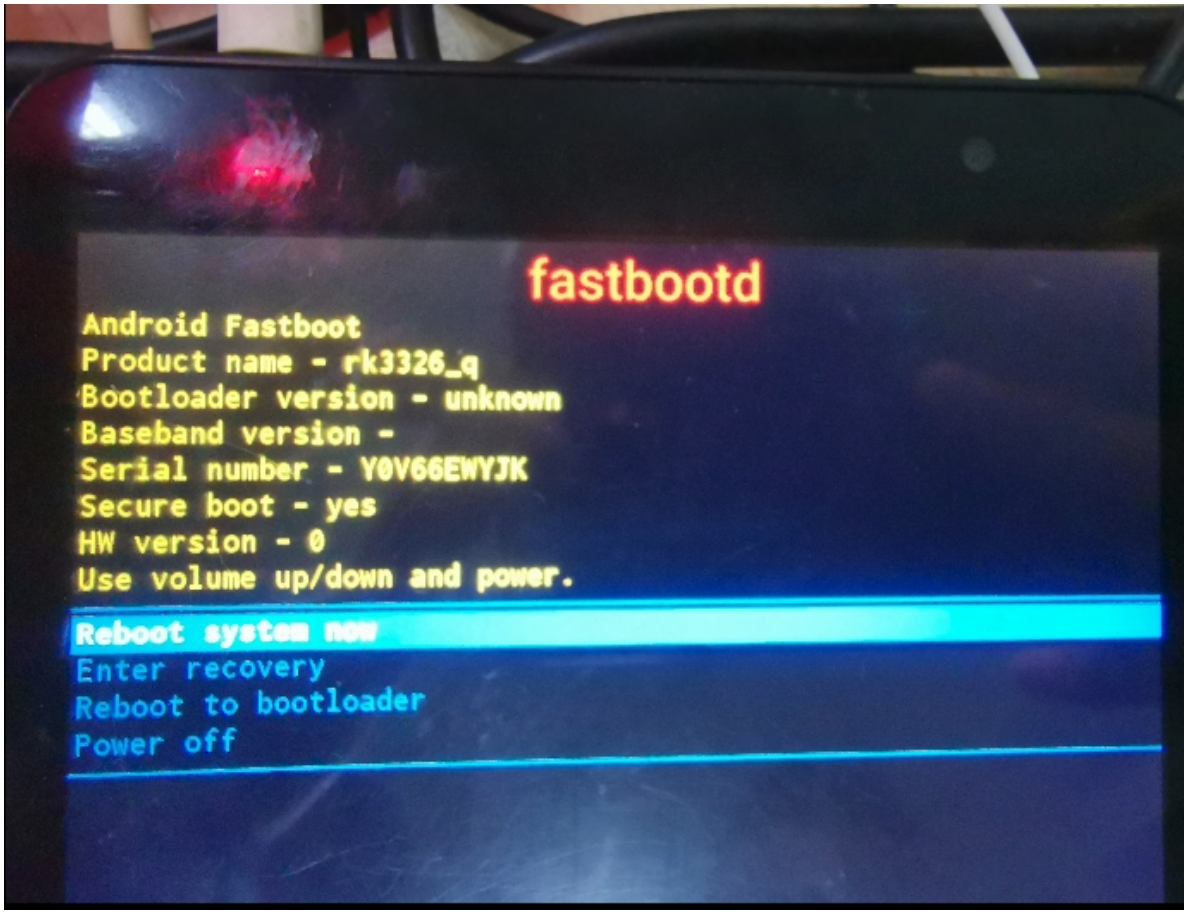
The new device with R supports dynamic partition, and already removes system/vendor/odm/product/system_ext partitions. Please flash super.img. Use `fastbootd` can flash system/vendor/odm etc. alone (the corresponding img files can be found under out). The version of adb and fastboot should be the latest. SDK provides the compiled tool package:

```
RKTools/linux/Linux_adb_fastboot (Linux_x86 version)
RKTools/windows/adb_fastboot (Windows_x86 version)
```

- Use the command to flash dynamic partition:

```
adb reboot fastboot
fastboot flash vendor vendor.img
fastboot flash system system.img
fastboot flash odm odm.img
```

Note: After entering fastbootd mode, relative information of the device will be displayed on the screen, as shown below:



Note: when non-dynamic partition needs to use fastboot, please enter bootloader:

```
adb reboot bootloader
```

How to use DTBO function

Android 10.0 and above versions support Device Tree Overlays function, which requires to flash dtbo.img during development, and is compatible with multiple products.
The modification method:

1. Find (or specify) the template file:

```
get_build_var PRODUCT_DTBO_TEMPLATE
```

For example:

```
PRODUCT_DTBO_TEMPLATE := $(LOCAL_PATH)/dt-
overlay.in(device/rockchip/rk356x/rk3566_r/dt-overlay.in)
```

1. Add or modify the required node:
   For example:

```
/dts-v1/;
/plugin/;

&chosen {
    bootargs_ext = "androidboot.boot_devices=${_boot_device}";
};

&firmware_android {
    vbmeta {
        status = "disabled";
    };
    fstab {
        status = "disabled";
    };
};

&reboot_mode {
    mode-bootloader = <0x5242C309>;
    mode-charge = <0x5242C30B>;
    mode-fastboot = <0x5242C303>;
    mode-loader = <0x5242C301>;
    mode-normal = <0x5242C300>;
    mode-recovery = <0x5242C303>;
};
```

Note: There must be alias in the dts when using dtbo, otherwise it cannot overlay successfully.

Modify fstab file

1. Fine (or specify) the template file:

```
get_build_var PRODUCT_FSTAB_TEMPLATE
```

For example:

```
PRODUCT_FSTAB_TEMPLATE := device/rockchip/rk356x/rk3566_eink/fstab_eink.in
```

1. Modify: add partition mounting, modify swap_zram parameter, modify data partition format and so on

Modify parameter.txt

Android 11 adds the tool that can generate parameter.txt, and support to compile parameter.txt according to the configuration parameters. If there is no configuration template file, it will find and add the modified parameter.txt file.

1. Find (or specify) the template file:

```
get_build_var PRODUCT_PARAMETER_TEMPLATE
```

For example:

```
PRODUCT_PARAMETER_TEMPLATE :=
device/rockchip/common/scripts/parameter_tools/parameter.in
```

1. Modify the partition size configuration(example as below):

```
BOARD_SUPER_PARTITION_SIZE := 2688548864
BOARD_DTBOIMG_PARTITION_SIZE := xxxx
BOARD_BOOTIMAGE_PARTITION_SIZE := xxxxx
BOARD_CACHEIMAGE_PARTITION_SIZE := xxxx
```

1. Not to use parameter generation tool:
   Just add a parameter.txt file to your device directory:
   For example:

```
device/rockchip/rk356x/rk3566_eink/parameter.txt
```

1. Only use the tool to generate parameter.txt(example as below):

```
parameter_tools --input
device/rockchip/common/scripts/parameter_tools/parameter.in --firmware-version
11.0 --machine-model rk3566_eink --manufacturer rockchip --machine rk3566_eink -
-partition-list
uboot_a:4096K,trust_a:4M,misc:4M,dtbo_a:4M,vbmeta_a:4M,boot_a:33554432,backup:30
0M,security:4M,cache:300M,metadata:4096,frp:512K,super:2G --output
parameter_new.txt
```

Note: If need to do the big version upgrade through OTA, please directly use the previous version's parameter.txt

Android common configuration

Create product lunch

Take RK356x platform as example to create a new rk3566_einkw6 product. The steps are as below:
1.Modify device/rockchip/rk356x/AndroidProducts.mk to add rk3566_einkw6 lunch

```
diff --git a/AndroidProducts.mk b/AndroidProducts.mk
index 4f60ff8..8b4def7 100644
--- a/AndroidProducts.mk
+++ b/AndroidProducts.mk
@@ -18,7 +18,8 @@ PRODUCT_MAKEFILES := \
         $(LOCAL_DIR)/rk3566_rgo/rk3566_rgo.mk \
         $(LOCAL_DIR)/rk3566_r/rk3566_r.mk \
         $(LOCAL_DIR)/rk3568_r/rk3568_r.mk \
-        $(LOCAL_DIR)/rk3566_eink/rk3566_eink.mk
+        $(LOCAL_DIR)/rk3566_eink/rk3566_eink.mk \
+       $(LOCAL_DIR)/rk3566_einkw6/rk3566_einkw6.mk

 COMMON_LUNCH_CHOICES := \
     rk3566_rgo-userdebug \
@@ -28,6 +29,8 @@ COMMON_LUNCH_CHOICES := \
```

```
    rk3568_r-userdebug \
    rk3568_r-user \
    rk3566_eink-userdebug \
-   rk3566_eink-user
+   rk3566_eink-user \
+   rk3566_einkw6-userdebug \
+   rk3566_einkw6-user
```

2.Create rk3566_einkw6 directory under device/rockchip/rk356x
Create referring to the existing rk3566_eink product directory in device/rockchip/rk356x. You can directly copy rk3566_eink to rk3566_einkw6, and then replace all the `rk3566_eink` under rk3566_einkw6 directory with `rk3566_einkw6`

Ebook Common Configuration

Reference dts

kernel/arch/arm64/boot/dts/rockchip/rk3566-rk817-eink-w103.dts
kernel/arch/arm64/boot/dts/rockchip/rk3566-rk817-eink-w6.dts

Bootup, Charging, Standby, Power off pictures

logo supports uboot logo, kernel logo and charging animation
uboot and kernel logo use bmp pictures, and the resolution of bmp picture are required t be the same as that of panel, otherwise it will not display. When the system is executing ./mkimage.sh, it will convert these bmp images to grey images. If customers need to replace the corresponding logo, please update the image in the following source code:
device/rockchip/rk356x/rk3566_eink/eink_logo

Note: The charging animation lasts for a while and then enter standby, but the screen still displays the image with current battery level and it will update the corresponding image when the battery level is updated. For example, current battery level is 25%, after the charging animation lasts for a period, it will enter standby, and the interface will display an icon with one grid of battery level, when the battery level reaches 40% by charging, the icon will be updated as two grids. Assuming that user short presses the power button now, it will switch from standby mode to charging animation.

The corresponding definitons of the images are as follows:

```
├── android_logo
|   └── bootanimation.zip //Android bootup animation
├── kernel_logo
|   └── kernel.bmp        //kernel logo
├── poweroff_logo
|   ├── poweroff_nopower.png    //power off image with low power
|   └── poweroff.png            //power off image
├── standby_logo
|   ├── standby_charge.png      //image of charging in standby
|   ├── standby_lowpower.png    //image of standby with low power
|   └── standby.png             //standby image
└── uboot_logo
    ├── battery_0.bmp       //battery level is more than %0 but less than 20%
    ├── battery_1.bmp       //battery level is more than 20% but less than 40%
    ├── battery_2.bmp       //battery level is more than 40% but less than 60%
```

```
    ├── battery_3.bmp       //battery level is more than 60% but less than %80
    ├── battery_4.bmp       //battery level is more than 80% but less than 100%
    ├── battery_5.bmp       //battery level is 100%
    ├── battery_fail.bmp    //uboot power is too low, failing to enter the
system
    └── uboot.bmp           //uboot logo
```

Above standby, power off and other images of Android part will finally be put in the path of /vendor/media/ of the board system.
Above images prefer to use the images in data partition (can be used to customize to display standby, power off image). If user images do not exist, it will use system images. All the images use png format, and the resolution should be the same as the resolution of the panel, otherwise it cannot display normally.

```
user power off image "/data/misc/poweroff.png"
user power off image with low power "/data/misc/poweroff_nopower.png"
user standby image "/data/misc/standby.png"
user standby image with low power "/data/misc/standby_lowpower.png"
user charging image in standby "/data/misc/standby_charge.png"
system power off image "/vendor/media/poweroff.png"
system power off image with low power "/vendor/media/poweroff_nopower.png"
system standby image "/vendor/media/standby.png"
system standby image with low power "/vendor/media/standby_lowpower.png"
system charging image in standby "/vendor/media/standby_charge.png"
```

kernel low battery pre-charging and low battery display instruction

dts configuration information is as follows:

```
charge-animation {
    compatible = "rockchip,uboot-charge";
    status = "okay";
    rockchip,uboot-charge-on = <1>;             // whether to enable U-Boot
charging
    rockchip,android-charge-on = <0>;           // ebook doesn't support android
charging, it must be configured as 0
    rockchip,uboot-exit-charge-level = <5>;     // when U-Boot is charging, the
lowest battery level that is allowed to bootup
    rockchip,uboot-exit-charge-voltage = <3650>;// when U-Boot is charging, the
lowest voltage that is allowed to bootup
    rockchip,screen-on-voltage = <3400>;        // when U-Boot is charging, the
lowest voltage that are allowed to light up the screen
    rockchip,uboot-low-power-voltage = <3350>;  // the lowest voltage that will
force U-Boot to enter charging mode unconditionally
    rockchip,system-suspend = <1>;              // whether to enter trust
standby with low power consumption when the screen is off (ATF support is
required)
    rockchip,auto-off-screen-interval = <20>;   // interval to turn off the
screen automatically, with unit second, default is 15s
    rockchip,auto-wakeup-interval = <10>;       // interval to wake up from
standby automatically, with unit second. If the value is 0 or null, auto wake up
from standby will be disabled, which is generally used for stress test.
    rockchip,auto-wakeup-screen-invert = <1>;   // whether to turn on/off the
screen when wake up from standby automatically
```

```
};
```

Panel parameter configuration and instruction

Ink panel configuration includes two parts, one is panel parameter and the other is ebc clk:
1.Panel parameter configuration
The panel parameters refer to datasheet of ink panel as below:

Table 1 Timing Parameters Table

Mode	3	Resolution 1872x1404				
SDCK [MHz]	33.333333					
Pixels Per SDCK	8					
Line	LSL	LBL	LDL	LEL	GDCK_STA	LGONL
Parameters[SDCK]	11	8	234	23	10	215
Line	-	-	-	-	-	-
Parameters[us]	0.33	0.24	7.02	0.69	0.3	6.45
Frame	FSL	FBL	FDL	FEL	-	FR [Hz]
Parameters [lines]	1	4	1404	12	-	84.99
Frame	-	-	-	-	-	-
Parameters [us]	8.28	33.12	11625.12	99.36	-	-

The panel parameter configurations are in dts file, e.g. panel parameter configuration of ES103TC1 is as follows:
panel,mirror means whether to mirror for the display data, some panel should be configured as 1 due to different scan direction.
panel,width-mm means the length of the screen, with unit mm, and the specific value can be obtained from the width and length of Active Area in panel datasheet.
panel,height-mm means the width of the screen, with unit mm, and the specific value can be obtained from the width and length of Active Area in panel datasheet.

```
        /* ES103TC1 */
        panel,width = <1872>; //panel resolution
        panel,height = <1404>;
        panel,vir_width = <1872>; //virtual width requires to be aligned with 16
pixel, and should be bigger than actual width
        panel,vir_height = <1404>; //virtual height shold be configured
according to actual height, and cannot be less than actual height
        panel,sdck = <33300000>;
        panel,lsl = <18>;
        panel,lbl = <17>;
        panel,ldl = <234>;
        panel,lel = <7>;
        panel,gdck-sta = <34>;
        panel,lgonl = <192>;
        panel,fsl = <1>;
        panel,fbl = <4>;
        panel,fdl = <1404>;
        panel,fel = <12>;
        panel,mirror = <0>;
        panel,panel_16bit = <1>; //configure 0 for 8bit panel and 1 for 16bit
panel
        panel,panel_color = <0>; //color panel symbol, 0,1,2...can be used to
specify different color panel later
        panel,width-mm = <157>;
        panel,height-mm = <210>;
```

2.ebc clk configuration instruction:
Different panels only need to calculate clk rates of ebc according to sdclk value of the panel and fill in ebc node.
The computing formula of 16bit panel is ebc_dclk = sdclk * 8;
The computing formula of 8bit panel is ebc_dclk = sdclk * 4;
Take ES103TC1 as example, ebc_dclk = 33300000 * 8 = 266400000;
The actual requirement is 266.4M, limited by its parent clock 1000M, actually it can only be configured with 250M at most, dts configuration is as below:

```
/* ES103TC1 */
&ebc {
    /* clock rate 1000M/n, (n=1~32) */
    assigned-clocks = <&cru CPLL_333M>, <&cru DCLK_EBC>;
    assigned-clock-rates = <250000000>, <250000000>;
    status = "okay";
};
```

Vcom voltage read/write

Vcom voltage read/write can be set through the tool RKTools/windows/RKDevInfoWriteTool_Setup_V1.1.0-ebook_210109.rar. For more details, please refer to the tool instruction. It also can read/write through kernel node, and the node path is /sys/devices/platform/ebc-dev/pmic_vcom. Check current vcom voltage cat /sys/devices/platform/ebc-dev/pmic_vcom, write vcom voltage value echo 1560 > /sys/devices/platform/ebc-dev/pmic_vcom, vcom voltage is negative, so 1560 means -1.56V.

The latest code also supports to flash Vcom voltage through SN tool, which is convenient for factory production.
The tool path: RKTools/windows/FactoryTool-ebook-1.70.00.rar

Waveform file flashing

The waveform file can be flashed directly using the image flashing tool RKTools/windows/AndroidTool/RKDevTool_Release_v2.8.zip.
cat /sys/devices/platform/ebc-dev/waveform_version node can check the version information of currently used waveform file.

The latest code can support to flash MP image and waveform file at the same time. Customers can decide whether to use or not according to actual situation.
The tool path: RKTools/windows/windows/FactoryTool-ebook-1.70.00.rar

RK customized mode instruction

Waveform mode

Table 1 Waveform Mode Summary

Mode	Supported pixel state transitions	Ghosting	Usage	Typical update time at 25 C 85 Hz (ms)
INIT	[0 1 2 3 ... 31]→ 30	N/A	Display initialization	2000
DU	[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]→ [0 30]	Low	Monochrome menu, text input, and touch screen/pen input	260
GC16	[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]→ [0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30]	Very Low	High quality images	450
GL16	[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]→ [0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30]	Medium	Text with white background	450
GLR16	[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]→ [0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]	Low	Text with white background	450
GLD16	[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]→ [0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31]	Low	Text and graphics with white background	450
A2	[0 29 30 31]→ [0 30]	Medium	Fast page flipping at reduced contrast	120
DU4	[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 29 30 31] → [0 10 20 30]	Medium	Anti-aliased text in menus / touch and screen/pen input	290

RK customized mode

The following modes are all RK customized modes, combining software algorithm and waveform data to implement different refresh methods to meet the requirements of different scenarios.

RK customized mode introduction

EPD mode usage introduction

Mode definition

```
    public static final String EPD_NULL ="-1";//invaid mode
    public static final String EPD_AUTO ="0";//auto mode. generally used in the
scenarios which require fast response such as keyboard, combining two waveform
data of GARY16 and DU modes, and it will automatically select GARY16 or DU
waveform data according to the grey value of the pixels. This mode implements
continuous multi-region refresh at the same time through algorithm, which has
the fastest refresh speed and supports 16 grey refresh.
    public static final String EPD_OVERLAY ="1";//handwriting mode. used for
handwriting mode
    public static final String EPD_FULL_GC16 ="2";//full refresh mdoe,
completely use GARY16 waveform data to refresh the screen. This mode has the
slowest refresh speed, with less residue but it will flicker. can be used to
periodically clear the residue caused by the refresh of other modes.
    public static final String EPD_FULL_GL16 ="3";
    public static final String EPD_FULL_GLR16 ="4";
    public static final String EPD_FULL_GLD16 ="5";
    public static final String EPD_FULL_GCC16 ="6";
```

```
    public static final String EPD_PART_GC16 ="7";//normal mode. part refresh
mode, comparing with the data of before and after frames, the changed pixels use
GRAY16 waveform data, and the waveform data of the unchanged pixels are 0.
support 16 grey refresh, comparing with EPD_FULL mode, there is no feeling of
flicker during refresh.
    public static final String EPD_PART_GL16 ="8";
    public static final String EPD_PART_GLR16 ="9";
    public static final String EPD_PART_GLD16 ="10";
    public static final String EPD_PART_GCC16 ="11";
    public static final String EPD_A2 ="12";//A2 mode, generally used for
reading mode. Comparing the data of before and after frames, the changed pixels
use A2 waveform data, and the waveform data of the unchanged pixels are 0. Only
support the refresh of balck and white. When the application selects this mode,
the system will automatically refresh with DU mdoe before refresh with A2 mode,
in order to adapt with the change from 16 grey scale to 2 grey scale. Besides,
this mode has dither algorithm. This mode has fast refresh speed, and can be
used in video, picture and other scenarios.
    public static final String EPD_DU ="13";//black and white mode. Comparing
with the data of before and after frames, the changed pixels use DU waveform
data, and the waveform data of the unchanged pixels are 0. This mode has fast
refresh speed, but only supports 16 grey scale from white to black.
```

Mode setting method

```
import android.os.EinkManager;

public static EinkManager mEinkManager;
public static String mEinkMode;

if(mEinkManager == null){
    //Get Eink service
    mEinkManager = (EinkManager)
mContext.getSystemService(Context.EINK_SERVICE);
}
//Get current mode
mEinkMode = mEinkManager.getMode();
//Set as EPD_A2 mode
mEinkManager.setMode(EinkManager.EinkMode.EPD_A2);
//Set as EPD_PART_GC16 mode
mEinkManager.setMode(EinkManager.EinkMode.EPD_PART_GC16);
//Set as EPD_AUTO mode
mEinkManager.setMode(EinkManager.EinkMode.EPD_AUTO);
....

//full screen refresh of one frame, note, single frame refresh has different
interface from other mode setting, after setting need to invoke postInvalidate
to send the frame
mEinkManager.sendOneFullFrame();
mView.postInvalidate();
```

【Note that full screen refresh of one frame has separate interface, generally used for screen refresh, current mode will not be changed, need to cooperate with UI to invoke postInvalidate to send the frame to refresh】
mEinkManager.sendOneFullFrame();
mView.postInvalidate();

EPD related system attribute

General system attribute instruction

1）sys.power.status：1 means low battery, 0 means normal battery
sys.power.connected 1 means charging, 0 means non-charging
2）persist.vendor.fullmode_cnt：
FULL mode count, when the time of the system using EPD_PART and other mode to refresh reaches the set value of persist.vendor.fullmode_cnt, it will automatically use EPD_FULL to refresh once to clear the residue.
3）ro.need.white.with.standby：
When the value is y, it means to refresh one white frame, and then display logo. When the value is n, it means to directly display logo. logo images include various images of standby and power off.

Advanced system attribute instruction

1）sys.eink.mode：
When the value is the value of EPD_MODE, this attribute works and the mode keeps unchanged.
2）debug.dump:
When the value is 1, hwc will dump out display buf that is transferred to kernel driver. It will automatically close after dump20 frames, and save in the directory of /data/dump. You can use 7yuv tool to parse and check buf content.

Document instruction

Peripheral support list

DDR/EMMC/NAND FLASH/WIFI/3G/CAMERA support lists keep updating in redmine, through the following link:

```
https://redmine.rockchip.com.cn/projects/fae/documents
```

Android document

```
RKDocs\android
```

Android_SELinux(Sepolicy) developer guide

```
RKDocs/android/Rockchip_Developer_Guide_Android_SELinux(Sepolicy)_CN.pdf
```

Wi-Fi document

```
RKDocs/android/wifi/
├── Rockchip_Introduction_Android10.0_WIFI_Configuration_CN&EN.pdf
└── Rockchip_Introduction_REALTEK_WIFI_Driver_Porting_CN&EN.pdf
```

3G/4G module instruction document

```
RKDocs/common/mobile-net/
├── Rockchip_Introduction_3G_Data_Card_USB_File_Conversion_CN.pdf
├── Rockchip_Introduction_3G_Dongle_Configuration_CN.pdf
└── Rockchip_Introduction_4G_Module_Configuration_CN&EN.pdf
```

Kernel document

```
RKDocs\common
```

DDR related document

```
RKDocs/common/DDR/
├── Rockchip-Developer-Guide-DDR-CN.pdf
├── Rockchip-Developer-Guide-DDR-EN.pdf
├── Rockchip-Developer-Guide-DDR-Problem-Solution-CN.pdf
├── Rockchip-Developer-Guide-DDR-Problem-Solution-EN.pdf
└── Rockchip-Developer-Guide-DDR-Verification-Process-CN.pdf
```

Audio module document

```
RKDocs/common/Audio/
├──
Rockchip_Developer_Guide_Audio_Call_3A_Algorithm_Integration_and_Parameter_Debug
ging_CN.pdf
├── Rockchip_Developer_Guide_Linux4.4_Audio_CN.pdf
└── Rockchip_Developer_Guide_RK817_RK809_Codec_CN.pdf
```

CRU module document

```
RKDocs/common/CRU/
├── Rockchip_Developer_Guide_Linux3.10_Clock_CN.pdf
└── Rockchip_RK3399_Developer_Guide_Linux4.4_Clock_CN.pdf
```

GMAC module document

```
RKDocs/common/GMAC/
└── Rockchip_Developer_Guide_Ethernet_CN.pdf
```

PCie module document

```
RKDocs/common/PCie/
└── Rockchip-Developer-Guide-linux4.4-PCIe.pdf
```

I2C module document

```
RKDocs/common/I2C/
└── Rockchip_Developer_Guide_I2C_CN.pdf
```

PIN-Ctrl GPIO module document

```
RKDocs/common/PIN-Ctrl/
└── Rockchip-Developer-Guide-Linux-Pin-Ctrl-CN.pdf
```

SPI module document

```
RKDocs/common/SPI/
└── Rockchip-Developer-Guide-linux4.4-SPI.pdf
```

Sensor module document

```
RKDocs/common/Sensors/
└── Rockchip_Developer_Guide_Sensors_CN.pdf
```

IO-Domain module document

```
RKDocs/common/IO-Domain/
└── Rockchip_Developer_Guide_Linux_IO_DOMAIN_CN.pdf
```

Leds module document

```
RKDocs/common/Leds/
└── Rockchip_Introduction_Leds_GPIO_Configuration_for_Linux4.4_CN.pdf
```

Thermal control module document

```
RKDocs/common/Thermal/
├── Rockchip-Developer-Guide-Linux4.4-Thermal-CN.pdf
└── Rockchip-Developer-Guide-Linux4.4-Thermal-EN.pdf
```

PMIC power management module document

```
RKDocs/common/PMIC/
├── Archive.zip
├── Rockchip_Developer_Guide_Power_Discrete_DCDC_EN.pdf
├── Rockchip-Developer-Guide-Power-Discrete-DCDC-Linux4.4.pdf
├── Rockchip-Developer-Guide-RK805.pdf
├── Rockchip_Developer_Guide_RK817_RK809_Fuel_Gauge_CN.pdf
├── Rockchip_RK805_Developer_Guide_CN.pdf
└── Rockchip_RK818_RK816_Introduction_Fuel_Gauge_Log_CN.pdf
```

MCU module document

```
RKDocs/common/MCU/
└── Rockchip_Developer_Guide_MCU_EN.pdf
```

Power consumption and sleep module document

```
RKDocs/common/power/
├── Rockchip_Developer_Guide_Power_Analysis_EN.pdf
└── Rockchip_Developer_Guide_Sleep_and_Resume_CN.pdf
```

UART module document

```
RKDocs/common/UART/
├── Rockchip-Developer-Guide-linux4.4-UART.pdf
└── Rockchip-Developer-Guide-RT-Thread-UART.pdf
```

DVFS CPU/GPU/DDR frequency scaling related document

```
RKDocs/common/DVFS/
├── Rockchip_Developer_Guide_CPUFreq_CN.pdf
├── Rockchip_Developer_Guide_CPUFreq_EN.pdf
├── Rockchip_Developer_Guide_Devfreq_CN.pdf
├── Rockchip_Developer_Guide_Linux4.4_CPUFreq_CN.pdf
└── Rockchip_Developer_Guide_Linux4.4_Devfreq_CN.pdf
```

EMMC/SDMMC/SDIO module document

```
RKDocs/common/MMC
└── Rockchip-Developer-Guide-linux4.4-SDMMC-SDIO-eMMC.pdf
```

PWM module document

```
RKDocs/common/PWM/
├── Rockchip-Developer-Guide-Linux-PWM-CN.pdf
└── Rockchip_Developer_Guide_PWM_IR_CN.pdf
```

USB module document

```
RKDocs/common/usb/
├── putty20190213_162833_1.log
├── Rockchip-Developer-Guide-Linux4.4-RK3399-USB-DTS-CN.pdf
├── Rockchip-Developer-Guide-Linux4.4-USB-CN.pdf
├── Rockchip-Developer-Guide-Linux4.4-USB-FFS-Test-Demo-CN.pdf
├── Rockchip-Developer-Guide-Linux4.4-USB-Gadget-UAC-CN.pdf
├── Rockchip-Developer-Guide-USB-Initialization-Log-Analysis-CN.pdf
├── Rockchip-Developer-Guide-USB-Performance-Analysis-CN.pdf
├── Rockchip-Developer-Guide-USB-PHY-CN.pdf
└── Rockchip-Developer-Guide-USB-SQ-Test-CN.pdf
```

HDMI-IN function document

```
RKDocs/common/hdmi-in/
└── Rockchip_Developer_Guide_HDMI_IN_CN.pdf
```

Security module document

```
RKDocs/common/security/
├── Efuse process explain .pdf
├── RK3399_Efuse_Operation_Instructions_V1.00_20190214_EN.pdf
├── Rockchip_Developer_Guide_Secure_Boot_V1.1_20190603_CN.pdf
├── Rockchip_Developer_Guide_TEE_Secure_SDK_CN.pdf
├── Rockchip_RK3399_Introduction_Efuse_Operation_EN.pdf
├── Rockchip-Secure-Boot2.0.pdf
├── Rockchip-Secure-Boot-Application-Note-V1.9.pdf
└── Rockchip Vendor Storage Application Note.pdf
```

uboot introduction document

```
RKDocs\common\u-boot\Rockchip-Developer-Guide-UBoot-nextdev-CN.pdf
```

Trust introduction document

```
RKDocs/common/TRUST/
├── Rockchip_Developer_Guide_Trust_CN.pdf
└── Rockchip_Developer_Guide_Trust_EN.pdf
```

Camera document

```
RKDocs\common\camera\HAL3\
```

Tool document

```
RKDocs\common\RKTools manuals
```

PCBA development and usage document

```
RKDocs\android\Rockchip_Developer_Guide_PCBA_Test_Tool_CN&EN.pdf
```

Panel driver debugging guide

```
RKDocs\common\display\Rockchip_Developer_Guide_DRM_Panel_Porting_CN.pdf
```

HDMI debugging guide

```
RKDocs\common\display\Rockchip_Developer_Guide_HDMI_Based_on_DRM_Framework_CN.pdf
```

Graphic display DRM Hardware Composer（HWC) issue analyzing

```
RKDocs\common\display\Rockchip FAQ DRM Hardware Composer V1.00-20181213.pdf
```

DRM display developer guide

```
RKDocs\common\display\Rockchip DRM Display Driver Development Guide V1.0.pdf
```

RGA related issues analyzing

```
RKDocs\common\display\Rockchip_RGA_FAQ.pdf
```

Graphic display framework common issue analysis

include frameworks、GPU.Gralloc、GUI、HWComposer、HWUI、RGA

```
RKDocs\common\display\Rockchip_Trouble_Shooting_Graphics
```

Tool usage

StressTest

Use the Stresstest tool to do the stress test for the various functions on the target devices to make sure the whole system running stably. SDK can start StressTest application and perform stress test of various functions by entering `“83991906=”` code in the calculator.
The test items of Stresstest tool mainly include:

Module related

- Camera stress test: including Camera on/off, Camera taking photo and Camera switch.
- Bluetooth stress test: including Bluetooth on/off.
- Wi-Fi stress test: including Wi-Fi on/off, (plan to add ping test and iperf test).

Non module related

- Fly mode on/off test
- Suspend and resume stress test
- Video playing stress test
- Reboot stress test

- Recovery stress test
- ARM frequency scaling test
- GPU frequency scaling test
- DDR frequency scaling test

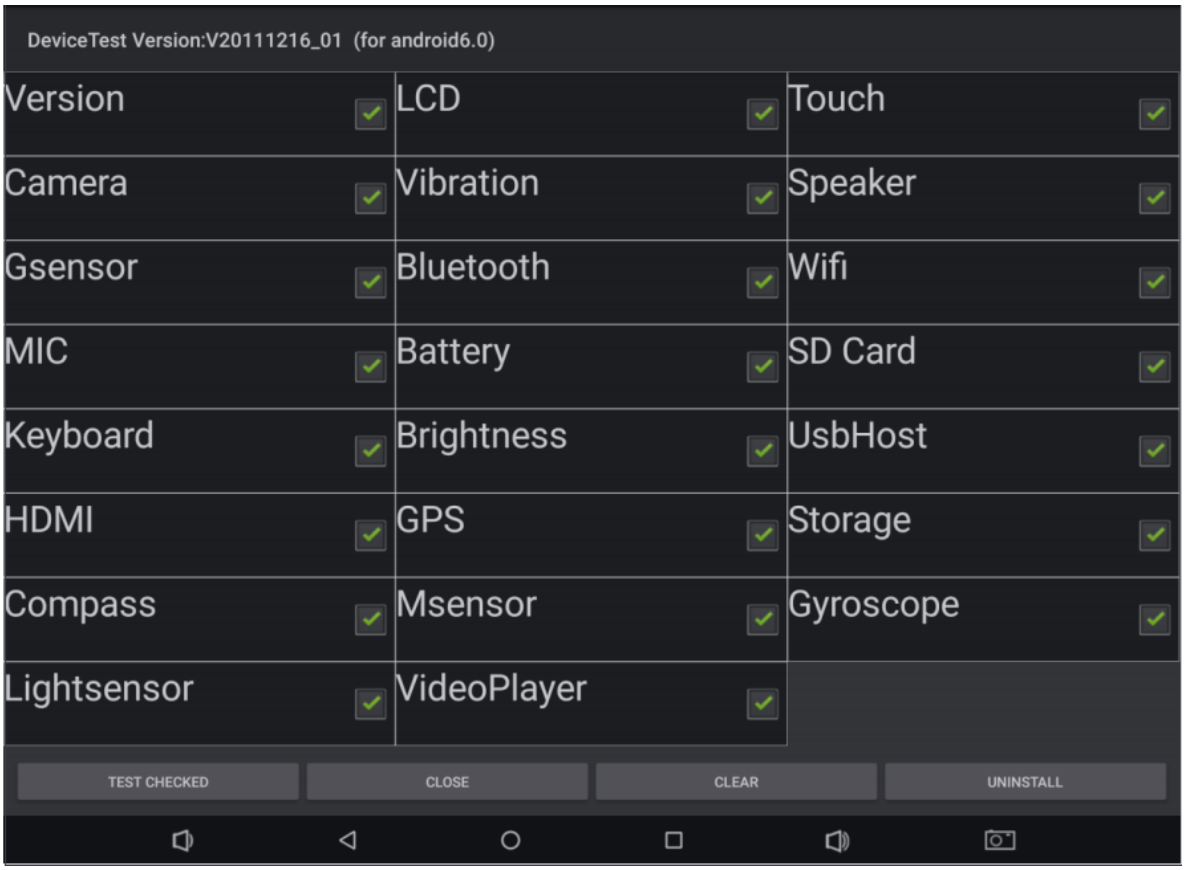
PCBA test tool

PCBA test tool is used to help quickly identify good and bad product features during production to improve the production efficiency. Current test items include panel (LCD), wireless (Wi-Fi), Bluetooth, DDR/EMMC memory, SD card, USB HOST, key, speaker earphone (Codec).
These test items include automatic test item and manual test item. Wireless network, DDR/EMMC, Ethernet are automatic test items, while key, SD card, USB Host, Codec are manual test items.
For the detailed PCBA function configuration and usage, please refer to:

```
RKDocs\android\Rockchip_Developer_Guide_PCBA_Test_Tool_CN&EN.pdf_V1.1_20171222.pdf。
```

DeviceTest

DeviceTest is used for the whole device test in factory, which mainly test whether the peripheral components work normally after assembling. SDK will enter DeviceTest by entering “000.=” code in the calculator, as shown below:



In factory, you can test the corresponding peripheral according to this interface. Click “TEST CHECKED” to test the items one by one. If succeed, click pass, if fail, click failed, the final result will display on the screen, as shown below. Red means failed item, others are pass, and the factory can repair accordingly based on the test result. Besides, if customers need to customize the tool, please contact FAE to apply for the corresponding source code.

USB driver

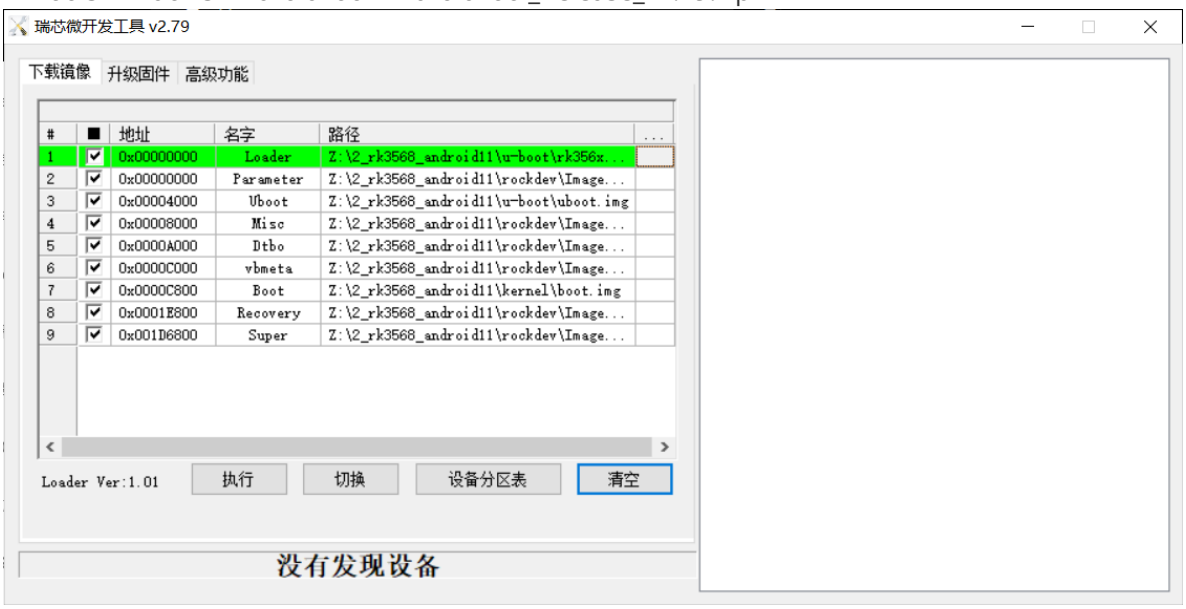
Rockchip USB driver install package includes ADB and image flashing driver

```
RKTools\windows\DriverAssitant_v5.1.1.zip
```

Development flashing tool

Windows version

RKTools/windows/AndroidTool/AndroidTool_Release_v2.79.zip



Linux version

RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.56.zip

```
Linux_Upgrade_Tool_v1.56$ sudo ./upgrade_tool -h
Program Data in /home/wlq/.config/upgrade_tool

----------------------Tool Usage ----------------------
Help:              H
Quit:              Q
Version:           V
Clear Screen:      CS
------------------Upgrade Command ------------------
ChooseDevice:            CD
ListDevice:                  LD
SwitchDevice:            SD
UpgradeFirmware:         UF <Firmware> [-noreset]
UpgradeLoader:           UL <Loader> [-noreset]
DownloadImage:           DI <-p|-b|-k|-s|-r|-m|-u|-t|-re image>
DownloadBoot:            DB <Loader>
EraseFlash:              EF <Loader|firmware> [DirectLBA]
PartitionList:           PL
WriteSN:                 SN <serial number>
ReadSN:              RSN
----------------Professional Command -----------------
```

```
TestDevice:             TD
ResetDevice:            RD [subcode]
ResetPipe:              RP [pipe]
ReadCapability:         RCB
ReadFlashID:            RID
ReadFlashInfo:          RFI
ReadChipInfo:           RCI
ReadSector:             RS  <BeginSec> <SectorLen> [-decode] [File]
WriteSector:            WS  <BeginSec> <File>
ReadLBA:                RL  <BeginSec> <SectorLen> [File]
WriteLBA:               WL  <BeginSec> <File>
EraseLBA:               EL  <BeginSec> <EraseCount>
EraseBlock:             EB <CS> <BeginBlock> <BlokcLen> [--Force]
-------------------------------------------------------
```

Tool to implement SD upgrading and boot

It is used to implement SD card upgrading, SD card boot, SD card PCBA test.

```
RKTools\windows\SDDiskTool_v1.59.zip
```

Write SN tool

RKTools\windows\RKDevInfoWriteTool_Setup_V1.0.3.rar
Install after unzip RKDevInfoWriteTool_Setup_V1.0.3.rar
Use admin ID to open the software

For the tool instruction, please refer to:

```
RKDocs\common\RKTools manuals\RKDevInfoWriteTool_User_Guide_V1.0.3.pdf
```

DDR welding test tool

It is used to test DDR hardware conncection, troubleshooting hardware issues such as virtual welding.

```
RKTools\windows\Rockchip_Platform_DDR_Test_Tool_V1.38_Release_Annoucement_CN.7z
RKTools\windows\Rockchip_Platform_DDR_Test_Tool_V1.38_Release_Annoucement_EN.7z
```

efuse flashing tool

It is used to flash efuse, suitable for RK3288W/RK3368/RK3399 platforms.

```
RKTools\windows\efuse_v1.37.rar
```

efuse/otp sign tool

It is used to sign efuse/otp of image.

```
RKTools\windows\SecureBootTool_v1.94.zip
```

Factory production image flashing tool

It is uesed for batch image flashing in factory.

```
RKTools\windows\FactoryTool_1.66.zip
```

Image update tool

It is used to modify update.img.

```
RKTools\windows\FWFactoryTool_V5.52.rar
```

userdata partition data prebuilt tool

It is the tool used to make userdata partition pre-built data package.

```
RKTools\windows\OemTool_v1.3.rar
```

System debugging

ADB tool

Overview

ADB (Android Debug Bridge) is a tool in Android SDK which can be used to operate and manage Android simulator or the real Android device. The functions mainly include:

- Run the device shell (command line)
- Manage the port mapping of the simulator or the device
- Upload/download files between the computer and the device
- Install the local apk to simulator or Android device
  ADB is a “client – server” program. Usually the client is PC and the server is the actual Android device or simulator. The ADB can be divided into two categories according to the way PC connects to the Android device:
- Network ADB: PC connects to STB device through cable/wireless network.
- USB ADB: PC connects to STB device through USB cable.

USB adb usage

USB adb usage has the following limitations:

- Only support USB OTG port
- Not support multiple clients at the same time (such as cmd window, eclipse etc.)
- Support host connects to only one device but not multiple devices
  The connection steps are as below:
  1、The device already running Android system, setting -> developer option -> connect to the computer, enable usb debugging switch.
  2、PC connects to the device USB otg port only through USB cable, and then the computer connects with Android device through below command:

```
adb shell
```

3、Execute the command ”adb devices” to see if the connection is successful or not. If the device serial number shows up, the connection is successful.

Network ADB usage requirement

ADB early versions only support device debugging through USB, and the function of debugging Android devices through tcp/ip is added from adb v1.0.25.
If you need to use network ADB to debug the device, must meet below conditions:
1、The device must have network port, or connect the network through Wi-Fi.
2、The device and PC are already in the local network and the device has IP address.
3、Need to confirm the device and PC can ping each other.
4、PC already installs abd.
5、Confirm Android device adbd process (adb background process) is already run. adbd process will monitor port 5555 to do adb connection debugging.

SDK network ADB port configuration

SDK doesn’t enable the network ADB by default. Need to manually enable in the developer option.
Setting-System-Advanced-Developer options-Open net adb

Network ADB usage

This chapter assumes the device IP is 192.168.1.5. This IP will be used for adb connection and device debugging in the following context.
1、Firstly the Android device should boot up, if possible, confirm adbd is started (use ps command to check).
2、In PC cmd, input:

```
adb connect 192.168.1.5:5555
```

If successful, it will prompt relative hints, if fail, you can execute kill-server command and then retry connection.

```
adb kill-server
```

3、After connected, you can input ADB related commands to debug in PC, such as adb shell, it will connect the device through TCP/IP which is the same as USB debugging.
4、After debugging, input the following command to disconnect the connection in PC:

```
adb disconnect 192.168.1.5:5555
```

Manually change the network ADB port number

If SDK doesn't add adb port number configuration, or want to change adb port number, you can change through below method:
1、Firstly also connect the device normally through USB, execute adb shell in windows cmd to enter.
2、Set ADB monitor port:

```
#setprop service.adb.tcp.port 5555
```

3、Look up adbd pid using ps command.
4、Reset adbd.

kill -9, This pid is just the one found in last step.

After killing adbd, adbd will automatically restart after Android init process. After adbd restart, if service.adb.tcp.port is set, it will automatically change to monitor network request.

ADB commonly used command elaboration

（1）Check the device situation
Check the Android device or simulator connected to computer:

```
adb devices
```

The return result is the serial number or IP and port number, status of the Android device connected to PC.
（2）Install APK
Install the specific apk file to the device:

```
adb install <apk file path>
```

For example:

```
adb install “F:\WishTV\WishTV.apk”
```

Re-install application:

```
adb install -r “F:\WishTV\WishTV.apk”
```

(3) Uninstall APK
Complete uninstall:

```
adb uninstall <package>
```

For example:

```
adb uninstall com.wishtv
```

(4) Use rm to remove apk file:

```
adb shell rm <filepath>
```

For example:

```
adb shell rm “system/app/WishTV.apk”
```

Note: remove “WishTV.apk” file in the directory of “system/app”.
(5) Enter shell of the device and simulator
Enthe the shell environment of the device or simulator:

```
adb shell
```

(6) Upload the file to the device from PC
Use push command can upload any file or folder from PC to the device. Generally local path means the computer and remote path means the single board device connected with ADB.
adb push
For example:

```
adb push “F:\WishTV\WishTV.apk” “system/app”
```

Note: upload local “WishTV.apk” file to the “system/app” directory of the Android system.
(7) Download the file from the device to PC
Use pull command can download the file or folder from the device to local computer.

```
adb pull <remote path> <local path>
```

For example:

```
adb pull system/app/Contacts.apk F:\
```

Note: download the file or folder from the “system/app” directory of Android system to local “F:\” directory.

（8）Check bug report

Run adb bugreport command can check all the error message report generated by system. The command will show all dumpsys, dumpstate and logcat information of the Android system.

（9）Check the device system information

The specific commands in adb shell to check the device system information.

```
adb shell getprop
```

Logcat tool

Android logcat system provides the function to record and check the system debugging information. The logcats are all recorded from various softwares and some system buffer. The buffer can be checked and used through Logcat. Logcat is the function most commonly used by debugging program. The function shows the program running status mainly by printing logcat. Because the amount of logcat is very large, need to do filtering and other operations.

Logcat command usage

Use logcat command to check the contents of the system logcat buffer:
The basic format:

```
[adb] logcat [<option>] [<filter-spec>]
```

For example:

```
adb shell
logcat
```

The commonly used logcat filter method

Several ways to control the logcat output:

- Control the logcat output priority
  For example:

```
adb shell
logcat *:w
```

Note: show the logcat information with priority of warning or higher.

- Control the logcat label and output priority
  For example:

```
adb shell
logcat ActivityManager:I MyApp:D *:S
```

Note: support all the logcat information except those with label of “ActivityManager” and priority of “Info” above, label of “MyApp” and priority of “Debug” above.

- Only output the logcat with the specific label
  For example:

```
adb shell
logcat WishTV:* *:S
```

or

```
adb shell
logcat –s WishTV
```

Note: only output the logcat with label of WishTV.

- Only output the logcat with the specific priority and label
  For example:

```
adb shell
logcat WishTV:I *:S
```

Note: only output the logcat with priority of I and label of WishTV.

Procrank tool

Procrank is a debugging tool with Android, running in the shell environment of the device, used to output the memory snapshot of the process in order to effectively observe the memory usage status of the process.
Include the following memory information:

- VSS：Virtual Set Size The memory size used by virtual (including the memory used by the shared lib)
- RSS：Resident Set Size The actually used physical memory size (including the memory used by the shared lib)
- PSS：Proportional Set Size The actually used physical memory size (allocate the memory used by the shared lib in proportion)
- USS：Unique Set Size The physical memory used exclusively by the process (not including the memory used by the shared lib)
  Note:
- USS size represents the memory size only used by the process, and it will be completely recovered after the process is killed.
- VSS/RSS includes the memory used by the shared lib, so it is not helpful to check the memory status of the single process.
- PSS is the shared memory status used by the specific single process after the shared memory is allocated in proportion.

Use procrank

Make sure the terminal has the root authority before executing procrank
su
The command format:

```
procrank [ -W ] [ -v | -r | -p | -u | -h ]
```

The commonly used command instructions:
-v：order by VSS
-r：order by RSS
-p：order by PSS
-u：order by USS
-R：convert to order by increasing[decreasing] method
-w：only display the statistical count of working set
-W：reset the statistical count of working set
-h：help
For example:
Output the memory snapshot:

```
procrank
```

Output the memory snapshot in VSS decreasing order:

```
procrank –v
```

Procrank is output in PSS order by default.

Search the specific content information

Use below command format to view the memory status of the specific process:

```
procrank | grep [cmdline | PID]
```

cmdline means the target application name, PID means the target application process.
Output the memory status used by systemUI process:

```
procrank | grep “com.android.systemui”
```

or:

```
procrank  | grep 3396
```

Trace the process memory status

Analyze if there is memory leakage in the process by tracing the memory usage status. Use the script to continuously output the process memory snapshot, and compare with USS segment to see if there is memory leakage in this process.
For example: output the application memory usage of the process named com.android.systemui to see if there is leakage:
1、Write the script test.sh

```
#!/bin/bash
while true;do
adb shell procrank | grep “com.android.systemui”
sleep 1
done
```

2、After connect to the device by adb tool, run the script: ./test.sh

Dumpsys tool

Dumpsys tool is a debugging tool in Android system, running in the shell environment of the device, and provides the service status information running in the system. The running service means the service process in the Android binder mechanism.
The conditions for dumpsys to output the print:
1、Only print the services already loaded to ServiceManager.
2、If the dump function in the service code is not implemented, there will be no information output.

Use Dumpsys

- View Dumpsys help
  Function: output dympsys help information.

```
  dumpsys  -help
```

- View the service list of Dumpsys
  Function: output all the printable service information of dumpsys, developer can pay attention to the service names required for debugging.

```
  dumpsys  -l
```

- Output the specific service information
  Function: output the specific service dump information.
  Format: dumpsys [servicename]
  For example: execute below command can output the service information of SurfaceFlinger:

```
dumpsys SurfaceFlinger
```

- Output the specific service and application process information
  Function: output the specific service and application process information.
  Format: dumpsys [servicename][application name]

For example: execute below command to output the memory information for the service named meminfo and process named com.android.systemui:

```
dumpsys meminfo com.android.systemui
```

Note: the service name is case sensitive and must input the full service name.

Last log enable

- Add the following two nodes in dts file

```
ramoops_mem: ramoops_mem {
    reg = <0x0 0x110000 0x0 0xf0000>;
    reg-names = "ramoops_mem";
};

ramoops {
    compatible = "ramoops";
    record-size = <0x0 0x20000>;
    console-size = <0x0 0x80000>;
    ftrace-size = <0x0 0x00000>;
    pmsg-size = <0x0 0x50000>;
    memory-region = <&ramoops_mem>;
};
```

```
- Check last log in the device
130|root@rk3399:/sys/fs/pstore # ls
dmesg-ramoops-0   Log saved after last kernel panic
pmsg-ramoops-0    Log of last user space, android log
ftrace-ramoops-0    Print function trace during some period
console-ramoops-0  kernel log when last_log was enabled last time, but only save
the log with higher priority than default log level
```

- Usage:

```
cat dmesg-ramoops-0
cat console-ramoops-0
logcat -L （pmsg-ramoops-0） pull out by logcat and parse
cat ftrace-ramoops-0
```

FIQ mode

You can input fiq command through the serial port to check the system status when the device crashes or gets stuck. The specific command is as below:

```
127|console:/ $ fiq
debug>  help
FIQ Debugger commands:
 pc            PC status
```

```
regs          Register dump
allregs       Extended Register dump
bt            Stack trace
reboot [<c>]  Reboot with command <c>
reset [<c>]   Hard reset with command <c>
irqs          Interupt status
kmsg          Kernel log
version       Kernel version
sleep         Allow sleep while in FIQ
nosleep       Disable sleep while in FIQ
console       Switch terminal to console
cpu           Current CPU
cpu <number>  Switch to CPU<number>
ps            Process list
sysrq         sysrq options
sysrq <param> Execute sysrq with <param>
```

log auto collection

- Collection content

  ```
  android: android log
  kernel : kernel log
  ```

- Enable method
- Open Developer options
- Setting-System-Advanced-Developer options-Android bug collector
- Save path of log

  ```
  data/vendor/logs/
  ```

Common issues

What is current kernel version and u-boot version?

```
The corresponding kernel version of Android11.0 is: develop-4.19, u-boot branch
is next-dev branch
```

How to acquire the corresponding RK release version for current SDK

Rockchip Android11.0 SDK includes AOSP source code and RK changed code. RK changed libs are involved in xml under the directory `.repo/manifests/include`, while AOSP default libs are in `.repo/manifests/default.xml`。
Version confirm:

- RK modification part

```
vim .repo/manifests/include/rk_checkout_from_aosp.xml
<project groups="pdk" name="platform/build" path="build/make" remote="rk"
revision="refs/tags/android-11.0-mid-rkr1">
```

Means RK version is android-11.0-mid-rkr1

- AOSP part

```
vim .repo/manifests/default.xml
<default revision="refs/tags/android-10.0.0_r14"...>
```

Means OASP version is android-10.0.0_r14
Just provide the above two version information when needed.
You can directly acquire tag information through the following command for single lib:

```
/kernel$ git tag
android-11.0-mid-rkr1
develop-4.4-20190201
```

RK version is incremental with the format of android-11.0-mid-rkrxx, so current latest tag is android-11.0-mid-rkr1

How to confirm if local SDK is already updated to the latest SDK version released by RK

When RK SDK is released, the commit information corresponding to the version will be submited under the .repo/manifests/commit/ directory. Customers can confirm whether SDK is updated completely or not by comparing with the commit information. The specific operations are as follows:

- First confirm RK version of SDK according to the instruction of "How to acquire the corresponding RK release version for current SDK". Below take RKR6 version as example to introduce.
- Use the following command to save local commit information

```
.repo/repo/repo manifest -r -o release_manifest_rkr6_local.xml
```

- Comparing .repo/manifests/commit/commit_release_rkr6.xml with release_manifest_rkr6_local.xml can confirm whether SDK code is completely updated or not, while .repo/manifests/commit/commit_release_rkr6.xml is the commit information released along with RKR6 version.

RM310 4G configuration

4G function is disabled in SDK by default. If need to enable, operate as below:

```
vim device/rockchip/common/BoardConfig.mk
#for rk 4g modem
-BOARD_HAS_RK_4G_MODEM ?= false
+BOARD_HAS_RK_4G_MODEM ?= true
```

Recovery rotation configuration

Support Recovery rotation with 0/90/180/270 degree. Disabled by default (that is to rotate 0 degree). The rotation configuration is described as below:

```
vim device/rockchip/common/BoardConfig.mk
#0:   ROTATION_NONE rotate 0 degree
#90:  ROTATION_RIGHT rotate 90 degrees
#180: ROTATION_DOWN rotate 180 degrees
#270: ROTATION_LEFT rotate 270 degrees
# For Recovery Rotation
TARGET_RECOVERY_DEFAULT_ROTATION ?= ROTATION_NONE
```

Android Surface rotation

For Android system display rotation, you can modify the following configuration with the parameters 0/90/180/270

```
# For Surface Flinger Rotation
SF_PRIMARY_DISPLAY_ORIENTATION ?= 0
```

Replace some remote of AOSP source code

The speed for customer to download RK release code is relatively slow. You can change the remote of AOSP to domestic mirror source, or Google mirror source for foreign customers, to improve the downloading speed. The detailed method is described as below:
After executing repo init (or unpacking base package), modify .repo/manifests/remote.xml. Change the remote fetch of AOSP from

```
< remote name="aosp" fetch="./" review="https://10.10.10.29" />
```

to
for domestic customers: (here we take Tsinghua university mirror source as example. You can change to other domestic mirror source)

```
< remote name="aosp" fetch="https://aosp.tuna.tsinghua.edu.cn" />;
```

for foreign customers: (Google mirror source)

```
< remote name="aosp" fetch="https://android.googlesource.com" />
```

Change userdata partition file system to EXT4

The default file system of data partition is f2fs. Recommend to change the file system of data partition to ext4 for the product without battery, as it can reduce the risk of data loss after abnormal power down. The modification method is as below:
Take rk3566_r as example:

```
device/rockchip/common$ git diff
diff --git a/scripts/fstab_tools/fstab.in b/scripts/fstab_tools/fstab.in
index 6e78b00..a658332 100755
--- a/scripts/fstab_tools/fstab.in
+++ b/scripts/fstab_tools/fstab.in
@@ -20,6 +20,6 @@ ${_block_prefix}system_ext /system_ext  ext4 ro,barrier=1
${_flags},first_stage_
 # For sdmmc
 /devices/platform/${_sdmmc_device}/mmc_host*        auto  auto    defaults
   voldmanaged=sdcard1:auto
 #  Full disk encryption has less effect on rk3326, so default to enable this.
-/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,quota,formattable,reservedsize=128M,checkpoint=fs
+#/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065
latemount,wait,check,fileencryption=aes-256-xts:aes-256-
cts:v2+inlinecrypt_optimized,quota,formattable,reservedsize=128M,checkpoint=fs
 # for ext4
-#/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
latemount,wait,formattable,check,fileencryption=software,quota,reservedsize=128M
,checkpoint=block
+/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
latemount,wait,formattable,check,fileencryption=software,quota,reservedsize=128M
,checkpoint=block
```

```
device/rockchip/rk356x$ git diff
diff --git a/rk3566_r/recovery.fstab b/rk3566_r/recovery.fstab
index 7532217..cf789ac 100755
--- a/rk3566_r/recovery.fstab
+++ b/rk3566_r/recovery.fstab
@@ -7,7 +7,7 @@
 /dev/block/by-name/odm                  /odm                ext4
defaults                  defaults
 /dev/block/by-name/cache                /cache              ext4
defaults                  defaults
 /dev/block/by-name/metadata             /metadata           ext4
defaults                  defaults
-/dev/block/by-name/userdata             /data               f2fs
defaults                  defaults
+/dev/block/by-name/userdata             /data               ext4
defaults                  defaults
 /dev/block/by-name/cust                 /cust               ext4
defaults                  defaults
```

```
 /dev/block/by-name/custom                /custom              ext4
defaults                  defaults
 /dev/block/by-name/radical_update        /radical_update      ext4
defaults                  defaults
```

Modify power on/off animation and tones

Reference document:

```
RKDocs\android\Rockchip_Introduction_Android_Power_On_Off_Animation_and_Tone_Cus
tomization_CN&EN.pdf
```

APP performance mode setting

Configure the file: package_performance.xml in device/rockchip/rk3xxx/. Add the package names which need to use performance mode in the node: (use aapt dump badging (file_path.apk) to acquire the package name)

```
< app package="package name" mode="whether to enable the acceleration, 1 for
enable, 0 for disable"/>
```

Take antutu as example as below:

```
< app package="com.antutu.ABenchMark"mode="1"/>
< app package="com.antutu.benchmark.full"mode="1"\/>
< app package="com.antutu.benchmark.full"mode="1"\/>
```

It will package the file into the image when compiling.

Debugging method of GPU related issues

You can do the initial debugging for the issues referring to the following documents.

```
RKDocs\android\Rockchip_User_Guide_Dr.G_CN&EN.pdf
```

OTP and efuse instruction

OTP support chipset

- RK3326
- PX30
- RK3566
- RK3568
  EFUSE support chipset
- RK3288
- RK3368

- RK3399
  Refer to the document for image signing and otp/efuse flashing:

```
RKDocs\common\security\Rockchip-Secure-Boot-Application-Note-V1.9.pdf
```

How to judge from the code whether OTP/EFUSE of the device is already flashed or not

The status of OTP/EFUSE will be transmitted through kernel cmdline and fuse.programmed in cmdline is used to mark the status of OTP/EFUSE. The details are as follows:

- "fuse.programmed=1": the software image package is already signed by secure-boot and efuse/otp of the hardware device is already flashed.
  -"fuse.programmed=0": the software image package is already signed by secure-boot but efuse/otp of the hardware device is not flashed.
- there is no fuse.programmed in cmdline: the software image package is not signed by secure-boot (Miniloader doesn't tramsmit), or Miniloader is too old to support transmission.

Enable/disable selinux

Refer to the following modification, false to disable, true to enable

```
device/rockchip/common$
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -67,7 +67,7 @@ endif

 # Enable android verified boot 2.0
 BOARD_AVB_ENABLE ?= false
-BOARD_SELINUX_ENFORCING ?= false
+BOARD_SELINUX_ENFORCING ?= true
```

Warning “There's an internal problem with your device.” pops up after boot up

There are two reasons to pop up the warning:

1. Image mismatch, the fingerprints of system/boot/vendor are inconsistent.
2. The device is enabled with a configuration that supports IO debugging. This problem can be solved by using the previous compile kernel command in the documentation.
3. For projects with IO debugging, regardless of the above two reasons, please merge the following patches directly to eliminate the warning:

```
diff --git
a/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
b/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
index 595c340..d4e495a 100644
---
a/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
+++
b/services/core/java/com/android/server/wm/ActivityTaskManagerService.java
@@ -6555,7 +6555,7 @@ public class ActivityTaskManagerService extends
IActivityTaskManager.Stub {
                } catch (RemoteException e) {
                }
```

- if (!Build.isBuildConsistent()) {
- if (0 && !Build.isBuildConsistent()) {

```
                Slog.e(TAG, "Build fingerprint is not consistent, warning
user");
                mUiHandler.post(() -> {
                    if (mShowDialogs) {
```

```
```

How to enable the setting options for Ethernet in Settings

There is no default option of Ethernet setting in the system Settings. If Ethernet is needed in the project, it can be turned on as follows:

```
  --- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -146,3 +146,6 @@ endif
 ifeq ($(strip $(BOARD_USES_AB_IMAGE)), true)
 DEVICE_MANIFEST_FILE :=
device/rockchip/$(TARGET_BOARD_PLATFORM)/manifest_ab.xml
 endif

+# for ethernet
+BOARD_HS_ETHERNET := true
```

About AVB

For AVB related instruction and configurations, you can refer to the document

```
RKDocs/common/u-boot/Rockchip_Developer_Guide_UBoot_Nextdev_CN.pdf
```

Cannot use IO commands

IO commands rely on DEVMEM which is disabled by default, so it is not able to use IO by default. If need to use IO commands for debugging, you can modify as follows:

```
wlq@ubuntu:~/1_source_code/a3_rk3399_android10.0_29/kernel$ vim
kernel/configs/android-11.config
```

For GO products, need to modify:

```
wlq@ubuntu:~/1_source_code/a3_rk3399_android10.0_29/kernel$ vim
kernel/configs/android-11-go.config
```

delete the following line:

```
# CONFIG_DEVMEM is not set
```

```
wlq@ubuntu:~/1_source_code/a3_rk3399_android10.0_29/kernel$ vim
configs/q/android-4.19/android-base.config
```

delete the following line:

```
# CONFIG_DEVMEM is not set
```

Both of them need to be deleted.

The SN naming rule

The SN must begin with a letter and be no more than 14 bytes.

APPENDIX A Compiling and development environment setup

Initializing a Build Environment

This section describes how to set up your local work environment to build the Android source files. You must use Linux or Mac OS; building under Windows is not currently supported.
For an overview of the entire code-review and code-update process, see Life of a Patch.
Note: All commands in this site are preceded by a dollar sign ($) to differentiate them from output or entries within files. You may use the Click to copy feature at the top right of each command box to copy all lines without the dollar signs or triple-click each line to copy it individually without the dollar sign.

Choosing a Branch

Some requirements for the build environment are determined by the version of the source code you plan to compile. For a full list of available branches, see Build Numbers. You can also choose to download and build the latest source code (called master), in which case you will simply omit the branch specification when you initialize the repository.
After you have selected a branch, follow the appropriate instructions below to set up your build environment.

Setting up a Linux build environment

These instructions apply to all branches, including master.
The Android build is routinely tested in house on recent versions of Ubuntu LTS (14.04) and Debian testing. Most other distributions should have the required build tools available.
For Gingerbread (2.3.x) and newer versions, including the master branch, a 64-bit environment is required. Older versions can be compiled on 32-bit systems.
Note: See Requirements for the complete list of hardware and software requirements, then follow the detailed instructions for Ubuntu and Mac OS below.

Installing the JDK

The master branch of Android in the Android Open Source Project (AOSP) comes with prebuilt versions of OpenJDK below prebuilts/jdk/ so no additional installation is required.
Older versions of Android require a separate installation of the JDK. On Ubuntu, use OpenJDK. See JDK Requirements for precise versions and the sections below for instructions.
For Ubuntu >= 15.04
Run the following:

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

For Ubuntu LTS 14.04
There are no available supported OpenJDK 8 packages for Ubuntu 14.04. The Ubuntu 15.04 OpenJDK 8 packages have been used successfully with Ubuntu 14.04. Newer package versions (e.g. those for 15.10, 16.04) were found not to work on 14.04 using the instructions below.

1. Download the .deb packages for 64-bit architecture from old-releases.ubuntu.com:

```
openjdk-8-jre-headless_8u45-b14-1_amd64.deb with SHA256
0f5aba8db39088283b51e00054813063173a4d8809f70033976f83e214ab56c0
openjdk-8-jre_8u45-b14-1_amd64.deb with SHA256
9ef76c4562d39432b69baf6c18f199707c5c56a5b4566847df908b7d74e15849
openjdk-8-jdk_8u45-b14-1_amd64.deb with SHA256
6e47215cf6205aa829e6a0a64985075bd29d1f428a4006a80c9db371c2fc3c4c
```

1. Optionally, confirm the checksums of the downloaded files against the SHA256 string listed with each package above. For example, with the sha256sum tool:

```
    sha256sum {downloaded.deb file}
```

1. Install the packages:

```
    sudo apt-get update
```

Run dpkg for each of the .deb files you downloaded. It may produce errors due to missing dependencies:

```
    sudo dpkg -i {downloaded.deb file}
```

To fix missing dependencies:

```
sudo apt-get -f install
```

Update the default Java version - optional
Optionally, for the Ubuntu versions above update the default Java version by running:

```
sudo update-alternatives --config javasudo update-alternatives --config javac
```

Note: If, during a build, you encounter version errors for Java, see Wrong Java version for likely causes and solutions.
Installing required packages (Ubuntu 14.04)
You will need a 64-bit version of Ubuntu. Ubuntu 14.04 is recommended.

```
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
```

Note: To use SELinux tools for policy analysis, also install the python-networkx package. Note: If you are using LDAP and want to run ART host tests, also install the libnss-sss:i386 package.

Configuring USB Access

Under GNU/linux systems (and specifically under Ubuntu systems), regular users can't directly access USB devices by default. The system needs to be configured to allow such access.
The recommended approach is to create a file /etc/udev/rules.d/51-android.rules (as the root user) and to copy the following lines in it. must be replaced by the actual username of the user who is authorized to access the phones over USB.

```
# adb protocol on passion (Rockchip products)
SUBSYSTEM=="usb", ATTR{idVendor}=="2207", ATTR{idProduct}=="0010", MODE="0600",
OWNER="<username>"
```

Those new rules take effect the next time a device is plugged in. It might therefore be necessary to unplug the device and plug it back into the computer.
This is known to work on both Ubuntu Hardy Heron (8.04.x LTS) and Lucid Lynx (10.04.x LTS). Other versions of Ubuntu or other variants of GNU/linux might require different configurations.

References : http://source.android.com/source/initializing.html

APPENDIX B SSH public key operation instruction

APPENDIX B-1 SSH public key generation

Use the following command to generate:

```
ssh-keygen -t rsa -C "user@host"
```

Please replace user@host with your email address.

It will generate the key file in your directory after the command is executed successfully.

Please keep carefully the generated private key file id_rsa and password, and send id_rsa.pub to SDK release server admin through email.

APPENDIX B-2 Use key-chain to manage the key

Recommend you use the simple tool keychain to manage the key.
The detailed usage is as follows:

1. Install keychain software package:

```
$sudo aptitude install keychain
```

1. Configure to use the key:

```
$vim ~/.bashrc
```

Add the following command:

```
eval `keychain --eval ~/.ssh/id_rsa`
```

Among which, id_rsa is the file name of the private key.
Log in the console again after configuring as above, and it will prompt to input the password.
Only need to input the password used for generating the key if there is one.
Besides, please avoid using sudo or root user unless you know clearly how to deal with, otherwise it will cause the authority and key management problems.

APPENDIX B-3 Multiple devices use the same ssh public key

In order to use on different devices, you can copy ssh private key file id_rsa to the target device "~/.ssh/id_rsa".
Below hint will show up if using the wrong private key. Please replace with the correct private key.

After adding the correct private key, you can use git to clone code, shown as below picture:

Below error may occur when adding ssh private key:

```
Agent admitted failture to sign using the key
```

Input the following command at console can fix it.

```
ssh-add ~/.ssh/id_rsa
```

APPENDIX B-4 Switch different ssh public keys on one device

You can refer to ssh_config document to configure ssh.

```
~$ man ssh_config
```

Use the following commands to configure ssh for current user.

```
~$ cp /etc/ssh/ssh_config ~/.ssh/config
~$ vi .ssh/config
```

As below picture, identify another directory ssh file "~/.ssh1/id_rsa" as certificate private key. In this way, you can switch different keys.

APPENDIX B-5 Key authority management

The server can real-time monitor for the specific key the download times, IP and other information. If any abnormal case is found, it will prohibit the download authority of the corresponding key.
Please keep carefully the private key file. DO NOT re-authorize it to the third party.

APPENDIX B-6 Git authority application instruction

Refer to above chapters, generate the public key file, and send email to fae@rock-chips.com applying for SDK code download authority.