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

Rockchip Android 10.0 SDK Developer Guide

Status: [ ] Draft [√] Released [ ] Modifying	File No.:	RK-KF-YF-219
	Current Version:	V1.2.3
	Author:	Wu Liangqing
	Finish Date:	2020-05-6
	Auditor:	Chen Haiyan
	Finish Date:	2020-05-6

Version no.	Author	Revision Date	Revision Description	Remark
V1.0	Wu Liangqing	2019-12-13	Initial version release	
V1.2	Wu Liangqing	2019-12-31	Release for RK3399	
V1.2.1	Wu Liangqing	2020-03-24	Added some common issues	
V1.2.2	Wu Liangqing	2020-04-16	Release for RK3368；Added some common issues	
V1.2.3	Wu Liangqing	2020-05-06	Release for RK3288	

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

Rockchip Android 10.0 SDK Chipset support

Chipset platform	Support or not	SDK version
RK3326	Support	RKR1
PX30	Support	RKR1
RK3126C	Support	RKR3
RK3399	support	RKR5
RK3368	support	RKR7
RK3288	support	RKR8

Rockchip Android 10.0 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_Qt/manifests.git -m Android10.xml
```

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_Android10.0_SDK_Release.tar.gz.*` as an example, you can sync the source code through the following command after getting the initial package:

```
mkdir Rockcip_Android10.0_SDK
cat Rockchip_Android10.0_SDK_Release.tar.gz* | tar -zx -C
Rockcip_Android10.0_SDK
cd Rockcip_Android10.0_SDK
.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 rk3328_box-userdebug
============================================================
```

Compiling command introduction

Chipset	Uboot compiling	kernel compiling	Android compiling
RK3326	cd uboot ./make.sh rk3326	cd kernel make ARCH=arm64 rockchip_defconfig android-10.config rk3326.config make ARCH=arm64 rk3326-863-lp3-v10-rkisp1.img -j24	source build/envsetup.sh lunch rk3326_q-userdebug make -j24 ./mkimage.sh
PX30	cd uboot ./make.sh px30	cd kernel make ARCH=arm64 rockchip_defconfig android-10.config rk3326.config make ARCH=arm64 px30-evb-ddr3-v10-avb.img -j24	source build/envsetup.sh lunch PX30_Android10-userdebug make -j24 ./mkimage.sh

Chipset	Uboot compiling	kernel compiling	Android compiling
RK3368	cd uboot ./make.sh rk3368	cd kernel make ARCH=arm64 rockchip_defconfig android-10.config make ARCH=arm64 rk3368-808-evb.img -j20	source build/envsetup.sh lunch rk3368_Android10-userdebug make -j24 ./mkimage.sh
RK3399	cd uboot ./make.sh rk3399	cd kernel make ARCH=arm64 rockchip_defconfig android-10.config rk3399.config excavator make ARCH=arm64 rk3399-sapphire-excavator-edp-avb.img -j24 RK3399 LPDDR4+RK809 IND development board make ARCH=arm64 rk3399-evb-ind-lpddr4-android-avb.img -j24	source build/envsetup.sh lunch rk3399_Android10-userdebug make -j24 ./mkimage.sh
RK3126C	cd uboot ./make.sh rk3126	cd kernel make ARCH=arm rockchip_defconfig android-10.config make ARCH=arm rk3126-bnd-d708-avb.img -j24	source build/envsetup.sh lunch rk3126c_q-userdebug make -j24 ./mkimage.sh
RK3328	cd uboot ./make.sh rk3328	cd kernel make ARCH=arm64 rockchip_defconfig android-10.config ## unicom device BOX compiling command: make ARCH=arm64 rk3328-box-liantong-avb.img -j24 ## EVB board compiling command: make ARCH=arm64 rk3328-evb-android-avb.img -j24	source build/envsetup.sh ## unicom device BOX compiling command： lunch rk3328_box-userdebug ## EVB board ATV compiling command: lunch rk3328_atv-userdebug make -j24 ./mkimage.sh
RK3229	cd uboot ./make.sh rk322x	cd kernel make ARCH=arm rockchip_defconfig android-10.config ## EVB board compiling command: make ARCH=arm rk3229-evb-android-avb.img -j24	source build/envsetup.sh ## EVB board BOX compiling command: lunch rk3328_box-userdebug make -j24 ./mkimage.sh

Chipset	Uboot compiling	kernel compiling	Android compiling
RK3288	cd uboot ./make.sh rk3288	cd kernel make ARCH=arm rockchip_defconfig android-10.config ## EVB board compiling command: make ARCH=arm rk3288-evb-android-rk808-edp-avb.img -j24	source build/envsetup.sh ## EVB board: lunch rk3288_Android10-userdebug make -j24 ./mkimage.sh

Other compiling instruction

Android10.0 cannot directly flash kernel.img and resource.img

Android10.0 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 need to use the `BOOT_IMG=xxx` parameter to specify the path of boot.img when compiling. The command is as below:
Take `RK3326 863` device as an example, replace the corresponding boot.img and dts when compiling:

```
cd kernel
make ARCH=arm64 rockchip_defconfig
make ARCH=arm64 BOOT_IMG=../rockdev/rk3326_q/boot.img rk3326-863-lp3-v10-rkisp1.img -j24
```

After compiling, you can directly flash the boot.img under kernel directory（**Note: it is zboot.img for 32bit platform, such as 3126c**） 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.

Support to upgrade from P to Q version:

Take RK3326 as example

- Compiling command:

```
source build/envsetup.sh
lunch rk3326_pie-userdebug
make -j24
./mkimage.sh
```

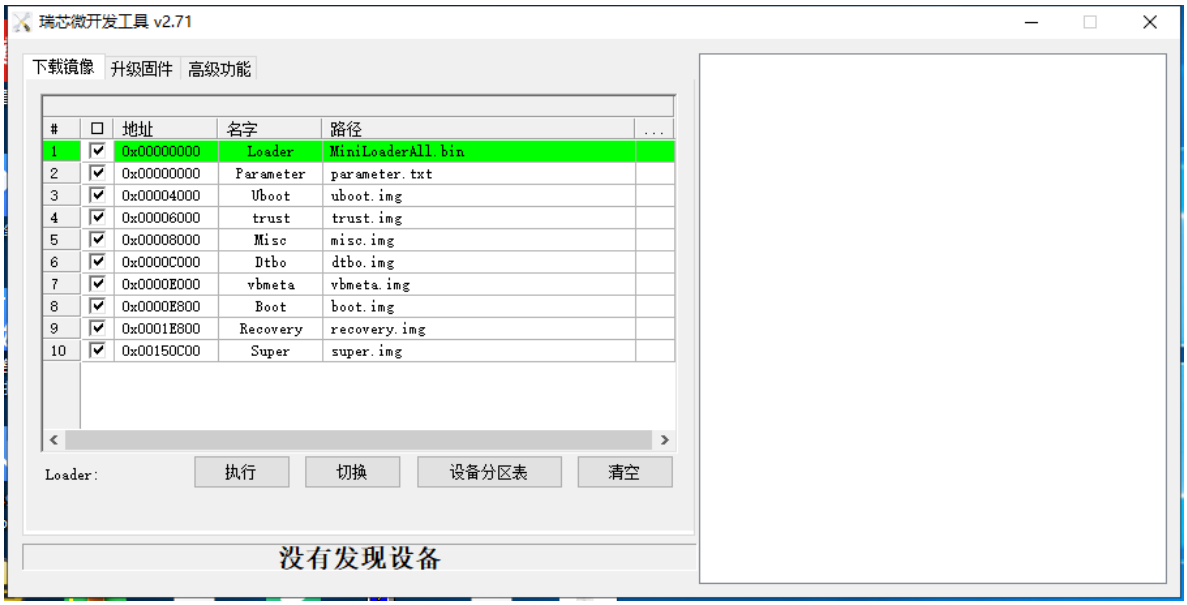
- Please flash odm.img to oem partition. Attach the flashing tool configuration:

Image flashing

Image flashing tool

Windows flashing tool:

```
RKTools/windows/AndroidTool/AndroidTool_Release_v2.71
```



Linux tool:

```
RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.43
```

There are more details in the tool instruction chapter.

Image instruction

After complete compiling, it will generate the following file:（take RK3326 as example, here lunch rk3326_qgo-userdebug)

```
rockdev/Image-rk3326_qgo/
├── boot.img
├── config.cfg
├── dtbo.img
├── kernel.img
├── MiniLoaderAll.bin
├── misc.img
├── odm.img
├── parameter.txt
├── pcba_small_misc.img
├── pcba_whole_misc.img
├── recovery.img
├── resource.img
├── super.img
├── system.img
├── trust.img
├── uboot.img
├── update.img
├── vbmeta.img
└── vendor.img
```

Just use the tool to flash the following files:

```
rockdev/Image-rk3326_qgo/
├── boot.img
├── dtbo.img
├── MiniLoaderAll.bin
├── misc.img
├── parameter.txt
├── recovery.img
├── super.img
├── trust.img
├── uboot.img
├── vbmeta.img
```

or you can directly flash `update.img`

Image instruction

Image	Instruction
boot.img	including ramdis、kernel、dtb
dtbo.img	Device Tree Overlays refer to dtbo chapter instruction later

| kernel.img | including kernel, currently it cannot be flashed alone, need to be packaged into boot.img for flashing.
MiniLoaderAll.bin	including first level loader
misc.img	including recovery-wipe boot symbol information, after flashing it will enter recovery
odm.img	including android odm, involved in super.img partition, and should be flashed together with fastboot
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
resource.img	including the log of dtb, kernel and uboot stages and uboot charging logo, currently it cannot be flashed alone, need to be packaged into boot.img for flashing
super.img	including the contents of odm、vendor、system partitions
system.img	including android system, involved in super.img partition, and should be flashed together with fastboot
trust.img	including BL31、BL32
uboot.img	including uboot image
vbmeta.img	including avb verification information, used for AVB verification
vendor.img	including android vendor, involved in super.img partition, and should be flashed together with fastboot
update.img	including the above img files to be flashed, can be used for the tool to directly flash the whole image package

Use fastboot to flash dynamic partition

The new device with Q supports dynamic partition, and already removes system/vendor/odm partitions. Please flash super.img. Use `fastbootd` can flash system/vendor/odm alone. 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:

The way to flash GSI:

- After the device is unlocked, enter fastbootd, only need to flash system.img of GSI and misc.img of the image, and after flashing it will enter recovery to do factory reset. Attach the complete flashing process as below:

1. Reboot to bootloader, lock->unlock the device:

```
adb reboot bootloader
fastboot oem at-unlock-vboot ## for the customers already flashing avb public
key, please refer to the corresponding document to unlock.
```

2. Reset to factory setting, reboot to fastbootd:

```
fastboot flash misc misc.img
fastboot reboot fastboot  ## now it will enter fastbootd
```

3. Start to flash GSI

```
fastboot delete-logical-partition product  ## (optional) for the device with
small partition space, you can execute this command to delete product partition
first and then flash GSI
fastboot flash system system.img
fastboot reboot  ## after flashing successfully, reboot the device
```

- Use DSU(Dynamic System Updates) to flash GSI, and current Rockchip platform already supports DSU by default. As this function requires large memory, it is not recommended to use on the device with 1G DDR or less. For the instruction and usage of DSU, please refer to Android official website:
  https://source.android.com/devices/tech/ota/dynamic-system-updates

Android 10.0 features

Compiling system

SDK version number change

First of all, the macro `PLATFORM_VERSION` is canceled in PDK code, and in later AOSP, this macro will be changed to 10.0, which will make the variable judged wrong by compiling system, so uniformly change it to `PLATFORM_SDK_VERSION`, if you find some feature not workable or crashed, go to the source code directory to check if this macro is called or not.
The corresponding relationship between them is as below:

PLATFORM_VERSION	PLATFORM_SDK_VERSION
7.0	24
7.1	25
8.0	26
8.1	27
9.0	28
10.0	29
...	...

mk file loading repeatedly

This issue always exists from early version without much influence, which only affects some compiling speed. New version introduces `readonly` mechanism, and the variable specified as read only cannot be assigned value again, so the compiling will report error when it is loaded repeatedly. Recommend the following struct to avoid repeat loading:

```
# First lunching is Q, api_level is 29
PRODUCT_SHIPPING_API_LEVEL := 29
PRODUCT_FSTAB_TEMPLATE := $(LOCAL_PATH)/fstab.in
PRODUCT_DTBO_TEMPLATE := $(LOCAL_PATH)/dt-overlay.in
PRODUCT_BOOT_DEVICE := ff390000.dwmmc,ff3b0000.nandc

include device/rockchip/common/build/rockchip/DynamicPartitions.mk
include device/rockchip/common/BoardConfig.mk
include device/rockchip/rk3326/rk3326_q/BoardConfig.mk
$(call inherit-product, device/rockchip/rk3326/device-common.mk)
$(call inherit-product, device/rockchip/common/device.mk)
$(call inherit-product, frameworks/native/build/tablet-10in-xhdpi-2048-dalvik-
heap.mk)

PRODUCT_CHARACTERISTICS := tablet

PRODUCT_NAME := rk3326_q
PRODUCT_DEVICE := rk3326_q
PRODUCT_BRAND := rockchip
PRODUCT_MODEL := rk3326_q
PRODUCT_MANUFACTURER := rockchip
PRODUCT_AAPT_PREF_CONFIG := tvdpi
#
## add Rockchip properties
#
PRODUCT_PROPERTY_OVERRIDES += ro.sf.lcd_density=160
```

> Comparing with old version, it mainly adjusts the order, and deletes some include {.is-info}

copy mechanism of mk file

in mk, macro `PRODUCT_COPY` can only be executed once, so when the target file name for copying is the same, only the command declared previously can be executed. The common issue is that, `adb` doesn't work after bringup.

fstab dynamic generator

As there are many configurable items in Android, and SDK will be used for industrial applications, the configuration conflict is unavoidable. Add this script to reduce the additional changes when modifying the configurations. The usage is to:
Create template file, and configure in mk:
`PRODUCT_FSTAB_TEMPLATE := $(LOCAL_PATH)/fstab.in`

```
# Android fstab file.
#<src>                                                <mnt_point>         <type>
<mnt_flags and options>                              <fs_mgr_flags>
# The filesystem that contains the filesystem checker binary (typically /system)
cannot
# specify MF_CHECK, and must come before any filesystems that do specify
MF_CHECK
${_block_prefix}system /system ext4 ro,barrier=1
${_flags_vbmeta},first_stage_mount
${_block_prefix}vendor /vendor ext4 ro,barrier=1 ${_flags},first_stage_mount
${_block_prefix}odm    /odm    ext4 ro,barrier=1 ${_flags},first_stage_mount
/dev/block/by-name/metadata /metadata ext4 nodev,noatime,nosuid,discard,sync
wait,formattable,first_stage_mount
/dev/block/by-name/misc         /misc               emmc      defaults
defaults
/dev/block/by-name/cache          /cache              ext4
noatime,nodiratime,nosuid,nodev,noauto_da_alloc,discard
wait,check

/devices/platform/*usb*   auto vfat defaults      voldmanaged=usb:auto

/dev/block/zram0                                  none                swap
defaults                                               zramsize=50%
# For sdmmc
/devices/platform/ff370000.dwmmc/mmc_host*        auto  auto    defaults
voldmanaged=sdcard1:auto,encryptable=userdata
#  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,fsync_mode=nobarrier
latemount,wait,check,fileencryption=software,quota,formattable,reservedsize=128M
,checkpoint=fs
```

dtbo.img dynamic generator

The reason to add this function is the same above. It can be compatible with other functions, such as `SD Boot`, this function requires to modify `boot_devices` as the address of `sdmmc`. The usage is to:

Create template file, and configure in mk. **Note, when using dtbo there must be alias existing in dts, otherwise it will fail to overlay:**

```
PRODUCT_DTBO_TEMPLATE := $(LOCAL_PATH)/dt-overlay.in
PRODUCT_BOOT_DEVICE := ff390000.dwmmc,ff3b0000.nandc //it can boot from sd card when changed to sdmmc
```

```
/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>;
};
```

ioctl control mechanism of sepolicy

The kernel node of libsync lib access is changed from `/dev/sw_sync` to `/sys/kernel/debug/sync/sw_sync`, and this modification will lead to many sepolicy issues. So, need to describe ioctl control mechanism of sepolicy.

- After adding ioctl in rules, also need to declare the specific iocmd, and must add based on the source code, for example:

```
allow hal_graphics_composer_default debugfs_sw_sync:file { ioctl };
allowxperm hal_graphics_composer_default debugfs_sw_sync:file ioctl {
SYNC_IOC_MERGE SW_SYNC_IOC_INC SW_SYNC_IOC_CREATE_FENCE };
```

New Feature or update

Screen rotation

Android 10.0 cancels ConfigStoreHAL, which has many issues and costs high memory, and reverts back to `prop` method. For this, SDK already modified accordingly. The usage is to:
Specify the rotation direction in mk file:

```
# set screen rotation: 0/90/180/270
SF_PRIMARY_DISPLAY_ORIENTATION := 0
```

`manifest` upgrade

After introducing `treble` technology (from `Android Oreo`), all hal should be declared in manifest, and `target-level` should be updated in new Android versions, so if your manifest.xml file is not modified, the compiling will fail. Referring to:

```
device/rockchip/rk3326/manifest.xml
```

Introduce `boot header 2`

Based on the original `header 1`, add `dtb` support. Need to package `dtb` (multiple) to `boot.img` when compiling. For this, new product needs to specify the directory of dtb:

```
BOARD_INCLUDE_DTB_IN_BOOTIMG := true
BOARD_PREBUILT_DTBIMAGE_DIR := kernel/arch/arm64/boot/dts/rockchip
```

Fot the device upgrading to Q, need to set null:

```
# No need to place dtb into boot.img for the device upgrading to Q.
BOARD_INCLUDE_DTB_IN_BOOTIMG :=
BOARD_PREBUILT_DTBIMAGE_DIR :=
```

Introduce `early mount`

In the latest Android 10, ramdisk is back to boot.img again, so mount fstab configuration of the partition needs to be configured in fstab file again. But different from 8.1, fstab of ramdisk also supports earlymount, while 8.1 earlymount can only be configured in dts file. **in 10.0, need to ensure the node is empty or in disabled state**. Therefore, need to copy the fstab file to `vendor/etc` and `ramdisk` at the same time. Here recommend to use `fstab dynamic generator`, which will automatically generate fstab file supporting earlymount and copy to the corresponding directory. If not use the generator, then copy through the following macro:

```
ifndef PRODUCT_FSTAB_TEMPLATE
$(warning Please add fstab.in with PRODUCT_FSTAB_TEMPLATE in your product.mk)
# To use fstab auto generator, define fstab.in in your product.mk,
# Then include the device/rockchip/common/build/rockchip/RebuildFstab.mk in your
AndroidBoard.mk

PRODUCT_COPY_FILES += \

$(TARGET_DEVICE_DIR)/fstab.rk30board:$(TARGET_COPY_OUT_VENDOR)/etc/fstab.rk30boa
rd \

$(TARGET_DEVICE_DIR)/fstab.rk30board:$(TARGET_COPY_OUT_RAMDISK)/fstab.rk30board
endif # Use PRODUCT_FSTAB_TEMPLATE
```

New `kernel config` requirement

It adds many config in new Android 10.0 and later GMS certificate, so it is recommended to use new kernel compiling command in new compiling system: `make ARCH=arm64 rockchip_defconfig android-10.config`

Dynamic partition

The variables in this function are all read only, so you can separate them into separate files and handle them in modules. The usage is to:
Just include this file in mk file:

```
include device/rockchip/common/build/rockchip/DynamicPartitions.mk
```

If the reserved size of these partitions（system/vendor/odm）is too small, when push big file, you can modify this file:

```
BOARD_SYSTEMIMAGE_PARTITION_RESERVED_SIZE := 52428800
BOARD_VENDORIMAGE_PARTITION_RESERVED_SIZE := 52428800
BOARD_ODMIMAGE_PARTITION_RESERVED_SIZE := 52428800
```

fastbootd

This function is workable only after the dynamic partition is enabled. There are two points should be modified. **Note not to modify vendorId here, currently it uses google's bootloader driver, Windows platform also can recognize and flash image normally**:

- init.recovery.rk30board.rc file

```
on early-fs
    setprop sys.usb.controller "ff300000.usb" //modify to the controller of the
corresponding platform
    setprop sys.usb.configfs 1

on fs && property:sys.usb.configfs=1
    write /config/usb_gadget/g1/bcdDevice 0x0310
    write /config/usb_gadget/g1/bcdUSB 0x0200
    write /config/usb_gadget/g1/os_desc/b_vendor_code 0x1
    write /config/usb_gadget/g1/os_desc/qw_sign "MSFT100"
    write /config/usb_gadget/g1/configs/b.1/MaxPower 500
    symlink /config/usb_gadget/g1/configs/b.1 /config/usb_gadget/g1/os_desc/b.1
```

- reboot mode supports fastboot. The template mentioned in previous `dtbo dynamic generator` is just the template supporting reboot mode.

File encryption

Actually this function is not new, but it is just introduced in rockchip sdk, as it is also the mandatary function required by android 10.
Enable file encryption, modify fstab file. The template in `fstab dynamic generator` is just the template supporting file encryption:

```
/dev/block/by-name/userdata       /data               f2fs
noatime,nodiratime,nosuid,nodev,discard,inline_xattr
wait,check,notrim,fileencryption=software,quota,reservedsize=128M
```

Besides the modification of fstab file, it also supports loading partition by stage to improve the bootup speed. **Note, loading partition by stage can only be used for file encryption, please check especially for disk encryption, otherwise it will fail to boot**:
Example of loading by stage:

```
# do mount_all early can imporve boot time when FBE
# is enabled
on fs
    mount_all /vendor/etc/fstab.${ro.hardware} --early
on late-fs
    # Start services for bootanim
    start servicemanager
    start hwcomposer-2-1
    start gralloc-2-0
    start surfaceflinger
    start bootanim

    # Mount RW partitions which need run fsck
    mount_all /vendor/etc/fstab.${ro.hardware} --late
```

Example of normal loading:

```
on fs
    mount_all /vendor/etc/fstab.${ro.hardware}
```

User data check point (UDC)

This function requires data partition should be f2fs, modify fstab file to support this function. The template in `fstab dynamic generator` is just the template supporting UDC:

```
/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065,fsync_mode=nobarrier
latemount,wait,check,fileencryption=software,quota,formattable,reservedsize=128M
,checkpoint=fs
```

avb vbmeta adds public key metadata

It requires certificate to unlock the higher level avb lock. For more details please refer to uboot security related document. SDK specifies the following macro to enable the function of compiling metadata:

```
BOARD_AVB_ENABLE := true
BOARD_AVB_METADATA_BIN_PATH := \
   external/avb/test/data/atx_metadata.bin
```

Android common configuration

Create product lunch

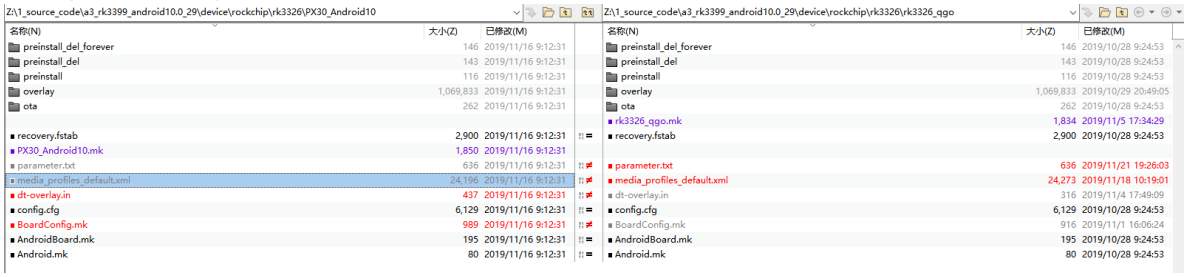
Take RK3326 platform as example to create a new PX30_Android10 product. The steps are as below:
1.Modify device/rockchip/rk3326/AndroidProducts.mk to add PX30_Android10 lunch

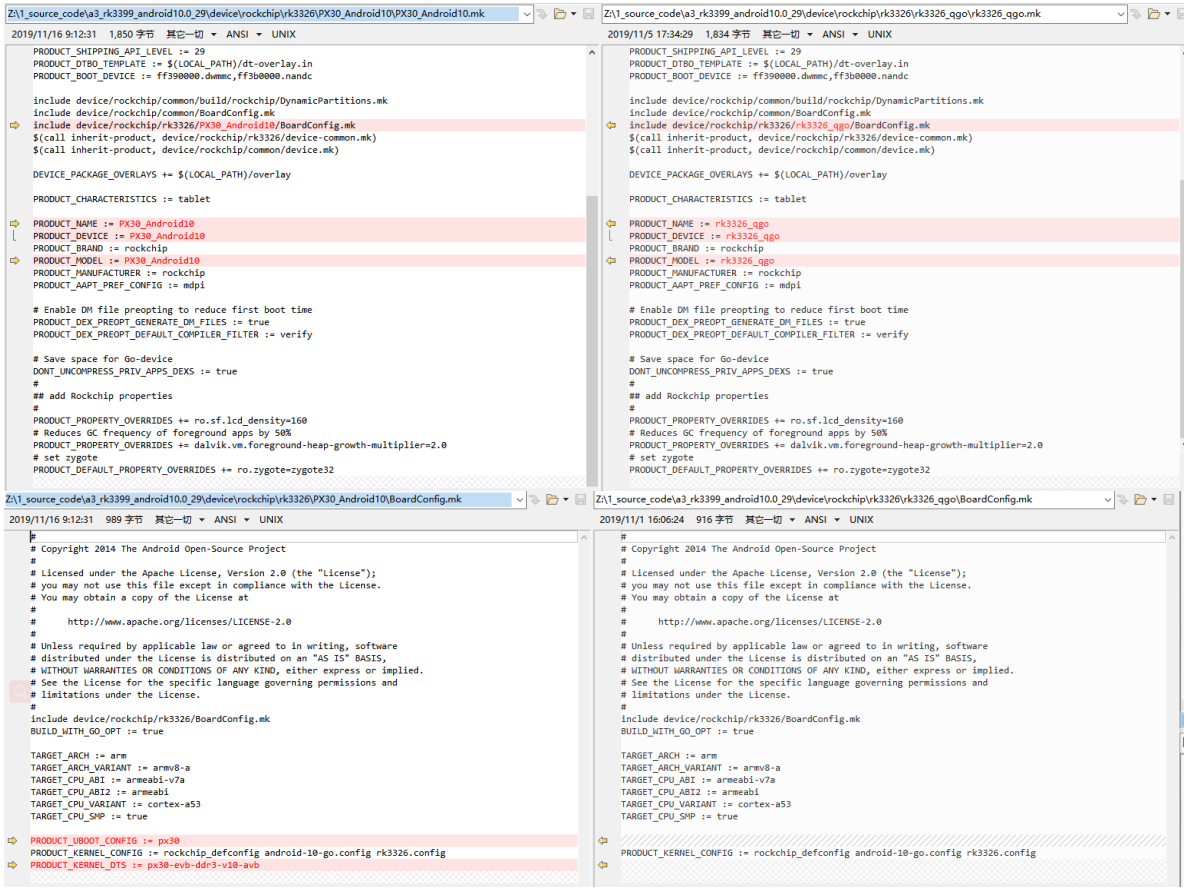
```
diff --git a/AndroidProducts.mk b/AndroidProducts.mk
index 44d8475..e2f7c02 100755
--- a/AndroidProducts.mk
+++ b/AndroidProducts.mk
@@ -15,11 +15,14 @@
 #

 PRODUCT_MAKEFILES := \
+        $(LOCAL_DIR)/PX30_Android10/PX30_Android10.mk \
         $(LOCAL_DIR)/rk3326_pie/rk3326_pie.mk \
         $(LOCAL_DIR)/rk3326_q/rk3326_q.mk \
         $(LOCAL_DIR)/rk3326_qgo/rk3326_qgo.mk

 COMMON_LUNCH_CHOICES := \
+       PX30_Android10-userdebug \
+       PX30_Android10-user \
         rk3326_q-userdebug \
         rk3326_q-user \
         rk3326_qgo-userdebug \
```

2.Create PX30_Android10 directory under device/rockchip/rk3326

Create referring to the existing rk3326_qgo product directory in device/rockchip/. You can directly copy rk3326_qgo to PX30_Android10, and then replace all the `rk3326_qgo` under PX30_Android10 directory with `PX30_Android10`

The specific changes are as below:





GMS certification

If need to pass GMS for Android 10.0, need to download GMS package and put in vendor directory.

1. Through RK manifest (**Recommend**, one key deployment, including RK modification, reducing various GTS issues)

- Acquire GMS package
  Firstly need to contact FAE to acquire GMS package access(**should have MADA**), after approved, use the following command to switch and re-sync the source code:

```
.repo/repo/repo init -m Android10_Express.xml
.repo/repo/repo sync -c
```

- Open to configure GMS

```
Configure in device/rockchip/rkxxxx/BoardConfig.mk:
# Google Service and frp overlay


BUILD_WITH_GOOGLE_MARKET := true
BUILD_WITH_GOOGLE_FRP := true
BUILD_WITH_GOOGLE_GMS_EXPRESS := false (this should be set true for Express
customers, for funding customers, please refer to GMS document to modify)
```

2. Acquire GMS package from lab or other channel by self (**Not recommend**)
   After acquiring, put in vendor directory, now the structure of vendor is as below:

```
vendor$ ls
partner_gms  partner_modules (original directory: rockchip  widevine )
```

- Open to configure GMS

```
Configure by self, need to compile mainline modules and GMS package app. Please
merge the following patches to suit the Rockchip GMS environment:
RKDocs/android/patches/gms/rockchip_gms_*.tar.gz
```

Kernel dts instruction

Create new product dts

You can select the corresponding dts according to the configuration in the following table as reference to create new product dts.

Soc	PMIC	DDR	Type	Model	DTS
RK3326	RK817	DDR3	development board	evb	rk3326-evb-lp3-v10-avb
PX30	RK809	DDR3	development board	evb	px30-evb-ddr3-v10-avb
RK3326	RK817	DDR3	tablet	device	rk3326-863-lp3-v10-rkisp1
RK3399	RK808	LPDDR3	development board	excavator	rk3399-sapphire-excavator-edp-avb
RK3399	RK809	LPDDR4	development board	IND evb	rk3399-evb-ind-lpddr4-android-avb

Soc	PMIC	DDR	Type	Model	DTS
RK3399	RK808	LPDDR4	tablet	BQ25703 two batteries	rk3399-tve1030g-avb
RK3399	RK818	LPDDR3	tablet	edp panel	rk3399-mid-818-android
RK3126C	RK816	DDR3	tablet	device	rk3126-m88
RK3368	RK808	LPDDR3	development board	evb	rk3368-808-evb
RK3288	RK808	LPDDR3	development board	evb	rk3288-evb-android-rk808-edp-avb

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.pd
f
```

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
```

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.
- Wifi stress test: including WiFi 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 in 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

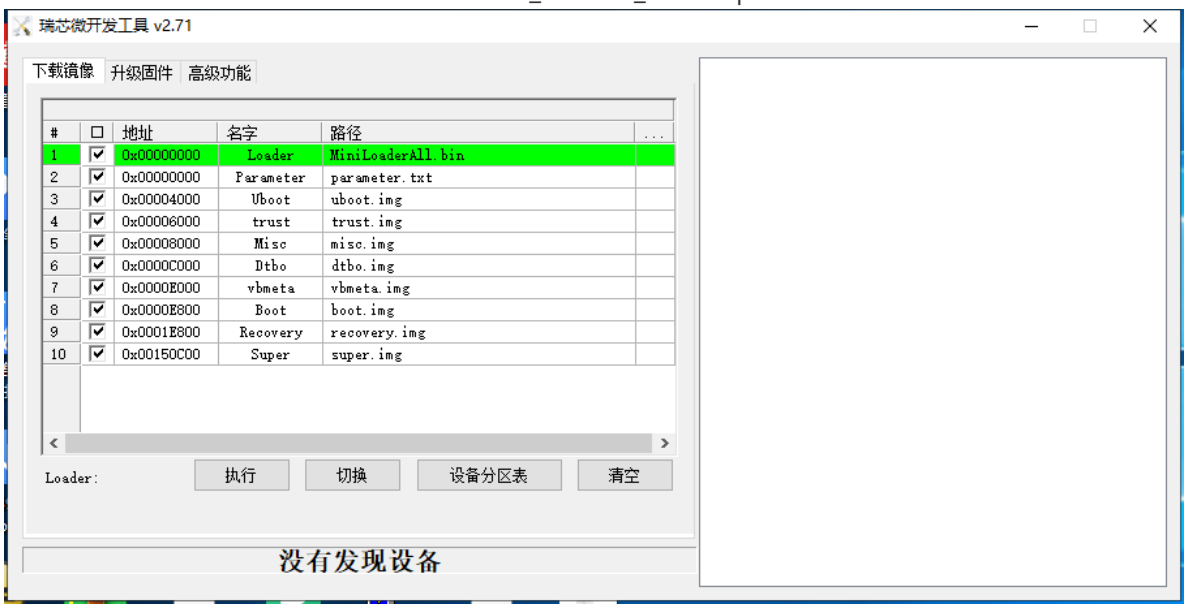
Rockchip USB driver install package includes ADB and image flashing driver

```
RKTools\windows\DriverAssitant_v4.5.zip
```

Development flashing tool

Windows version

RKTools/windows/AndroidTool/AndroidTool_Release_v2.71.zip



Linux version

RKTools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool_v1.43.zip

```
Linux_Upgrade_Tool_v1.43$ 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 RK3288/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”
```

或者 :

```
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 Android10.0 is: develop-4.19, u-boot branch
is next-dev branch
```

How to acquire the corresponding RK release version for current SDK

Rockchip Android10.0 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" revi
sion="refs/tags/android-10.0-mid-rkr2">
```

Means RK version is android-10.0-mid-rkr2

- 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-10.0-mid-rkr1
android-10.0-mid-rkr2
develop-4.4-20190201
```

RK version is incremental with the format of android-10.0-mid-rkrxx, so current latest tag is android-10.0-mid-rkr2

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.

Replace uboot and kernel logo picture

uboot and kernel logo are the first and second logo picture displayed during bootup, and they can be changed according to the product requirement.
uboot logo source file: `kernel/logo.bmp`
kernel logo source file: `kernel/logo_kernel.bmp`
If need to change the picture, just use the bmp with the same name to replace, and re-compile kernel. The compiled file is in boot.img.
Note: Logo picture size currently only supports to 8M with 8, 16, 24, 32bit bmp format.

Power off charging and low battery precharging

Power off charging and low battery precharging can be configured in dts, as shown below:

```
charge-animation {
                compatible = "rockchip,uboot-charge";
                rockchip,uboot-charge-on = <1>;
                rockchip,android-charge-on = <0>;
                rockchip,uboot-low-power-voltage = <3400>;
                rockchip,screen-on-voltage = <3500>;
                status = "okay";
        };
```

Note:
rockchip,uboot-charge-on: uboot power off charging is mutually exclusive with android power off charging
rockchip,android-charge-on: android power off charging is mutually exclusive with uboot power off charging
rockchip,uboot-low-power-voltage: configure the voltage for low battery precharging to boot, it can be configured according to the actual requirement
rockchip,screen-on-voltage: configure the voltage for low battery precharging to light the panel, it can be configured according to the actual requirement

Uboot charging logo package and replace

Charging logo path, you can directly replace with the file with the same name, and the format should be the same as original file.

```
u-boot/tools/images/
├── battery_0.bmp
├── battery_1.bmp
├── battery_2.bmp
├── battery_3.bmp
├── battery_4.bmp
├── battery_5.bmp
└── battery_fail.bmp
```

If uboot charging is enabled, but there is no charging logo displaed, maybe it is because the picture is not packaged into resource.img. You can package per the following command:

```
cd u-boot
./scripts/pack_resource.sh ../kernel/resource.img
cp resource.img ../kernel/resource.img
```

After executing the above command, uboot charging logo will be packaged into resource.img in kernel directory. Now need to re-package resource.img into boot.img. You can execute ./mkiamge.sh in android root directory, and then flash boot.img under rockdev/.

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
```

A/B system configuration

A/B system means there are two sets of system A and B workable on the device. You can think one is system partition, and the other is backup partition. A/B is disabled by default. If need to enable, configure as below in BoardConfig.mk of the corresponding chipset（take RK3326 as example）:

```
vim device/rockchip/rk3326_q/BoardConfig.mk
#AB image definition
-BOARD_USES_AB_IMAGE := false
+BOARD_USES_AB_IMAGE := 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
```

SD card boot function

Precondition

- The device doesn't have EMMC or NAND memory, if does, need to erase flash first.
- SD card function should be enabled, if SD card is disabled by default, need to configure in dts first to enable it, such as RK3326.

Code modification

Take RK3399 as example

Android part

```
device/rockchip/rk3399$
diff --git a/rk3399_Android10/rk3399_Android10.mk
b/rk3399_Android10/rk3399_Android10.mk
index d00e0a3..c031ecb 100755
--- a/rk3399_Android10/rk3399_Android10.mk
+++ b/rk3399_Android10/rk3399_Android10.mk
@@ -17,7 +17,7 @@
 # First lunching is Q, api_level is 29
 PRODUCT_SHIPPING_API_LEVEL := 29
 PRODUCT_DTBO_TEMPLATE := $(LOCAL_PATH)/dt-overlay.in
-PRODUCT_BOOT_DEVICE := fe330000.sdhci
+PRODUCT_BOOT_DEVICE := fe330000.sdhci,fe320000.dwmmc
 include device/rockchip/common/build/rockchip/DynamicPartitions.mk
 include device/rockchip/common/BoardConfig.mk
 $(call inherit-product, $(SRC_TARGET_DIR)/product/full_base.mk)
```

Kernel part

Add supports-emmc field in sdmmc node of dts as below:

```
&sdmmc {
        clock-frequency = <150000000>;
        clock-freq-min-max = <100000 150000000>;
        supports-sd;
        bus-width = <4>;
        cap-mmc-highspeed;
        cap-sd-highspeed;
        disable-wp;
        supports-emmc;
        num-slots = <1>;
        sd-uhs-sdr104;
        vmmc-supply = <&vcc_sd>;
        vqmmc-supply = <&vccio_sd>;
        pinctrl-names = "default";
        pinctrl-0 = <&sdmmc_clk &sdmmc_cmd &sdmmc_cd &sdmmc_bus4>;
        status = "okay";
};
```

Tool

```
SDDiskTool_v1.59
```

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"
review="https://10.10.10.29" />;
```

for foreign customers: (Google mirror source)

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

GTVS update method

Enter vendor directory of SDK project, then execute

```
rsync -av --exclude '*.git' chenwei@10.10.10.165:/home/chenwei/google_gtvs .
```

As the directory is not consistent, need to adjust overlay location in gms.mk.sample

```
 # Add gms_tv specific overlay
-PRODUCT_PACKAGE_OVERLAYS += vendor/google/products/gms_tv_overlay
+PRODUCT_PACKAGE_OVERLAYS += vendor/google_gtvs/overlays/products/gms_tv_overlay

 # Overlay for GMS devices
 $(call inherit-product, device/sample/products/backup_overlay.mk)
 $(call inherit-product, device/sample/products/location_overlay.mk)
-PRODUCT_PACKAGE_OVERLAYS += vendor/google/products/gms_overlay
+PRODUCT_PACKAGE_OVERLAYS += vendor/google_gtvs/overlays/products/gms_overlay

 # Overrides
 PRODUCT_PROPERTY_OVERRIDES += \
```

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 power down abnormally. The modification method is as below:
Take RK3399_Android10 as example:

```
device/rockchip/common$ git diff
diff --git a/scripts/fstab_tools/fstab.in b/scripts/fstab_tools/fstab.in
index 266531a..52453ea 100755
--- a/scripts/fstab_tools/fstab.in
+++ b/scripts/fstab_tools/fstab.in
@@ -16,6 +16,6 @@ ${_block_prefix}product /product  ext4 ro,barrier=1
${_flags},first_stage_mount
 # For sdmmc
 /devices/platform/${_sdmmc_device}/mmc_host*        auto  auto    defaults
   voldmanaged=sdcard1:auto,encryptable=userdata
 #  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,fsync_mode=nobarrier
latemount,wait,check,fileencryption=software,quota,formattable,reservedsize=128M
,checkpoint=fs
+#/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065,fsync_mode=nobarrier
latemount,wait,check,fileencryption=software,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
wait,formattable,check,fileencryption=software,quota,reservedsize=128M
+/dev/block/by-name/userdata    /data      ext4
discard,noatime,nosuid,nodev,noauto_da_alloc,data=ordered,user_xattr,barrier=1
wait,formattable,check,fileencryption=software,quota,reservedsize=128M
```

```
device/rockchip/rk3399$ git diff
--- a/device.mk
+++ b/device.mk
@@ -27,7 +27,7 @@ PRODUCT_PACKAGES += \
     libion

 #enable this for support f2fs with data partion
-BOARD_USERDATAIMAGE_FILE_SYSTEM_TYPE := f2fs
+BOARD_USERDATAIMAGE_FILE_SYSTEM_TYPE := ext4

 # used for fstab_generator, sdmmc controller address
 PRODUCT_SDMMC_DEVICE := fe320000.dwmmc
diff --git a/rk3399_Android10/recovery.fstab b/rk3399_Android10/recovery.fstab
index 7532217..cf789ac 100755
--- a/rk3399_Android10/recovery.fstab
+++ b/rk3399_Android10/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
```

root function

The patch of root function:

```
RKDocs/android/patches/box/rootservice_for_android10.rar
```

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.

OTA upgrade from Android 9.0 to Android 10.0

Android10.0 SDK supports OTA upgrade from Android 9.0 version to Android10.0. Refer to the document for more details:

```
RKDocs\android\Rockchip_Introduction_OTA_from_Android9.0_to_Android10.0_CN&EN.pd
f
```

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
  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), ot Miniloader is too old to support transmission.

Partition change

Change the partition size

The partition size is defined in parameter file, which is in the product directory, for example, parameter file path of rk3326_qgo product is as below:

```
  device/rockchip/rk3326/rk3326_qgo/parameter.txt
```

Please refer to the document for the introduction of parameter file:

```
  RKDocs/common/RKTools manuals/Rockchip-Parameter-File-Format-Version1.4-CN.pdf
```

Add partition

The step to add new partition:

- Refer to the above instruction to add the partition size and address information in parameter file.
- Add the partition loading information in fstab. fstab file default path（some product is in the product directory）is：

  ```
  device/rockchip/common/scripts/fstab_tools/fstab.in
  ```

- Add the partition information in recovery.fstab. recovery.fstab file path is in the specific product directory, such as rk3326_qgo:

  ```
  device/rockchip/rk3326/rk3326_qgo/recovery.fstab
  ```

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
```

Flashing issue after using build.sh for PX30

Description: The FlashingTools prompts an error when flashing update.img compiled from build.sh: "Check Chip Fail!"

Solution: `cd RKTools` and patch the following CL:

```
diff --git a/linux/Linux_Pack_Firmware/rockdev/mkupdate_rk3326.sh
b/linux/Linux_Pack_Firmware/rockdev/mkupdate_rk3326.sh
index 7df28a9..722cd9d 100755
--- a/linux/Linux_Pack_Firmware/rockdev/mkupdate_rk3326.sh
+++ b/linux/Linux_Pack_Firmware/rockdev/mkupdate_rk3326.sh
@@ -15,7 +15,7 @@ if [ ! -f "package-file" ]; then
 #      pause
 fi
 ./afptool -pack ./ Image/update.img || pause
-./rkImageMaker -RK3326 Image/MiniLoaderAll.bin Image/update.img update.img -
os_type:androidos || pause
+./rkImageMaker -RKPX30 Image/MiniLoaderAll.bin Image/update.img update.img -
os_type:androidos || pause
 echo "Making update.img OK."
 #echo "Press any key to quit:"
 #read -n1 -s key
```

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 not consistent.
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
```

How to disable disk encryption for the userdata partition

userdata partition encryption configuration in fstb.in, remove the encryption as follow:(take rk3326 as an example)

```
diff --git a/rk3326_q/fstab.in b/rk3326_q/fstab.in
index 4d2828e..0e4ca04 100755
--- a/rk3326_q/fstab.in
+++ b/rk3326_q/fstab.in
@@ -15,4 +15,4 @@ ${_block_prefix}odm    /odm    ext4 ro,barrier=1
${_flags},first_stage_mount
 # For sdmmc
 /devices/platform/ff370000.dwmmc/mmc_host*        auto  auto    defaults
voldmanaged=sdcard1:auto,encryptable=userdata
 #  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,fsync_mode=nobarrier
latemount,wait,check,fileencryption=software,quota,formattable,reservedsize=128M
,checkpoint=fs
+/dev/block/by-name/userdata /data f2fs
noatime,nosuid,nodev,discard,reserve_root=32768,resgid=1065,fsync_mode=nobarrier
latemount,wait,check,quota,formattable,reservedsize=128M
```