密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

Rockchip Android GKI用户配置指南

文件状态: [ ] 草稿 [√] 正式发布 [ ] 正在修改	文件标识:	RK-KF-YF-722
	当前版本:	V1.0
	作者:	卞金晨/吴良清
	完成日期:	2022-09-01
	审核:	金华君
	审核日期:	2022-09-01

版本号	作者	修改日期	修改说明	备注
V1.0	卞金晨/吴良清	2022-09-01	发布初稿	

文档问题反馈：kenjc.bian@rock-chips.com/wlq@rock-chips.com

目录

1. 名词解释

- GKI (Generic Kernel Image)
  通用内核镜像：是由Google维护的，持续进行更新的内核分支编译出来的通用kernel镜像。
  详细介绍可以访问Google官方网站查看。
- KO (Kernel Object)
  内核模块驱动程序，可以在需要的时候加载，不需要时卸载。`gki-boot.img`中仅包含了通用驱动，因此所有第三方包括原厂的驱动都需要通过`ko`的形式被内核加载。
- ABI (Application Binary Interface)

  指内核中的接口，ko在调用内核接口时，必须确保此接口被暴露出来，否则无法调用。
  Rockchip默认提供了一部分常用的ABI (`android/abi_gki_aarch64.xml & android/abi_gki_aarch64_rockchip`)，如果需要使用的ABI不在此列表内，则需要提交到Google kernel的对应分支。有需要可以联系我们进行提交。

2. 相关目录

- vendor/rockchip/common

```
cf8fc3ce modular_kernel: move to vendor/rockchip/gki from this repo.
```

- device/rockchip/common

```
7bbf3681 gki_common: use mkcombinedroot to build gki.
2757127d build: dtbo: append GKI fix if GKI is enabled.
c30c3c23 fstab_tools: append the strings if -a is provided.
251370ad scripts: parameter_tools: Fix partition list bug.
7e96a837 BoardConfig_AB: Fix expr bug in header v2.
0086616e Make udc configuration available on GKI.
0d84b657 modules: add gki build support.
tag: android-12.1-mid-rkr10
```

- device/rockchip/rk3588

```
b943244 Add dtbo_gki_fix for Android GKI.
```

- mkcombinedroot
- RKTools

```
79fbee8 Linux_Pack_Firmware: package-file: Fix partition list bug.
```

3. 编译方法 (均以RK3588为例，其他平台类比)

3.1 Boot Header v4配置

- 确认uboot增加header v4格式解析支持

```
grep -c "CONFIG_XBC=y" u-boot/.config
```

执行结果为1时才支持，如果为0，请修改对应的uboot编译config。

- Android编译header v4格式镜像
  `device/rockchip/rk3588/BoardConfig.mk`中配置：

```
BOARD_BOOT_HEADER_VERSION :=4
```

- 分区表增加header v4格式的分区

默认情况下，配置header v4后，编译脚本会自动增加`vendor_boot`的分区，默认40M，如果要修改分区大小，请修改：
`device/rockchip/common/build/rockchip/Partitions.mk`

```
BOARD_VENDOR_BOOTIMAGE_PARTITION_SIZE := 41943040
```

如果有自定义了分区表，请自行增加`vendor_boot`分区并配置分区大小。**`boot`分区的大小必须配置为64M，否则无法烧写Google boot.img!!!**

```
BOARD_BOOTIMAGE_PARTITION_SIZE := 67108864
```

- 打包update.img大固件

默认情况下，Rockchip的打包脚本能够根据镜像自动生成`package-file`的打包配置文件，如果是自定义的打包脚本，请自行确认`vendor_boot`有被打包到大固件。

3.2 GKI驱动编译配置，使用gki_defconfig编译模块驱动程序

`device/rockchip/rk3588`：

```
diff --git a/BoardConfig.mk b/BoardConfig.mk
index 71d04b0..6dd2542 100644
--- a/BoardConfig.mk
+++ b/BoardConfig.mk
@@ -21,9 +21,10 @@ TARGET_2ND_CPU_VARIANT_RUNTIME := cortex-a76
 PRODUCT_UBOOT_CONFIG ?= rk3588
 PRODUCT_KERNEL_ARCH ?= arm64
 PRODUCT_KERNEL_DTS ?= rk3588-evb1-lp4-v10
-PRODUCT_KERNEL_CONFIG ?= rockchip_defconfig pcie_wifi.config
+PRODUCT_KERNEL_CONFIG ?= gki_defconfig rockchip_gki.config pcie_wifi.config
+#PRODUCT_KERNEL_CONFIG ?= rockchip_defconfig pcie_wifi.config
```

3.3 升级Google `boot.img`的方法

SDK的kernel源码仅用于ko编译，kernel的核心Image必须使用Google发布的，即Google的boot.img。默认配置位于`mkcombinedroot/modular_kernel.mk`：

```
BOARD_PREBUILT_BOOTIMAGE := \
    $(KERNEL_GKI_DIR)/prebuilts/boot-$(PRODUCT_KERNEL_VERSION).img
```

当前版本即`mkcombinedroot/prebuilts/boot-5.10.img`，在需要更新Google的`boot.img`时，请替换此处镜像。

3.4 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load文件中， load文件在SDK的mkcombinedroot/res/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

触摸屏驱动要在init阶段加载所以加到`vendor_ramdisk_modules.load`中

```
echo "gt1x-ts.ko" >> res/vendor_ramdisk_modules.load
```

触摸屏驱动要在init阶段加载所以加到`vendor_ramdisk_modules.load`中
同时要将也要在`res/debug_list.load`中添加，作为调试用，在这里面加编译的时候才会从kernel中更新对应的ko到vendor_boot.img中。 注意： `res/debug_list.load`仅做调试用，不需要提交到服务器上。

```
echo "gt1x-ts.ko" >> res/debug_list.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-r416183b/bin:$PATH
```

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

- KO编完后进到mkcombinedroot/下执行mkgki4.sh脚本打包vendor_boot.img

```
cd ../mkcombinedroot/
export MY_DTB=rk3588s-evb8-lp4x-v10
./mkgki4.sh
```

5. 验证

- 烧写`mkcombinedroot/out/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

4. 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-r416183b/bin:$PATH
```

- 编译ko

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

3. 进到`mkcombinedroot`目录。将需要更新的ko文件名添加到`res/debug_list.load`中
4. 进到`mkcombinedroot`目录，配置dtb，执行`./mkgki4.sh`将ko文件打包到`vender_boot.img`中

```
cd ../mkcombinedroot/
export MY_DTB=rk3588s-evb8-lp4x-v10
./mkgki4.sh
```

5. 烧写`mkcombinedroot/out/vendor_boot.img`镜像文件到机器中开机验证
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`

5. 开机log确认

5.1 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
```

5.2 Android阶段

GKI版本：`Linux version 5.10.117-android12-9-00037-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
```

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
```

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`

6. VTS相关测试项

使能GKI，编译固件后，请务必先确认各项功能均正常后，再确保以下测试能够测试通过：

```
FirmwareBootHeaderVerification
VtsBootconfigTest
vts_generic_boot_image_test
vts_gsi_boot_test
vts_security_avb_test
```