Rockchip Android13 GKI开发指南

文件标识：RK-KF-YF-751

发布版本：V1.1.0

日期：2023-03-23

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按"现状"提供，瑞芯微电子股份有限公司（"本公司"，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

"Rockchip"、"瑞芯微"、"瑞芯"均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

概述

本文介绍Android13的GKI开发的流程和注意点。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

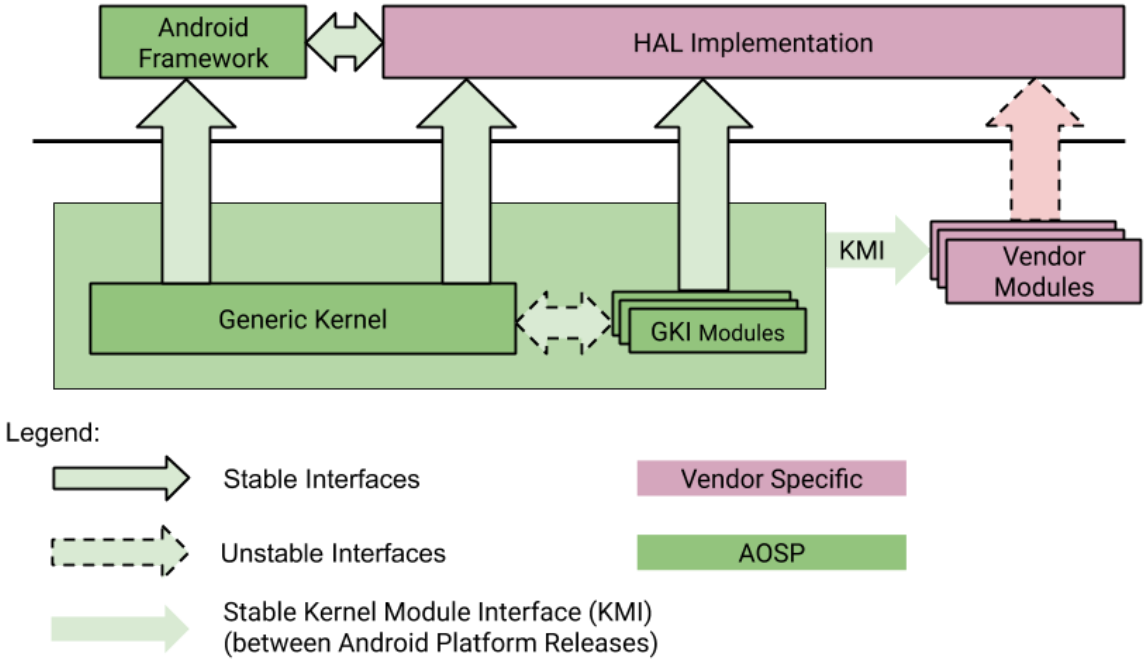
版本号	作者	修改日期	修改说明
V1.0.0	吴良清	2023-03-23	初始版本

目录

1 GKI介绍

1.1 什么是GKI

GKI：Generic Kernel Image 通用内核映像。
Android13 GMS和EDLA认证的一个难点是google强制要求要支持GKI。GKI通用内核映像，是google为了解决内核碎片化的问题，而设计的通过提供统一核心内核并将SoC和板级驱动从核心内核移至可加载模块中。核心内核为驱动模块提供了稳定的内核模块接口，模块驱动和核心内核可以独立进行更新。内核接口可以通过upstream的方式进行扩展。 Soc和板级厂商在驱动开发时需要使用已经定义的内核接口，如果要新加核心内核接口需要提交给google，这个周期会比较长，所以要提前做好开发准备。



1.2 什么产品需要使用GKI

- 使用Android13且需要过GMS认证和EDLA认证的产品
- 使用Android12 的RK3588和RK3588S的需要过GMS认证和EDLA认证的产品
- 不过GMS认证和EDLA认证的产品不强制要求使用GKI

1.3 GKI和非GKI的差别

- 通用内核boot.img

GKI	非GKI
由google定期发布boot.img，代码不能自己修改	由RK提供内核源码编译，可以自由修改

- 驱动模块

GKI	非GKI
以KO的形式加载，调用的内核接口必需是google发布的boot.img里面包含的	内嵌在boot中，由RK提供内核源码编译，可以自由修改和添加内核接口

- kernel代码

GKI	非GKI
RK发布的kernel源码仅用于编译驱动模块的KO	RK发布的kernel源码用于完整的内核和模块驱动的编译，模块以.o的形式内嵌编译

- uboot支持head4
- 分区差异
  GKI增加vendor_boot、init_boot、resource分区
- 启用AB分区

2 Rockchip Android13 GKI的适配情况

kernel版本是5.10

芯片	是否完成适配
RK3588/RK3588S	已适配
RK3568/RK3566	已适配
RK3326/RK3326S	已适配
PX30/PX30S	已适配
RK3399	正在适配

3 Google upstream kernel下载及编译

Google的boot.img是定期发布，时间间隔比较长。 我们可以下载google的upstream的kernel源码自己编译boot.img进行验证和debug。

Google Upstream kernel下载链接：

```
repo init -u https://android.googlesource.com/kernel/manifest -b common-
android13-5.10
```

需要链接google服务器下载

编译

```
tools/bazel run --config=fast //common:kernel_aarch64_dist -- --dist_dir=./out
```

生成boot.img

```
out/boot.img
```

4 Rockchip SDK中GKI相关目录介绍

- kernel KO文件路径

```
mkcombinedroot/vendor_ramdisk/lib/modules/
```

- Google boot.img路径

```
mkcombinedroot/prebuilts/boot-5.10.img
```

- KO文件加载顺序配置文件

```
mkcombinedroot$ tree res/
res/
├── board
|   ├── px30-mini-evb-ddr3-v11-avb.load
|   ├── rk3326-863-lp3-v10-rkisp1.load
|   ├── rk3326-evb-lp3-v10-avb.load
|   ├── rk3399-evb-ind-lpddr4-android-avb.load
```

```
|   ├── rk3399-evb-ind-lpddr4-v13-android-avb.load
|   ├── rk3562-evb1-lp4x-v10.load
|   ├── rk3562-rk817-tablet-v10.load
|   ├── rk3566-evb2-lp4x-v10.load
|   ├── rk3566-rk817-tablet.load
|   ├── rk3568-evb1-ddr4-v10.load
|   ├── rk3588-evb1-lp4-v10.load
|   ├── rk3588-evb7-lp4-v10.load
|   └── rk3588s-tablet-v10.load
├── bootconfig
├── debug_list.load
├── file_contexts.bin
├── ramdisk_modules.load
├── recovery_gki.mk
├── soc
|   ├── rk3326
|   |   └── vendor_ramdisk_modules.load
|   ├── rk3399
|   |   └── vendor_ramdisk_modules.load
|   ├── rk3562
|   |   └── vendor_ramdisk_modules.load
|   ├── rk356x
|   |   ├── vendor_ramdisk_modules.load
|   └── rk3588
|       ├── vendor_ramdisk_modules.load
├── vendor_gki.mk
├── vendor_image_info.txt
├── vendor_modules.load
├── vendor_ramdisk_gki.mk
```

其中：
res/soc/ 下面的是芯片平台相关的ko加载
res/board/下面是板级相关的ko加载

- GPU mali库的路径
  GPU的mali库是单独编译在vendor.img中，源文件路径在

```
RK3588：
vendor/rockchip/common/gpu/MaliG610/lib/modules/bifrost_kbase.ko
RK356X/RK3562：
vendor/rockchip/common/gpu/MaliG52/lib/modules/bifrost_kbase.ko
RK3326/RK3326-S：
PX30/PX30-S：
vendor/rockchip/common/gpu/MaliTDVx/lib/modules/mali_kbase.ko
RK3399：
vendor/rockchip/common/gpu/MaliT860/lib/modules/mali_kbase.ko
```

5 Rockchip GKI编译

5.1 代码修改

GKI需要打开AB系统才能使用，具体代码修改如下：

1. uboot需要打开AB配置

```
~/a2_Android13_sdk/u-boot$ git diff
diff --git a/configs/rk3568_defconfig b/configs/rk3568_defconfig
index fbd9820acc..e23e438792 100644
--- a/configs/rk3588_defconfig
+++ b/configs/rk3588_defconfig
@@ -207,6 +207,7 @@ CONFIG_RSA_N_SIZE=0x200
CONFIG_RSA_E_SIZE=0x10
CONFIG_RSA_C_SIZE=0x20
CONFIG_SHA512=y
CONFIG_LZ4=y
CONFIG_LZMA=y
CONFIG_SPL_GZIP=y
@@ -220,3 +221,4 @@ CONFIG_RK_AVB_LIBAVB_USER=y
CONFIG_OPTEE_CLIENT=y
CONFIG_OPTEE_V2=y
CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y
+CONFIG_ANDROID_AB=y
```

2. 增加板级的KO load文件
   Rockchip的GKI框架里面加载KO的load文件有区分芯片平台驱动和板级驱动，所以在开发新产品的时候需要增加板级驱动的KO load文件，load文件以device下面产品目录中定义PRODUCT_KERNEL_DTS的dts的名字命名，并且保存在mkcombinedroot/res/board/下面，如：

```
wlq@sys2_206:~/a0_Android13_gki$ tree mkcombinedroot/res/board/
mkcombinedroot/res/board/
├── px30-mini-evb-ddr3-v11-avb.load
├── rk3326-863-lp3-v10-rkisp1.load
├── rk3326-evb-lp3-v10-avb.load
├── rk3399-evb-ind-lpddr4-android-avb.load
├── rk3399-evb-ind-lpddr4-v13-android-avb.load
├── rk3562-evb1-lp4x-v10.load
├── rk3562-rk817-tablet-v10.load
├── rk3566-evb2-lp4x-v10.load
├── rk3566-rk817-tablet.load
├── rk3568-evb1-ddr4-v10.load
├── rk3588-evb1-lp4-v10.load
├── rk3588-evb7-lp4-v10.load
└── rk3588s-tablet-v10.load
```

板级的KO load文件里面放的是板级的驱动模块，如：触摸屏、camera、sensor等芯片平台以为的驱动模块，如：

```
wlq@sys2_206:~/a0_Android13_gki$ cat mkcombinedroot/res/board/rk3588-evb1-lp4-v10.load
cw2015_battery.ko
imx415.ko
ov50c40.ko
ov13855.ko
gt1x-ts.ko
snd-soc-es8323.ko
```

3. Android的device产品目录下配置GKI选项

```
~/a2_Android13_sdk/device/rockchip/rk3588$ git diff
diff --git a/rk3588_t/BoardConfig.mk b/rk3588_t/BoardConfig.mk
```

```
old mode 100644
new mode 100755
index 50da541..06da5f3
--- a/rk3588_t/BoardConfig.mk
+++ b/rk3588_t/BoardConfig.mk
@@ -15,10 +15,21 @@
 #
 include device/rockchip/rk3588/BoardConfig.mk
 BUILD_WITH_GO_OPT := false
+BOARD_BUILD_GKI := true

-# AB image definition
-BOARD_USES_AB_IMAGE := false
-BOARD_ROCKCHIP_VIRTUAL_AB_ENABLE := false
+ifeq ($(strip $(BOARD_BUILD_GKI)), true)
+    #for gki
+    # AB image definition
+    BOARD_USES_AB_IMAGE := true
+    BOARD_ROCKCHIP_VIRTUAL_AB_ENABLE := true
+    PRODUCT_KERNEL_CONFIG := gki_defconfig rockchip_gki.config
+else
+    BOARD_ROCKCHIP_VIRTUAL_AB_ENABLE := false
+    BOARD_USES_AB_IMAGE := false
+    PRODUCT_KERNEL_CONFIG := rockchip_defconfig android-13.config
+endif

 BOARD_GRAVITY_SENSOR_SUPPORT := true
 BOARD_COMPASS_SENSOR_SUPPORT := true
@@ -26,14 +37,21 @@ BOARD_SENSOR_COMPASS_AK8963-64 := true
 BOARD_GYROSCOPE_SENSOR_SUPPORT := true
 BOARD_PROXIMITY_SENSOR_SUPPORT := true
 BOARD_LIGHT_SENSOR_SUPPORT := true
 ifeq ($(strip $(BOARD_USES_AB_IMAGE)), true)
     include device/rockchip/common/BoardConfig_AB.mk
     TARGET_RECOVERY_FSTAB := device/rockchip/rk3588/rk3588_t/recovery.fstab_AB
 endif
-
+ifeq ($(strip $(BOARD_BUILD_GKI)), true)
+    #for gki
+    BOARD_SUPER_PARTITION_SIZE := 4294967296
+    BOARD_ROCKCHIP_DYNAMIC_PARTITIONS_SIZE := $(shell expr
$(BOARD_SUPER_PARTITION_SIZE) - 4194304)
+endif
 PRODUCT_UBOOT_CONFIG := rk3588
 PRODUCT_KERNEL_DTS := rk3588-evb1-lp4-v10
 BOARD_GSENSOR_MXC6655XA_SUPPORT := true
 BOARD_CAMERA_SUPPORT_EXT := true
```

5.2 编译

完整编译方式与非GKI的一样

```
source  build/envsetup.sh
lunch rk3588_t-userdebug
./build.sh -ACUKup
```

注意：这里编译的kernel只是为了编译出resource.img，kernel源码部分都是使用mkcombinedroot/vendor_ramdisk/lib/modules/下的ko文件直接打包成vendor_boot.img。内核部分使用的是google发布的boot.img，具体路径在mkcombinedroot/prebuilts/boot-5.10.img

编译完可以直接烧写 `rockdev/Image-rk3588_t/update.img`

在调试阶段也支持单独编译vendor_boot.img
编译命令：

```
make installclean;make vendorbootimage -j12
```

编译完可以直接烧写

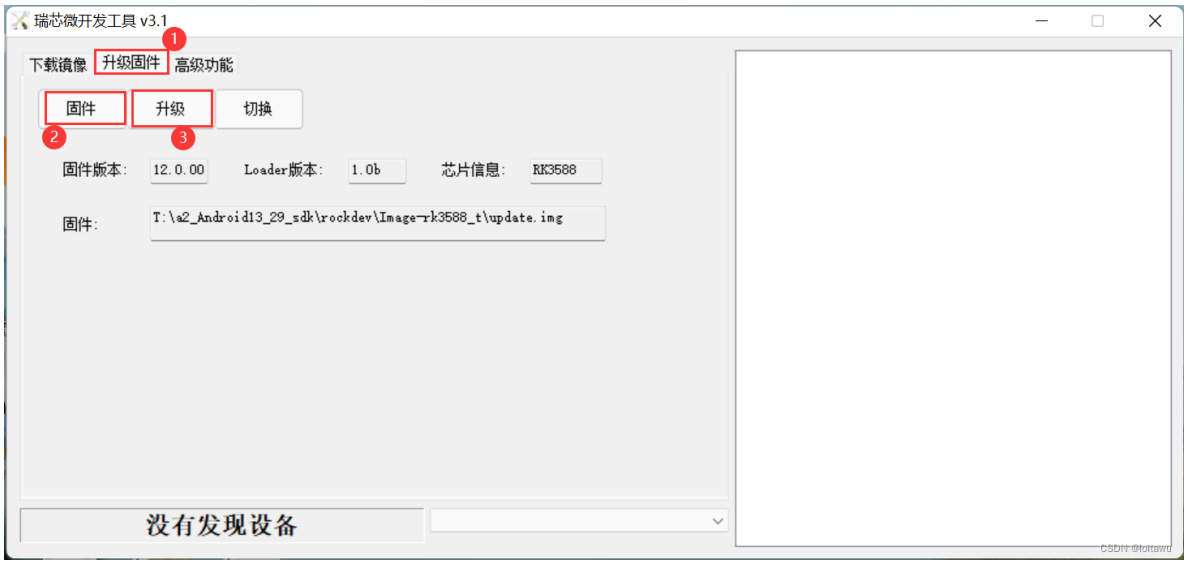
```
out/target/product/rk3588_t/vendor_boot.img
```

5.3 固件烧写

固件烧写分2中方式：

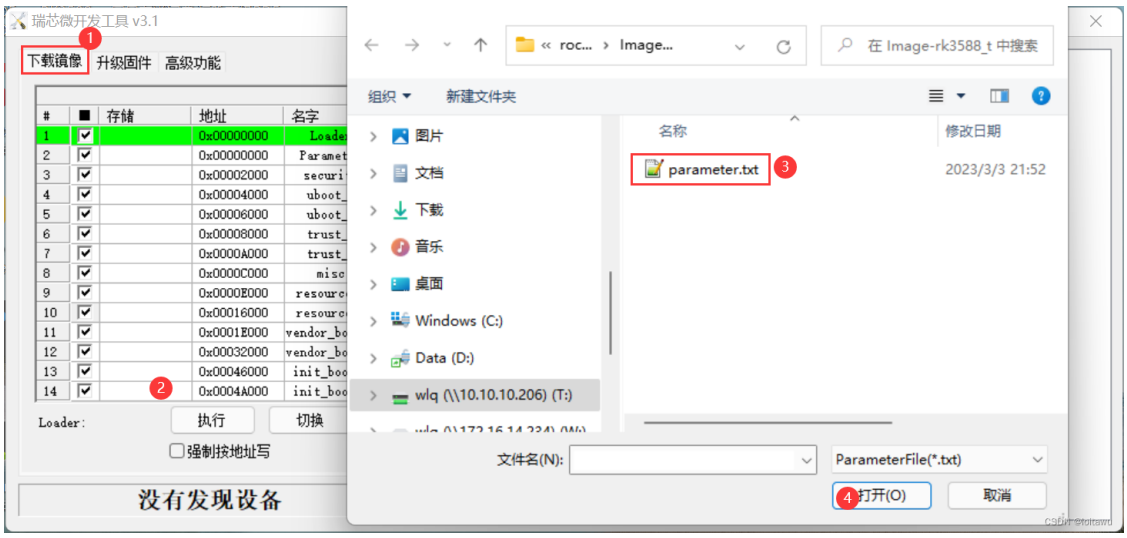
- 完整包update.img
  固件路径

```
rockdev/Image-rk3588_t/update.img
```



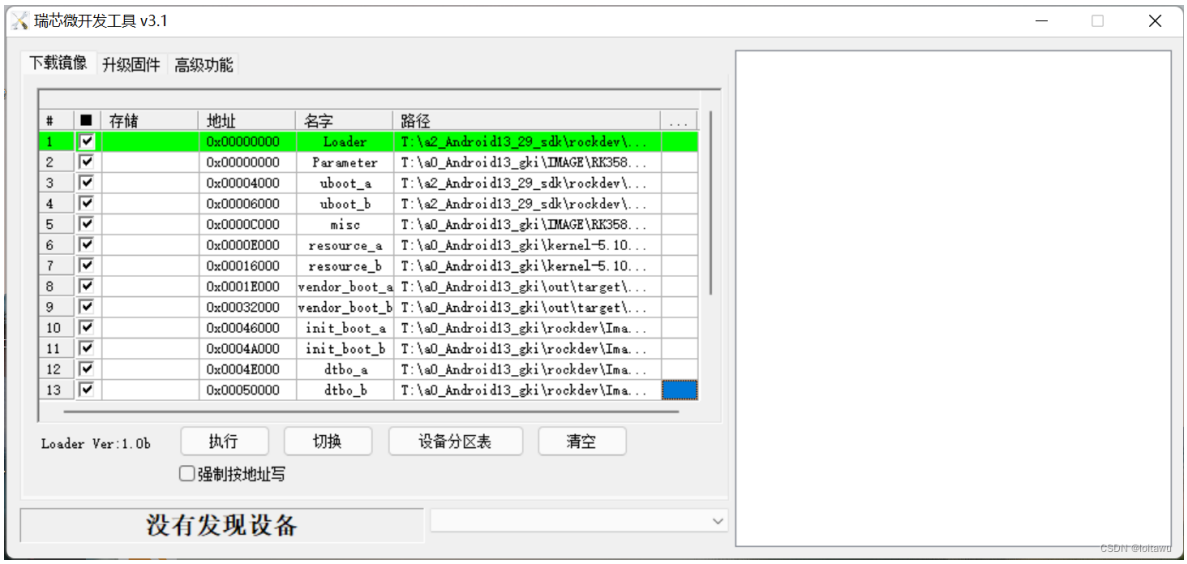
可以通过瑞芯微开发工具烧写

- 散包烧写
  首先导入配置文件，方法是在工具 空白处右键-导入配置-选择导入txt文件-选择parameter.txt



然后依次选择rockdev/Image-rk3588_t/下对应的img文件进行烧写，分区A和B导入的固件是同一个

```
rockdev/Image-rk3588_t
├── baseparameter.img
├── boot.img
├── dtbo.img
├── init_boot.img
├── MiniLoaderAll.bin
├── misc.img
├── parameter.txt
├── resource.img
├── super.img
├── uboot.img
├── update.img
├── vbmeta.img
└── vendor_boot.img
```



6 KO编译及修改

6.1 添加新的模块驱动的方法

1. 将驱动代码放到kernel-5.10对应的目录下，这里以新加触摸屏驱动gt1x为例进行说明。
   将gt1x的驱动放在`drivers/input/touchscreen/`下面，并添加对应的`Makefile`和`Kconfig`，这里按kernel的标准方式进行操作；

2. 增加一个自己的config文件， 在`arch/arm64/configs/`下新建一个`xxx_gki.config`，并将`CONFIG_TOUCHSCREEN_GT1X=m`(m表示编译为ko)添加到`xxx_gki.config`中；
3. 将ko文件名添加到板级的load文件中， load文件在SDK的mkcombinedroot/res/board目录下，如下

`.load`文件名称	对应分区	makefile解析	加载时间
vendor_ramdisk_modules.load	vendor_boot	vendor_ramdisk_gki.mk	ramdisk init阶段
vendor_modules.load	vendor	vendor_gki.mk	android启动时
recovery_modules.load	recovery	recovery_gki.mk	recovery阶段

这里以RK3588-EVB1的板级配置为例进行说明：
触摸屏驱动要在init阶段加载所以加到`rk3588-evb1-lp4-v10.load`中

```
echo "gt1x-ts.ko" >> res/board/rk3588-evb1-lp4-v10.load
```

注意 1：.load文件关乎驱动的加载顺序，请不要修改原有顺序，仅在需要时添加自己的驱动名称，否则可能会导致系统无法启动！！！

注意 2：如果使用A/B系统，请务必保证`vendor_ramdisk_modules.load`和`recovery_modules.load`文件内容一致，否则会导致无法启动！代码默认使用软链接将二者链接起来，请不要自己修改！！！

注意 3：如果是在android启动的时候加载的ko可以放在`vendor_modules.load`中，但需要注意：vendor下的ko不会被系统主动加载！如果仅需要在开机阶段自动加载，可以使用Rockchip提供的默认加载脚本`init.insmod.sh`，该脚本会自动加载`device/rockchip/common/rootdir/init.insmod.cfg`配置中的所有ko。

4. 编译
   进到kernel-5.10目录下进行ko文件编译

配置clang编译链（编译链版本请参考`build.sh`中的配置）

```
export PATH=../prebuilts/clang/host/linux-x86/clang-r450784d/bin:$PATH
```

```
make CROSS_COMPILE=aarch64-linux-gnu- LLVM=1 LLVM_IAS=1 ARCH=arm64 gki_defconfig
rockchip_gki.config xxx_gki.config && make CROSS_COMPILE=aarch64-linux-gnu-
LLVM=1 LLVM_IAS=1 ARCH=arm64 rk3588s-evb8-lp4x-v10.img -j32
```

5. 编译kernel后将编译生产的gt1x-ts.ko文件拷贝到
   mkcombinedroot/vendor_ramdisk/lib/modules/下面

```
find -name gt1x-ts.ko | xargs -I {} cp {}
../mkcombinedroot/vendor_ramdisk/lib/modules/
```

6. 编译vendor_boot.img
   在工程根目录下编译vendor_boot.img，命令如下。 这一步是将KO文件打包到vendor_boot.img，在降vendor_boot.img烧写到机器中。

```
make installclean;make vendorbootimage -j12
```

单独烧写vendor_boot.img，编译完的vendor_boot.img路径如下：

```
out/target/product/rk3588_t/vendor_boot.img
```

7. 验证

- 烧写`out/target/product/rk3588_t/vendor_boot.img`文件到机器中开机验证
- 如果是放在vendor分区的ko可以在系统起来后直接push到机器内的vendor分区中，手动挂载进行验证
- 如果有涉及到dts的修改，需要烧写kernel-5.10下的`resource.img`

附：AOSP定义的各类ko加载阶段

Boot mode	Storage	Display	Keypad	Battery	PMIC	TP	NFC/Wi-Fi/BT	Sensors	Camera
Recovery	Y	Y	Y	Y	Y	N	N	N	N
Charger	Y	Y	Y	Y	Y	N	N	N	N
Android	Y	Y	Y	Y	Y	Y	Y	Y	Y

6.2 调试ko方法

1. 在kernel-5.10目录下修改对应ko的驱动源码
2. 使用如下命令进行ko编译

- 配置clang编译链（编译链版本请参考`build.sh`中的配置）

```
export PATH=../prebuilts/clang/host/linux-x86/clang-r450784d/bin:$PATH
```

- 编译ko

```
make CROSS_COMPILE=aarch64-linux-gnu- LLVM=1 LLVM_IAS=1 ARCH=arm64
gki_defconfig rockchip_gki.config xxx_gki.config && make
CROSS_COMPILE=aarch64-linux-gnu- LLVM=1 LLVM_IAS=1 ARCH=arm64 rk3588s-evb8-
lp4x-v10.img -j32
```

3. 拷贝KO文件到`mkcombinedroot/vendor_ramdisk/lib/modules/`
4. 编译vendor_boot.img
   在工程根目录下编译vendor_boot.img，命令如下。 这一步是将KO文件打包到vendor_boot.img，在降vendor_boot.img烧写到机器中。

```
    make installclean;make vendorbootimage -j12
```

5. 验证
   - 烧写`out/target/product/rk3588_t/vendor_boot.img`文件到机器中开机验证
   - 如果是放在vendor分区的ko可以在系统起来后直接push到机器内的vendor分区中，手动挂载进行验证
   - 如果有涉及到dts的修改，需要烧写kernel-5.10下的`resource.img`
6. 调试完成后，将`vendor_ramdisk/lib/modules`的ko文件（被脚本自动拷贝）进行提交

有关打包工具的详细说明，请参考：`mkcombinedroot/README`

注意：kernel编译ko的时候如果有修改了config，则编译中间会卡住很长一段时间，这是在做编译优化，根据编译服务器硬件配置不同优化的速度也不同，即卡住的时间也不同。 所以这个卡住是正常现象。

7 开机log确认

7.1 uboot阶段

内容	header版本
vendor_ramdisk(v-ramdisk)	V3+
bootconfig	V4+

```
## Booting Android Image at 0x003ff000 ...
Kernel: 0x00400000 - 0x03088ffc (45604 KiB)
v-ramdisk:  0x0a200000 - 0x0a6944c8 (4690 KiB)
ramdisk:    0x0a6944c8 - 0x0a7e54df (1349 KiB)
bootconfig: 0x0a7e54df - 0x0a7e559c (1 KiB)
bootparams: 0x0a7e559c - 0x0a7e759c
```

7.2 Android阶段

GKI版本： `Linux version 5.10.117-android13-9-00037-gbc08447eb7bd`

```
[    0.000000][    T0] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[    0.000000][    T0] Linux version 5.10.117-android12-9-00037-gbc08447eb7bd
(build-user@build-host) (Android (7284624, based on r416183b) clang version
12.0.5 (https://android.googlesource.com/toolchain/llvm-project
c935d99d7cf2016289302412d708641d52d2f7ee), LLD 12.0.5
(/buildbot/src/android/llvm-toolchai
n/out/llvm-project/lld c935d99d7cf2016289302412d708641d52d2f7ee)) #1 SMP PREEMPT
Thu Aug 25 15:24:20 UTC 2022
```

Kernel command line： Header V4中不能存在androidboot.xxx这一类的命令行参数，这类参数全部在bootconfig中。此类参数可以通过 `cat /proc/bootconfig` 确认。

```
[    0.000000][    T0] Kernel command line: stack_depot_disable=on
kasan.stacktrace=off kvm-arm.mode=protected cgroup_disable=pressure
cgroup.memory=nokme
m storagemedia=emmc console=ttyFIQ0 firmware_class.path=/vendor/etc/firmware
init=/init rootwait ro loop.max_part=7 bootconfig buildvariant=userdebug earl
ycon=uart8250,mmio32,0xfeb50000 irqchip.gicv3_pseudo_nmi=0
```

7.3 KO加载

开始加载ko，可以看到log：

```
[    1.034730][    T1] Run /init as init process
[    1.036190][    T1] init: init first stage started!
[    1.040534][    T1] init: Loading module /lib/modules/io-domain.ko with args
''
[    1.042038][    T1] init: Loaded kernel module /lib/modules/io-domain.ko
```

7.4 KO加载报错

使用了未导出的符号，报错重启：

```
[    0.805736][    T1] cryptodev: Unknown symbol crypto_ahash_final (err -2)
[    0.806383][    T1] cryptodev: Unknown symbol sg_nents (err -2)
[    0.806972][    T1] cryptodev: Unknown symbol crypto_alloc_akcipher (err -2)
[    0.819768][    T1] Kernel panic - not syncing: Attempted to kill init!
exitcode=0x00007f00
```

注： 正常不会出现此问题，参考 `名词解释阶段--ABI进行处理`

7.5 bootcmdline解析出错

错误log

```
Failed to parse bootconfig: Value is redefined at 416.
```

现象：无法开机或者开机进到recovery
原因：cmdline中的字段重复了，导致解析cmdline出错，可以在开机到uboot的时候在串口按住crtl+p就会打印所有的cmdline信息，从打印的cmdline信息中检查哪个字段重复了，然后去代码里面找对应的定义的位置删除对应的字段即可。一般吧cmdline是在device和kernel的dts中定义，可以在这两个目录下搜索该重复的字段即可。

7.6 Mali KO加载失败

Mali KO加载失败表现为无法开机界面显示卡在'Rockchip kernel'的logo，logcat可以看到surfaceflinger crash。

```
04-27 22:45:27.653   366   366 F DEBUG   : *** *** *** *** *** *** *** *** ***
*** *** *** *** *** *** ***
04-27 22:45:27.653   366   366 F DEBUG   : Build fingerprint:
'rockchip/rk3562_t/rk3562_t:13/TQ2A.230305.008.F1/eng.wlq.20230427.101925:userde
bug/release-keys'
04-27 22:45:27.653   366   366 F DEBUG   : Revision: '0'
04-27 22:45:27.653   366   366 F DEBUG   : ABI: 'arm64'
04-27 22:45:27.653   366   366 F DEBUG   : Timestamp: 2023-04-27
22:45:27.509738048+0000
04-27 22:45:27.653   366   366 F DEBUG   : Process uptime: 2s
04-27 22:45:27.653   366   366 F DEBUG   : Cmdline: /system/bin/surfaceflinger
04-27 22:45:27.653   366   366 F DEBUG   : pid: 335, tid: 360, name:
surfaceflinger  >>> /system/bin/surfaceflinger <<<
04-27 22:45:27.653   366   366 F DEBUG   : uid: 1000
04-27 22:45:27.653   366   366 F DEBUG   : tagged_addr_ctrl: 0000000000000001
(PR_TAGGED_ADDR_ENABLE)
04-27 22:45:27.653   366   366 F DEBUG   : signal 6 (SIGABRT), code -1
(SI_QUEUE), fault addr --------
04-27 22:45:27.653   366   366 F DEBUG   : Abort message: 'no suitable EGLConfig
found, giving up'
04-27 22:45:27.653   366   366 F DEBUG   :     x0  0000000000000000  x1
0000000000000168  x2  0000000000000006  x3  000000710899d340
04-27 22:45:27.654   366   366 F DEBUG   :     x4  7568661f2b636d74  x5
7568661f2b636d74  x6  7568661f2b636d74  x7  7f7f7f7f7f7f7f7f
```

```
04-27 22:45:27.654   366   366 F DEBUG   :     x8  00000000000000f0  x9
000000739bcbda00  x10 0000000000000001  x11 000000739bcff6a0
04-27 22:45:27.654   366   366 F DEBUG   :     x12 000000710899d310  x13
0000000000000027  x14 000000710899d4e0  x15 00000000197b1a4f
04-27 22:45:27.654   366   366 F DEBUG   :     x16 000000739bd6dd58  x17
000000739bd48770  x18 0000007108812000  x19 00000000000000ac
04-27 22:45:27.654   366   366 F DEBUG   :     x20 00000000000000b2  x21
000000000000014f  x22 0000000000000168  x23 00000000ffffffff
04-27 22:45:27.654   366   366 F DEBUG   :     x24 b4000071bbca60b0  x25
000000710899dcb0  x26 000000710899dff8  x27 00000000000fe000
04-27 22:45:27.654   366   366 F DEBUG   :     x28 000000710899daf0  x29
000000710899d3c0
04-27 22:45:27.654   366   366 F DEBUG   :     lr  000000739bcef3f4  sp
00000071089ndroid.runtime/lib64/bionic/libc.so (__pthread_start(void*)+208)
(BuildId: e2429c64ab29f2d0ffc5a8f42c0c1b80)
04-27 22:45:27.655   366   366 F DEBUG   :       #09 pc 0000000000054c50
/apex/com.android.runtime/lib64/bionic/libc.so (__start_thread+64) (BuildId:
e2429c64ab29f2d0ffc5a8f42c0c1b80)
```

这个是因为GPU的ko不匹配，需要重新编译GPU的ko文件，并拷贝到vendor/rockchip/common/gpu下面对应的目录中，具体编译方法如下：
修改device下面的产品目录中kernel config配置：PRODUCT_KERNEL_CONFIG := gki_defconfig rockchip_gki.config增加对应芯片的gpu配置，具体如下

```
RK3588：
PRODUCT_KERNEL_CONFIG := gki_defconfig rockchip_gki.config
RK356X/RK3562：
PRODUCT_KERNEL_CONFIG := gki_defconfig rockchip_gki.config rk356x.config
RK3326/RK3326-S：
PX30/PX30-S：
PRODUCT_KERNEL_CONFIG := gki_defconfig rockchip_gki.config rk3326.config
RK3399：
PRODUCT_KERNEL_CONFIG := gki_defconfig rockchip_gki.config rk3399.config

rk3399.config需要按如下修改：
wlq@sys2_206:~/a0_Android13_gki/mkcombinedroot$ git diff configs/
diff --git a/configs/rk3399.config b/configs/rk3399.config
old mode 100644
new mode 100755
index 0d66674..a003ba5
--- a/configs/rk3399.config
+++ b/configs/rk3399.config
@@ -1,4 +1,11 @@
-CONFIG_MALI_MIDGARD=y
+CONFIG_MALI_MIDGARD=m
+CONFIG_MALI_PLATFORM_THIRDPARTY_NAME="rk"
+CONFIG_MALI_PLATFORM_THIRDPARTY=y
+CONFIG_MALI_DEBUG=y
+CONFIG_MALI_DEVFREQ=y
+CONFIG_MALI_DT=y
+CONFIG_MALI_EXPERT=y
+CONFIG_MALI_SHARED_INTERRUPTS=y
```

然后执行./build.sh -CK 编译kernel
编译完成后拷贝对应的mali ko到vendor下面，具体路径可以看上面的章节。

7.7 编译kernel报错

编译错误log：

```
BTF     .btf.vmlinux.bin.o
Segmentation fault (core dumped)
LD      .tmp_vmlinux.kallsyms1
KSYMS   .tmp_vmlinux.kallsyms1.S
AS      .tmp_vmlinux.kallsyms1.S
LD      .tmp_vmlinux.kallsyms2
KSYMS   .tmp_vmlinux.kallsyms2.S
AS      .tmp_vmlinux.kallsyms2.S
LD      vmlinux
BTFIDS  vmlinux
FAILED: load BTF from vmlinux: Unknown error -22Makefile:1293: recipe for target
'vmlinux' failed
make[1]: *** [vmlinux] Error 255
arch/arm64/Makefile:214: recipe for target 'rk3588-evb1-lp4-v10.img' failed
make: *** [rk3588-evb1-lp4-v10.img] Error 2
failed to build some targets (21 seconds)
```

解决方法：

- 下载最新版本pahole
  `git clone https://git.kernel.org/pub/scm/devel/pahole/pahole.git`
- 编译pahole

安装编译依赖库
`sudo apt-get install cmake`
`sudo apt-get install libdw-dev`

如果之前有安装过pahole，需要先卸载
`sudo apt-get --purge remove dwarves`

- 开始编译

ahole目录下执行
`mkdir build`
`cd build/`
`cmake -D__LIB=lib -DBUILD_SHARED_LIBS=OFF ..` 配置静态编译
`sudo make install`

pahole --version 查看版本确认是否安装成功

错误log：

```
  LD      .tmp_vmlinux.kallsyms2
  KSYMS   .tmp_vmlinux.kallsyms2.S
  AS      .tmp_vmlinux.kallsyms2.S
  LD      vmlinux
  BTFIDS  vmlinux
FAILED: load BTF from vmlinux: Unknown error -22Makefile:1293: recipe for target
'vmlinux' failed
make[1]: *** [vmlinux] Error 255
arch/arm64/Makefile:214: recipe for target 'rk3562-evb1-lp4x-v10.img' failed
```

解决方法：

```
diff --git a/scripts/link-vmlinux.sh b/scripts/link-vmlinux.sh
index 2631ab4ce9fd..04a58483569c 100755
--- a/scripts/link-vmlinux.sh
+++ b/scripts/link-vmlinux.sh
@@ -28,7 +28,7 @@
 # System.map is generated to document addresses of all kernel symbols

 # Error out on error
-set -e
+#set -e
```

8 调试技巧

8.1 打印更多KO加载的log

修改ratelimit的值，可以打印更多init的log，方便查问题，init信息太少会把ko加载的报错信息隐藏掉。

```
xxx@sys2_206:~/a0_Android13_gki/device/rockchip/common$ vim BoardConfig.mk
xxx@sys2_206:~/a0_Android13_gki/device/rockchip/common$ git diff
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 0d1c886..1761ed0 100755
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -392,3 +392,5 @@ ifeq ($(strip $(BOARD_BASEPARAMETER_SUPPORT)), true)
     endif
         BOARD_WITH_SPECIAL_PARTITIONS := baseparameter:1M
 endif
+
+BOARD_KERNEL_CMDLINE += printk.devkmsg=on
```

8.2 在RK的kernel打包中编译GKI使用的boot.img

先按正常编译步骤编译kernel，生成arch/arm64/boot/Image
用如下命令打包boot.img
mkbootimg --kernel arch/arm64/boot/Image --header_version 4 --output ../mkcombinedroot/prebuilts/boot-5.10.img

8.3 查看google发布的内核接口

标准的内核接口定义在android目录下：

```
:~/a5_google_kenrel/common$ tree a
android/ arch/
wlq@sys2_206:~/a5_google_kenrel/common$ tree android/
android/
├── abi_gki_aarch64
├── abi_gki_aarch64_core
├── abi_gki_aarch64_db845c
├── abi_gki_aarch64_exynos
├── abi_gki_aarch64_fips140
├── abi_gki_aarch64_galaxy
├── abi_gki_aarch64_generic
├── abi_gki_aarch64_hikey960
├── abi_gki_aarch64_rockchip
├── abi_gki_aarch64_type_visibility
├── abi_gki_aarch64_virtual_device
├── abi_gki_aarch64.xml
├── abi_gki_modules_exports
├── abi_gki_modules_protected
├── gki_aarch64_fips140_modules
├── gki_aarch64_modules
└── gki_system_dlkm_modules
```

8.4 kernel编译慢卡在LTO优化

kernel在编译的时候会进行LTO优化，这个耗时会很长，如果是做临时调试可以改用LTO thin，可以减少耗时，具体修改如下：

```
...
--- a/arch/arm64/configs/gki_defconfig
+++ b/arch/arm64/configs/gki_defconfig
@@ -97,7 +97,7 @@ CONFIG_CRYPTO_AES_ARM64_CE_BLK=y
 CONFIG_KPROBES=y
 CONFIG_JUMP_LABEL=y
 CONFIG_SHADOW_CALL_STACK=y
-CONFIG_LTO_CLANG_FULL=y
+CONFIG_LTO_CLANG_THIN=y
 CONFIG_CFI_CLANG=y
 CONFIG_MODULES=y
 CONFIG_MODULE_UNLOAD=y
 ...
```

注意，这个修改仅做临时调试用，正式编译时需要还原修改在编译对应KO。

9 如何提交内核接口到upstream

如果需要添加新的内核接口，可以生成对应的patch，再将patch通过瑞芯微的redmine系统提交个瑞芯微审核然后再统一提交给google

```
diff --git a/android/abi_gki_aarch64_rockchip b/android/abi_gki_aarch64_rockchip
index 85bd8bc134cf..3344cf064e06 100644
--- a/android/abi_gki_aarch64_rockchip
+++ b/android/abi_gki_aarch64_rockchip
@@ -2144,6 +2144,15 @@
   mmc_pwrseq_register
   mmc_pwrseq_unregister

+# required by r8168.ko
+  pci_set_mwi
+  pci_clear_mwi
+  proc_get_parent_data
+  skb_checksum_help
+  __skb_gso_segment
+  remove_proc_subtree
+  pci_choose_state
+
 # required by reboot-mode.ko
   devres_release
   kernel_kobj
```