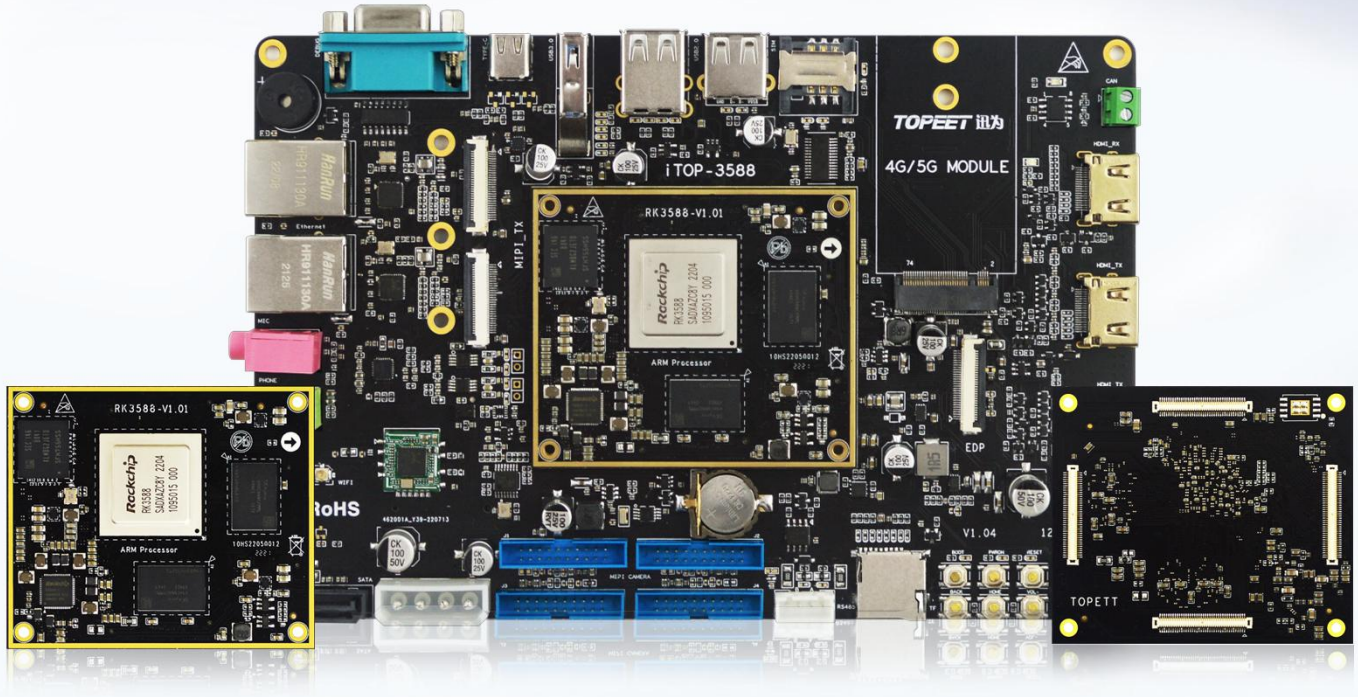
TOPEET北京迅为

强大的 AI 能力 更快更强

超长供货周期 | 7X24 小时稳定运行 | 8K 视频编解码



iTOP-RK3588 开发板使用手册

八核 64 位 CPU | 主频 2.4GHz | NPU 算力 6T | 4800 安防级别 ISP

更新记录

更新版本	修改内容
V1.0	初版

目录

版权声明

更多帮助

注意事项与维护

- ❖ 请注意和遵循标注在产品上的所有警示和指引信息；
- ❖ 请勿带电插拔核心板及外围模块；
- ❖ 使用产品之前，请仔细阅读本手册，并妥善保管，以备将来参考；
- ❖ 请使用配套电源适配器，以保证电压、电流的稳定；
- ❖ 请勿在冷热交替环境中使用本产品，避免结露损坏元器件；
- ❖ 请保持产品干燥，如果不慎被任何液体泼溅或浸润，请立刻断电并充分晾干；
- ❖ 请勿使用有机溶剂或腐蚀性液体清洗本产品；
- ❖ 请勿在多尘、脏乱的环境中使用本产品，如果长期不使用，请包装好本产品；
- ❖ 如果在震动场景使用，请做好核心板与底板的固定，避免核心板跌落损坏；
- ❖ 请勿在通电情况下，插拔核心板及外围模块(特别是串口模块)；
- ❖ 请勿自行维修、拆解本产品，如产品出现故障应及时联系本公司进行维修；
- ❖ 请勿自行修改或使用未经授权的配件，由此造成的损坏将不予保修；

资料的更新

为了确保您的资料是最新状态，请密切关注我们的动态，我们将会通过微信公众号和 QQ 群推送。

关注“迅为电子”微信公众号，不定期分享教程、资料和行业干货及产品一线资料。

迅为新媒体账号

官网：https://www.topeetboard.com

知乎 https://www.zhihu.com/people/topeetabc123

CSDN：https://blog.csdn.net/BeiJingXunWei

售后服务政策

1. 如产品使用过程中出现硬件故障可根据售后服务政策进行维修
2. 服务政策：参见官方网售后服务说明 https://www.topeetboard.com/sydymfl/Service/bx.html

送修地址：

1. 地址：北京市海淀区永翔北路 9 号中国航发大厦三层
2. 联系人：迅为开发板售后服务部
3. 电话：010-85270716
4. 邮编：100094
5. 邮寄须知：建议使用顺丰、圆通或韵达，且不接受任何到付

技术支持范围

1. 了解产品的软、硬件资源提供情况咨询
2. 产品的软、硬件手册使用过程中遇到的问题
3. 下载和烧写更新系统过程中遇到的问题
4. 产品用户的资料丢失、更新后重新获取
5. 产品的故障判断及售后维修服务。

PS：（由于嵌入式系统知识范围广泛，我们无法保证对各种问题都能一一解答，部分内容无法供技术支持，只能提供建议。 ）

技术支持

1. 周一至周五：（法定节假日除外）

上午 9:00 ~ 11:30 / 下午 13:30 ~ 17:30

2. QQ 技术交流群： 824412014
822183461
95631883
861311530

第 1 章 Linux 系统开发笔记

1.1 Linux 系统调试串口改为普通串口

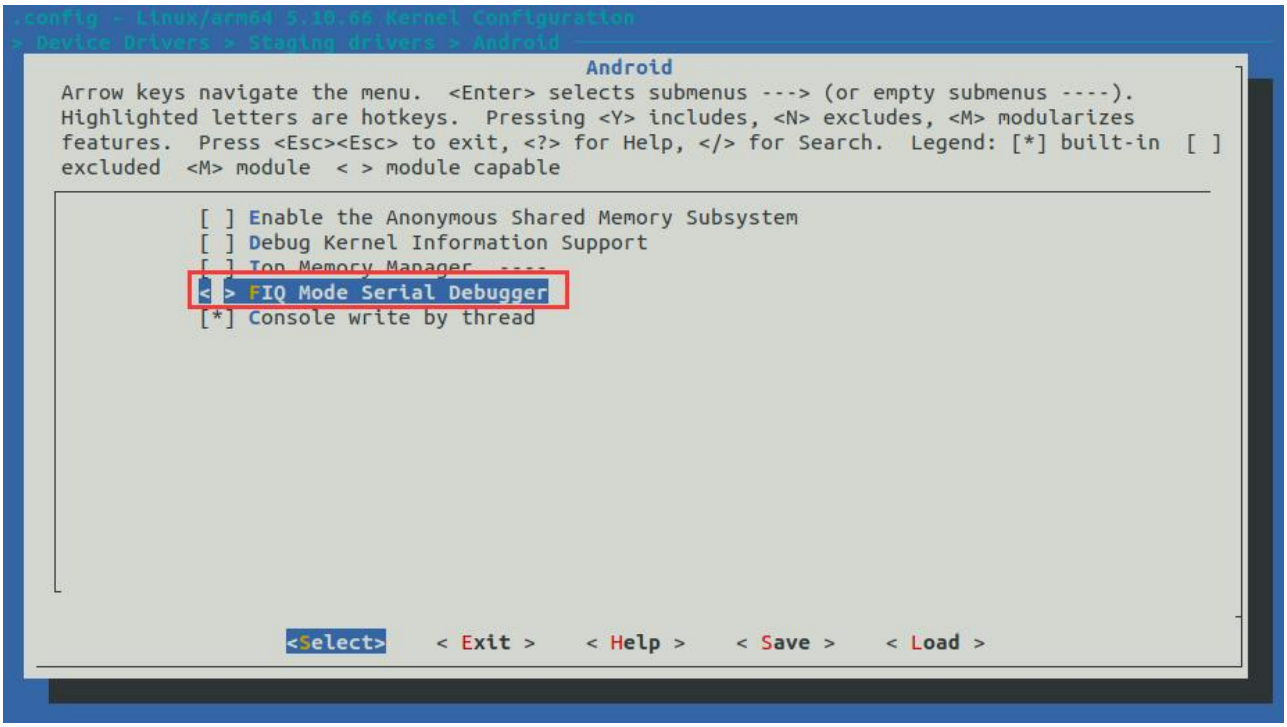
1.1.1 取消 FIQ Debugger 驱动

在源码内核目录输入以下命令打开图形配置界面

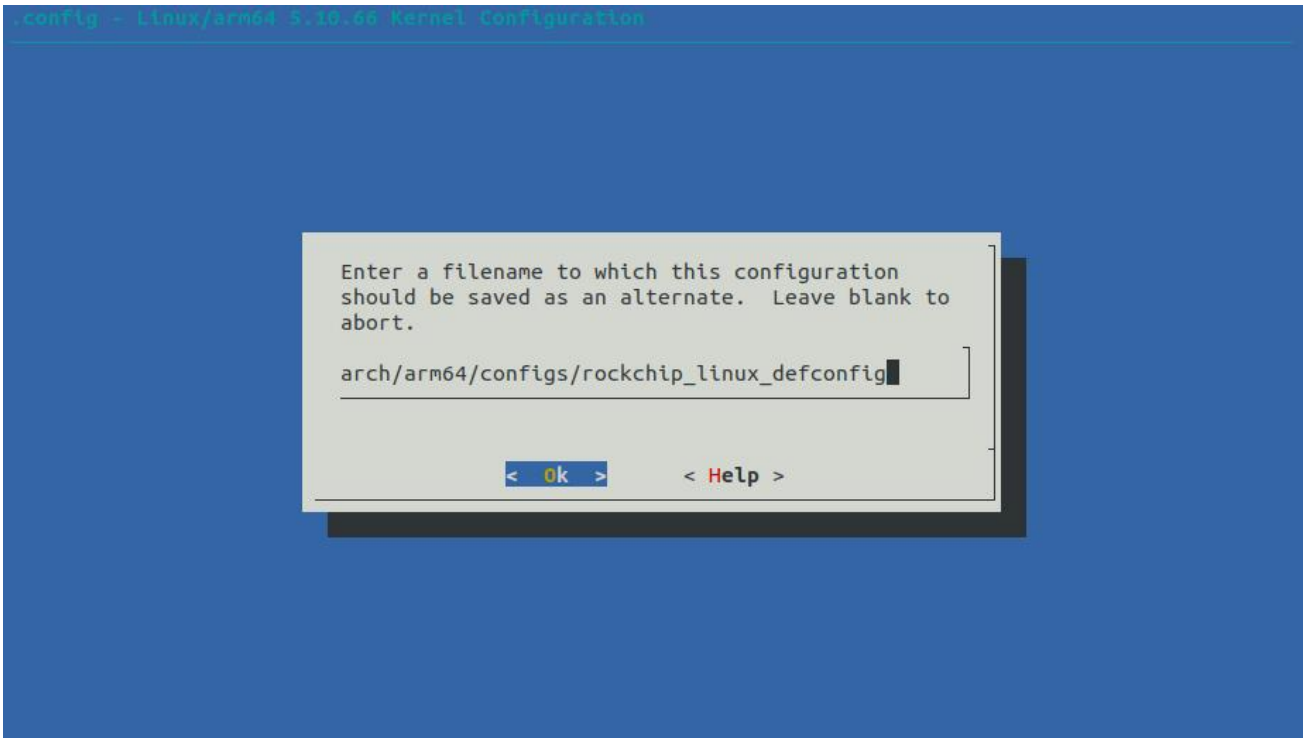
```
export ARCH=arm64
make rockchip_linux_defconfig
make menuconfig
```

按照以下路径，打开 FIQ Debugger 驱动勾选界面，如图所示：

```
> Device Drivers > Staging drivers > Android
```



将 FIQ Debugger 驱动取消勾选，保存退出，保存路径为 arch/arm64/configs/rockchip_linux_defconfig 如图所示：



1.1.2 修改设备树

使用以下命令，将 bootargs 修改为如下图所示：

```
vi arch/arm64/boot/dts/rockchip/rk3588-android.dtsi
```

```
/ {
        chosen: chosen {
                //bootargs = "earlycon=uart8250,mmio32,0xfe660000 console=ttyFIQ0";
                bootargs = "";
        };
```

修改后保存退出，使用以下命令打开设备树文件：

```
vi arch/arm64/boot/dts/rockchip/rk3588-evb7-lp4.dtsi
```

在任意位置加入，将 uart2 打开，如图所示：

```
&usbdp_phy0_dp {
        status = "okay";
};

&uart2{
        status = "okay";
};

&usbdp_phy0_u3 {
        status = "okay";
};
```

保存退出后，使用以下命令打开 uart2 设备树文件，

```
vi kernel/arch/arm64/boot/dts/rockchip/rk3588s.dtsi
```

将 pinctrl 从 uart2m1 改为 uart2m0，如下图所示：

```
uart2: serial@feb50000 {
        compatible = "rockchip,rk3588-uart", "snps,dw-apb-uart";
        reg = <0x0 0xfeb50000 0x0 0x100>;
        interrupts = <GIC_SPI 333 IRQ_TYPE_LEVEL_HIGH>;
        clocks = <&cru SCLK_UART2>, <&cru PCLK_UART2>;
        clock-names = "baudclk", "apb_pclk";
        reg-shift = <2>;
        reg-io-width = <4>;
        dmas = <&dmac0 10>, <&dmac0 11>;
        pinctrl-names = "default";
        pinctrl-0 = <&uart2m0_xfer>;
        status = "disabled";
};
```

保存退出后，回到源码根目录，使用以下命令单独编译内核，

```
./build.sh kernel
```

```
root@ubuntu:/home/topeet/linux/rk3588_linux#
root@ubuntu:/home/topeet/linux/rk3588_linux# ./build.sh kernel
```

编译完成后，将内核单独烧写到开发板，即可完成调试串口改为普通串口。

1.2 Linux 修改 uboot logo

本文档配套资料在网盘资料“iTOP-3588 开发板\02_【iTOP-RK3588 开发板】开发资料\09_Linux 系统开发配套资料\05_Linux 修改 uboot logo 配套资料”路径下。

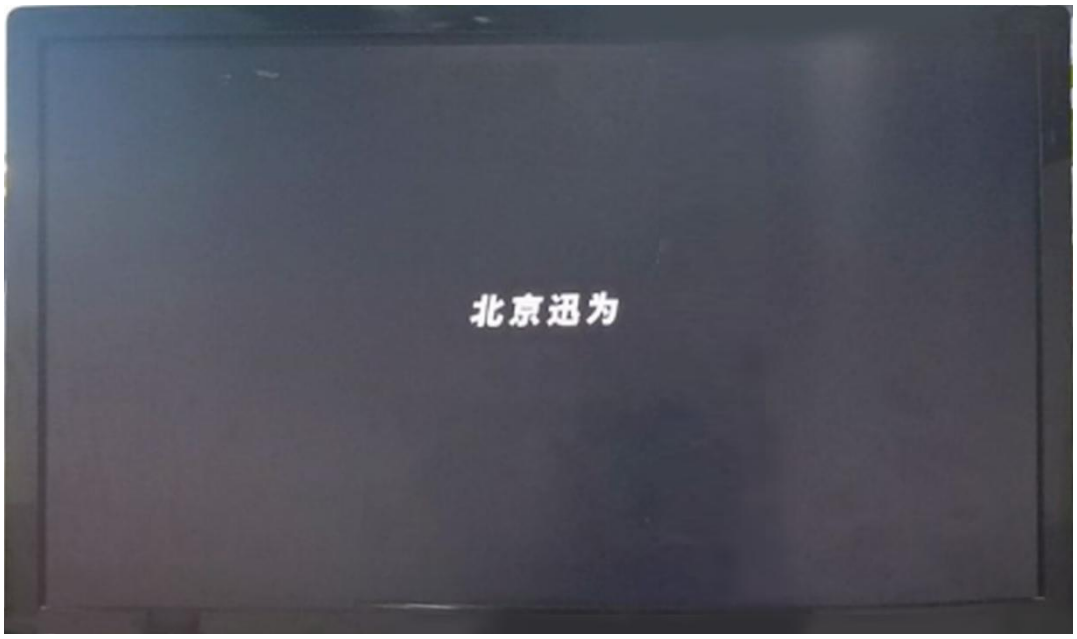
1.2.1 准备 logo

系统默认 uboot logo，如下图所示：

如果想要替换这个 logo,首先要制作一个新的 logo.bmp，图片属性和默认的 logo.bmp 要一样，width，height 都为偶数，否则会出现颠倒异常。

1.2.2 替换 logo

将制作好的 logo 替换 Linux 源码 3588_linux/kernel/logo.bmp 下的 logo.bmp 即可。作者替换后 logo 显示效果如下图所示：



1.3 Linux 修改 kernel logo

本文档配套资料在网盘资料“iTOP-3588 开发板\02_【iTOP-RK3588 开发板】开发资料\09_Linux 系统开发配套资料\06_Linux 修改内核 logo 配套资料”路径下。

1.3.1 准备 logo

系统默认内核 logo，如下图所示：

如果想要替换这个 logo,首先要制作一个新的 logo_kernel.bmp，图片属性和默认的 logo_kernel.bmp 要一样，width，height 都为偶数，否则会出现颠倒异常。

1.3.2 替换 logo

将制作好的 logo 替换 Linux 源码 3588_linux/kernel/logo_kernel.bmp 下的 logo_kernel.bmp 即可。作者修改后，logo 显示效果如下图所示：

1.4 音频开发

1.4.1 声卡查看

使用以下命令查看系统声卡，如下图所示：

```
aplay -l 或者 cat /sys/class/sound/card*/id
```

```
root@RK3588:/#
root@RK3588:/# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: rockchiphdmi1 [rockchip-hdmi1], device 0: rockchip-hdmi1 i2s-hifi-0 [rockchip-hdmi1 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: rockchipes8388 [rockchip-es8388], device 0: dailink-multicodecs ES8323.7-0011-0 [dailink-multicodecs ES8323.7-0011-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
root@RK3588:/# cat /sys/class/sound/card*/id
rockchiphdmi1
rockchipes8388
rockchiphdmiin
root@RK3588:/#
```

如上图，系统有两个声卡，第一个为底板上 es8388 声卡，第二个为 hdmi 音频。

1.4.2 修改默认声卡

HDMI 屏幕镜像默认使用的 hdmi 音频，如果想设置为开发板输出音频可按以下方法进行修改。

首先，使用以下命令查看系统识别的显卡自带的音频模块，如下图所示：

```
aplay -l
```

```
root@RK3588:/#
root@RK3588:/# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: rockchiphdmi1 [rockchip-hdmi1], device 0: rockchip-hdmi1 i2s-hifi-0 [rockchip-hdmi1 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: rockchipes8388 [rockchip-es8388], device 0: dailink-multicodecs ES8323.7-0011-0 [dailink-multicodecs ES8323.7-0011-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
root@RK3588:/#
root@RK3588:/#
```

由上图可看出，0 号卡上的设备 0 是 rockchip 的 HDMI 声卡，1 号卡上的设备 0 是 rockchip 的 es8388 声卡。然后使用以下命令打开配置文件并修改，如下图所示：

```
vi /usr/share/alsa/alsa.conf
```

```
#
# defaults
#

# show all name hints also for definitions without hint {} section
defaults.namehint.showall off
# show just basic name hints
defaults.namehint.basic on
# show extended name hints
defaults.namehint.extended off
#
defaults.ctl.card 0
defaults.pcm.card 0
defaults.pcm.device 0
defaults.pcm.subdevice -1
defaults.pcm.nonblock 1
defaults.pcm.compat 0
defaults.pcm.minperiodtime 5000         # in us
defaults.pcm.ipc_key 5678293
defaults.pcm.ipc_gid audio
defaults.pcm.ipc_perm 0660
defaults.pcm.tstamp_type default
defaults.pcm.dmix.max_periods 0
defaults.pcm.dmix.channels 2
defaults.pcm.dmix.rate 48000
defaults.pcm.dmix.format unchanged
```

此处默认是card 0 HDMI音频可以改为1设置为es8388声卡

修改完配置后重启生效。

1.5 SDK 开发

1.5.1 SDK 板级配置文件

SDK 的板级配置文件在软件开发中扮演着重要角色，用于指定硬件板卡的参数和设置。对于 RK3588 平台，其板级配置文件如下所示：

```
topeet@ubuntu:~/Linux/3588-linux$ ls device/rockchip/rk3588/
BoardConfig-ab-base.mk              BoardConfig-rk3588-evb7-lp4-v10.mk    boot.its
BoardConfig.mk                      BoardConfig-rk3588s-evb1-lp4x-v10.mk  parameter-ab.txt
BoardConfig-rk3588-evb1-lp4-v10.mk  BoardConfig-security-base.mk          parameter.txt
BoardConfig-rk3588-evb3-lp5-v10.mk  boot4recovery.its                     zboot.its
topeet@ubuntu:~/Linux/3588-linux$
```

其中 BoardConfig-rk3588-evb7-lp4-v10.mk 就是我们的 iTOP-RK3588 开发板所使用的板级配置文件；我们在 SDK 根目录下执行“./build.sh lunch”时所列举出来的文件就是从<SDK>/device/rockchip/rk3588/目录来的，如下所示：

```
topeet@ubuntu:~/Linux/3588-linux$ ./build.sh lunch

You're building on Linux
Lunch menu...pick a combo:

0. default BoardConfig.mk
1. BoardConfig-ab-base.mk
2. BoardConfig-rk3588-evb1-lp4-v10.mk
3. BoardConfig-rk3588-evb3-lp5-v10.mk
4. BoardConfig-rk3588-evb7-lp4-v10.mk
5. BoardConfig-rk3588s-evb1-lp4x-v10.mk
6. BoardConfig-security-base.mk
7. BoardConfig.mk
Which would you like? [0]:
```

这些.mk 文件其实是一个 sh 脚本文件，打开 BoardConfig-rk3588-evb7-lp4-v10.mk 文件内容如下所示：

vi device/rockchip/rk3588/BoardConfig-rk3588-evb7-lp4-v10.mk

```
#!/bin/bash

# Target arch
export RK_KERNEL_ARCH=arm64
# Uboot defconfig
export RK_UBOOT_DEFCONFIG=rk3588
# Uboot image format type: fit(flattened image tree)
export RK_UBOOT_FORMAT_TYPE=fit
# Kernel defconfig
export RK_KERNEL_DEFCONFIG=rockchip_linux_defconfig
# Kernel defconfig fragment
export RK_KERNEL_DEFCONFIG_FRAGMENT=rk3588_linux.config
# Kernel dts
export RK_KERNEL_DTS=rk3588-evb7-lp4-v10-linux
# boot image type
export RK_BOOT_IMG=boot.img
# kernel image path
export RK_KERNEL_IMG=kernel/arch/arm64/boot/Image
# kernel image format type: fit(flattened image tree)
export RK_KERNEL_FIT_ITS=boot.its
# parameter for GPT table
export RK_PARAMETER=parameter.txt
# Buildroot config
export RK_CFG_BUILDROOT=rockchip_rk3588
# Recovery config
export RK_CFG_RECOVERY=rockchip_rk3588_recovery
# Recovery image format type: fit(flattened image tree)
export RK_RECOVERY_FIT_ITS=boot4recovery.its
# Pcba config
export RK_CFG_PCBA=rockchip_rk3588_pcba
# target chip
export RK_CHIP=rk3588
# Set rootfs type, including ext2 ext4 squashfs
export RK_ROOTFS_TYPE=ext4
# debian version (debian10: buster, debian11: bullseye)
export RK_DEBIAN_VERSION=bullseye
# yocto machine
export RK_YOCTO_MACHINE=rockchip-rk3588-evb
#misc image
export RK_MISC=wipe_all-misc.img
# Define package-file
export RK_PACKAGE_FILE=rk3588-package-file
# Define WiFi BT chip
# Compatible with Realtek and AP6XXX WiFi : RK_WIFIBT_CHIP=ALL_AP
# Compatible with Realtek and CYWXXX WiFi : RK_WIFIBT_CHIP=ALL_CY
# Single WiFi configuration: AP6256 or CYW43455: RK_WIFIBT_CHIP=AP6256
export RK_WIFIBT_CHIP=ALL_AP
# Define BT ttySX
export RK_WIFIBT_TTY=ttyS8
# <dev>:<mount point>:<fs type>:<mount flags>:<source dir>:<image size(M|K|auto)>:[options]
export RK_EXTRA_PARTITIONS="oem:/oem:ext2:defaults:oem_normal:auto:resize@userdata:/userdata:ext2
:defaults:userdata_normal:auto:resize"
```

这个脚本是用于配置编译环境和参数的，它设置了许多变量，每个变量都对应着编译过程中的不同设置，其中一些重要的设置包括：

RK_ARCH: 目标架构，这里设置为 arm64。
RK_UBOOT_DEFCONFIG: U-Boot 的 defconfig 配置文件；rk3588_defconfig
RK_UBOOT_FORMAT_TYPE: U-Boot 镜像格式类型，这里设置为 fit（扁平化镜像树）。
RK_KERNEL_DEFCONFIG: 内核的配置名称，这里设置为 rockchip_linux_defconfig。
RK_KERNEL_DTS: 内核的设备树，这里设置为 rk3588-evb7-lp4-v10-linux。
RK_BOOT_IMG: 启动镜像的类型，这里设置为 boot.img。
RK_KERNEL_IMG: 内核镜像的路径，这里指定了路径。
RK_KERNEL_FIT_ITS: 内核镜像格式类型，这里设置为 boot.its。
RK_PARAMETER: GPT 表的参数，这里指定了一个文件 parameter.txt。
RK_CFG_BUILDROOT: Buildroot 的 defconfig 配置文件；rockchip_rk3588_defconfig。
RK_YOCTO_MACHINE: Yocto 的机型，这里设置为 rockchip-rk3588-evb。

1.5.2 U-Boot 开发

在 RK3588 平台上，uboot.img 是一个组合镜像，其中包含了多个不同的 image，每个 image 都是一个独立的二进制文件。这些 image 可以是 u-boot 本身、设备树（dtb 文件）、ARM Trusted Firmware（ATF）、OP-TEE 等。这些 image 被打包成一个 FIT 镜像，FIT 镜像是一种包含 image 的数据结构，通过 FIT 可以实现灵活的组合和加载多个 image。

FIT 配置文件.its 是一个文本文件，描述了 FIT 镜像中各个 image 的属性，如类型、加载地址、文件名等。在这个配置文件中，您需要定义各个 image 的属性，以及需要对这些 image 进行签名的方式。此外，还需要定义 FIT 镜像的布局和其他配置信息。

FIT 配置文件 its 通过 mkimage 工具生成 FIT 二进制镜像文件（.itb 文件），这个二进制文件就是 uboot.img 的内容。在生成过程中，工具会将各个 image 按照配置文件中的描述打包到 FIT 镜像中。

U-Boot 编译成功后，U-Boot 源码目录下会生成很多.bin 镜像以及.dtb 镜像：

```
topeet@ubuntu:~/Linux/3588-linux/u-boot$ ls *.bin *.dtb -al
-rw-rw-r-- 1 topeet topeet  194076 11月 30 17:43 bl31_0x00040000.bin
-rw-rw-r-- 1 topeet topeet   24576 11月 30 17:43 bl31_0x000f0000.bin
-rw-rw-r-- 1 topeet topeet   20480 11月 30 17:43 bl31_0xff100000.bin
-rw-rw-r-- 1 topeet topeet  455104 11月 30 17:43 rk3588_spl_loader_v1.09.111.bin
-rw-rw-r-- 1 topeet topeet  461200 11月 30 17:43 tee.bin
-rw-rw-r-- 1 topeet topeet 1306856 11月 30 17:43 u-boot.bin
-rw-rw-r-- 1 topeet topeet    7821 11月 30 17:43 u-boot.dtb
-rw-rw-r-- 1 topeet topeet 1306853 11月 30 17:43 u-boot-dtb.bin
-rwxrwxr-x 1 topeet topeet 1299032 11月 30 17:43 u-boot-nodtb.bin
topeet@ubuntu:~/Linux/3588-linux/u-boot$
```

具体介绍如下表所示：

镜像	说明
u-boot.bin	这是 U-Boot 的主要二进制文件，包含了 U-Boot 的启动代码、命令行解释器、驱动程序和其他功能。它是用于启动加载器的核心文件，负责初始化硬件、加载操作系统内核以及执行各种启动脚本。
u-boot.dtb	这是 U-Boot 的设备树二进制文件
u-boot-dtb.bin	这是在包含设备树信息的情况下生成的 U-Boot 二进制文件。它类似于 u-boot.bin，但还包括设备树数据，用于在引导过程中传递硬件配置信息给操作系统内核。
u-boot-nodtb.bin	这是不包含设备树信息的 U-Boot 二进制文件。它是 u-boot.bin 的变体，用于情况下，不需要传递设备树信息的情况
bl31_0x00040000.bin	这些文件是 ARM Trusted Firmware (ATF) 的组件，用于启动 ARM 架构的处理器。ATF 是一种用于 ARM 架构处理器的开源固件，负责初始化处理器、内存控制器、安全性和设备。不同的文件可能对应于不同的地址或配置。
bl31_0x000f0000.bin	
bl31_0xff100000.bin	
tee.bin	这是用于 ARM TrustZone 的可信执行环境（TEE）的二进制文件。TEE 是一种安全保护环境，用于保护敏感的计算和数据，例如加密和数字签名操作。

uboot.img 镜像最终由 u-boot-nodtb.bin、u-boot.dtb、bl31_0x00040000.bin、bl31_0x000f0000.bin、bl31_0xff100000.bin、tee.bin 这些镜像合并而成。

uboot.img 是 FIT 格式镜像，使用 its 文件来描述 image 的信息，最终通过<UBoot>/tools/mkimage 工具生成 itb 镜像；its 文件和生成的 itb 文件都在<U-Boot>/fit 目录下：

```
topeet@ubuntu:~/Linux/3588-linux/u-boot$ ls -al fit/*
-rw-rw-r-- 1 topeet topeet 2013696 11月 30 17:43 fit/uboot.itb
-rw-rw-r-- 1 topeet topeet    1998 11月 30 17:43 fit/u-boot.its
topeet@ubuntu:~/Linux/3588-linux/u-boot$
```

u-boot.its 文件中描述了有哪些镜像会参与合并成 uboot.itb，以及这些镜像的路径等信息：

```
/*
 * Copyright (C) 2020 Rockchip Electronic Co.,Ltd
 *
 * Simple U-boot fit source file containing ATF/OP-TEE/U-Boot/dtb/MCU
 */

/dts-v1/;
```

```
/ {
        description = "FIT Image with ATF/OP-TEE/U-Boot/MCU";
        #address-cells = <1>;

        images {

                uboot {
                        description = "U-Boot";
                        data = /incbin/("u-boot-nodtb.bin");
                        type = "standalone";
                        arch = "arm64";
                        os = "U-Boot";
                        compression = "none";
                        load = <0x00200000>;
                        hash {
                                algo = "sha256";
                        };
                };
                atf-1 {
                        description = "ARM Trusted Firmware";
                        data = /incbin/("./bl31_0x00040000.bin");
                        type = "firmware";
                        arch = "arm64";
                        os = "arm-trusted-firmware";
                        compression = "none";
                        load = <0x00040000>;
                        hash {
                                algo = "sha256";
                        };
                };
                atf-2 {
                        description = "ARM Trusted Firmware";
                        data = /incbin/("./bl31_0x000f0000.bin");
                        type = "firmware";
```

```
                arch = "arm64";
                os = "arm-trusted-firmware";
                compression = "none";
                load = <0x000f0000>;
                hash {
                        algo = "sha256";
                };
        };
        atf-3 {
                description = "ARM Trusted Firmware";
                data = /incbin/("./bl31_0xff100000.bin");
                type = "firmware";
                arch = "arm64";
                os = "arm-trusted-firmware";
                compression = "none";
                load = <0xff100000>;
                hash {
                        algo = "sha256";
                };
        };
        optee {
                description = "OP-TEE";
                data = /incbin/("tee.bin");
                type = "firmware";
                arch = "arm64";
                os = "op-tee";
                compression = "none";

                load = <0x8400000>;
                hash {
                        algo = "sha256";
                };
        };
        fdt {
```

```
                        description = "U-Boot dtb";
                        data = /incbin/("./u-boot.dtb");
                        type = "flat_dt";
                        arch = "arm64";
                        compression = "none";
                        hash {
                                algo = "sha256";
                        };
                };
        };

        configurations {
                default = "conf";
                conf {
                        description = "rk3588-evb";
                        rollback-index = <0x0>;
                        firmware = "atf-1";
                        loadables = "uboot", "atf-2", "atf-3", "optee";

                        fdt = "fdt";
                        signature {
                                algo = "sha256,rsa2048";

                                key-name-hint = "dev";
                                sign-images = "fdt", "firmware", "loadables";
                        };
                };
        };
};
```

对于 ARM Trusted Firmware 以及 OP-TEE，它们是闭源的，RK 只提供了二进制镜像文件，并没提供源码。ARM Trusted Firmware 固件对应<U-Boot>/bl31.elf， OP-TEE 固件对应<UBoot>/tee.bin，如下所示：

```
topeet@ubuntu:~/Linux/3588-linux/u-boot$ ls -al tee.bin bl31.elf
-rw-rw-r-- 1 topeet topeet 348960 11月 30 17:43 bl31.elf
-rw-rw-r-- 1 topeet topeet 461200 11月 30 17:43 tee.bin
topeet@ubuntu:~/Linux/3588-linux/u-boot$
```

编译 U-Boot 源码之前，这两个镜像是不存在的，编译之后才会出现；实际上来自于<SDK>/rkbin/bin/rk35/rk3588_bl31_v1.33.elf 和<SDK>/rkbin/bin/rk35/rk3588_bl32_v1.12.bin 这两个镜像文件。打包 uboot.itb 的过程中，会将<SDK>/rkbin/bin/rk35/rk3588_bl31_v1.33.elf 拷贝到<U-Boot>/bl31.elf，将<SDK>/rkbin/bin/rk35/rk3588_bl32_v1.12.bin 拷贝到<U-Boot>/tee.bin。

bl31.elf 固件并不是直接打包进 uboot.itb，而是将 bl31.elf 分解成多个 bl31_xxx.bin 文件，最终将这些 bl31_xxx.bin 镜像打包进 uboot.itb。

```
topeet@ubuntu:~/Linux/3588-linux/u-boot$ ls -al bl31_0x*
-rw-rw-r-- 1 topeet topeet 194076 11月 30 17:43 bl31_0x00040000.bin
-rw-rw-r-- 1 topeet topeet  24576 11月 30 17:43 bl31_0x000f0000.bin
-rw-rw-r-- 1 topeet topeet  20480 11月 30 17:43 bl31_0xff100000.bin
topeet@ubuntu:~/Linux/3588-linux/u-boot$
```

其中 rk3588_spl_loader_v1.09.111.bin 文件就是 MiniLoaderAll.bin 镜像，MiniLoaderAll.bin 是运行在 RK3588 平台 U-Boot 之前的一段 Loader 代码（也就是比 U-Boot 更早阶段的 Loader）。MiniLoaderAll.bin 由两部分构成：TPL(Tiny Program Loader) + SPL(Secondary Program Loader)构成。

TPL（Tiny Program Loader）和 SPL（Secondary Program Loader）是引导加载程序的两个阶段，用于启动 Rockchip 芯片上的设备。它们协同工作以初始化硬件并加载引导程序。TPL 负责初始化设备中的基本硬件和加载 SPL，而 SPL 则负责加载更大的引导程序或操作系统。TPL 在启动流程的早期阶段执行，主要用于启动 SPL。SPL 则在 TPL 执行后，负责进一步初始化硬件并加载更大的引导程序，如 U-Boot 或操作系统。

TPL、SPL 分别有两种实现方案：开源版本和闭源版本。这里使用的是闭源版本，在 RK 的文档中，将 RK 提供的闭源 TPL 镜像<SDK>/rkbin/bin/rk35/rk3588_ddr_lp4_2112MHz_lp5_2736MHz_v1.09.bin 称为 ddr bin ，将闭源 SPL 镜像 <SDK>/rkbin/bin/rk35/rk3588_spl_v1.11.bin 称为 miniloader。

rk3588_spl_v1.11.bin 镜像文件是由<U-Boot>/tools/boot_merger 工具制作而成，该工具由 RK 提供。使用 boot_merger 工具制作 rk3588_spl_v1.11.bin 镜像时需要提供一个.ini 文件，.ini 文件用于描述 image 的信息；在<SDK>/rkbin/RKBOOT/目录下有很多.ini 文件，如下图所示：

```
topeet@ubuntu:~/Linux/3588-linux$ ls rkbin/RKBOOT
PX30MINIALL.ini             RK30MINI.ini                RK3326MINIALL.ini           RKNANO.ini
PX30MINIALL_SLC.ini         RK310B.ini                  RK3326MINIALL_SLC.ini       RKNPULIONMINIALL.ini
PX30MINIALL_WO_FTL.ini      RK310BMINIALL.ini           RK3328MINIALL.ini           RKPANDA.ini
PX3SEMINIALL.ini            RK310BMINI.ini              RK3366MINIALL.ini           RKSMART.ini
PX3SEMINIALL_SLC.ini        RK3126.ini                  RK3368BOXMINIALL.ini        RV1106MINIALL_EMMC_TB.ini
PX5KERNEL4.4MINIALL.ini     RK3126MINIALL.ini           RK3368HMINIALL.ini          RV1106MINIALL_EMMC_TB_NOMCU.ini
PX5MINIALL.ini              RK3126MINIALL_SLC.ini       RK3368.ini                  RV1106MINIALL.ini
RK1806MINIALL.ini           RK3126MINIALL_WO_FTL.ini    RK3368MINIALL.ini           RV1106MINIALL_SPI_NOR_TB_GC2093.ini
RK1808MINIALL.ini           RK3128.ini                  RK3399MINIALL.ini           RV1106MINIALL_SPI_NOR_TB_GC3003.ini
RK1808MINIALL_WO_FTL.ini    RK3128MINIALL.ini           RK3399MINIALL_SPINOR.ini    RV1106MINIALL_SPI_NOR_TB.ini
RK302A.ini                  RK3128MINIALL_SLC.ini       RK3399PROMINIALL.ini        RV1106MINIALL_SPI_NOR_TB_NOMCU.ini
RK302AMINIALL.ini           RK3128XMINIALL.ini          RK3528MINIALL.ini           RV1106MINIALL_SPI_NOR_TB_SC230AI.ini
RK302AMINI.ini              RK3188MINIALL.ini           RK3566MINIALL.ini           RV1106MINIALL_SPI_NOR_TB_SC3336_SC3338.ini
RK3032MINIALL.ini           RK322XATMINIALL.ini         RK3566MINIALL_NAND.ini      RV110XMINIALL.ini
RK3032MINIALL_SLC.ini       RK322XHMINIALL.ini          RK3566MINIALL_ULTRA.ini     RV1126MINIALL_EMMC_TB.ini
RK3036_ECHOMINIALL.ini      RK322XMINIALL.ini           RK3568MINIALL.ini           RV1126MINIALL_FTL.ini
RK3036.ini                  RK3288.ini                  RK3568MINIALL_NAND.ini      RV1126MINIALL.ini
RK3036MINIALL.ini           RK3288MINIALL.ini           RK3568MINIALL_RAMBOOT.ini   RV1126MINIALL_IPC.ini
RK3036MINIALL_SLC.ini       RK3308MINIALL.ini           RK3568MINIALL_SPI_NAND.ini  RV1126MINIALL_LP3_EMMC_TB.ini
RK30B.ini                   RK3308MINIALL_SPI_NAND.ini  RK3588MINIALL.ini           RV1126MINIALL_LP4_EMMC_TB.ini
RK30BMINIALL.ini            RK3308MINIALL_UART4.ini     RK3588MINIALL_IPC.ini       RV1126MINIALL_RAMBOOT.ini
RK30BMINI.ini               RK3308MINIALL_WO_FTL.ini    RK3588MINIALL_RAMBOOT.ini   RV1126MINIALL_SPI_NOR_TB.ini
RK30.ini                    RK3326AARCH32MINIALL.ini    RKCAYMAN.ini                RV1126MINIALL_SPI_NOR_TINY.ini
RK30MINIALL.ini             RK3326AARCH32MINIALL_SLC.ini RKCROWN.ini                SDBOOT.ini
topeet@ubuntu:~/Linux/3588-linux$
```

对于 RK3588 平台来说，它默认使用的是 RK3588MINIALL.ini。文件内容如下：

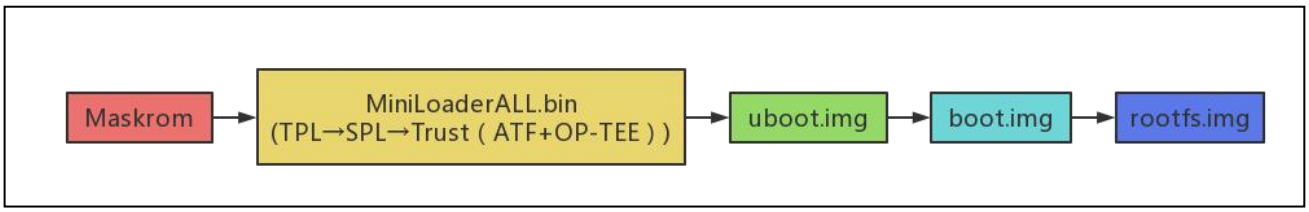
```
[CHIP_NAME]
NAME=RK3588
[VERSION]
MAJOR=1
MINOR=11
[CODE471_OPTION]
NUM=1
Path1=bin/rk35/rk3588_ddr_lp4_2112MHz_lp5_2736MHz_v1.09.bin
Sleep=1
[CODE472_OPTION]
NUM=1
Path1=bin/rk35/rk3588_usbplug_v1.10.bin
[LOADER_OPTION]
NUM=2
LOADER1=FlashData
LOADER2=FlashBoot
FlashData=bin/rk35/rk3588_ddr_lp4_2112MHz_lp5_2736MHz_v1.09.bin
FlashBoot=bin/rk35/rk3588_spl_v1.11.bin
[OUTPUT]
PATH=rk3588_spl_loader_v1.09.111.bin
[SYSTEM]
NEWIDB=true
[FLAG]
471_RC4_OFF=true
RC4_OFF=true
[BOOT1_PARAM]
WORD_0=0x0
WORD_1=0x0
WORD_2=0x0
WORD_3=0x0
WORD_4=0x0
WORD_5=0x0
WORD_6=0x0
WORD_7=0x0
```

FlashData 指定了 tpl 镜像的路径；FlashBoot 指定了 spl 镜像的路径。所以默认情况下，编译 uboot 生成的 rk3588_spl_loader_v1.09.111.bin 使用的是 RK 闭源的 ddr bin 和 RK 闭源的 miniloader。

1.5.3 镜像启动顺序

在 Rockchip 的 64 位 SoC 平台上，为了实现可信环境，使用了 ARM Trusted Firmware（ATF）与 OP-TEE 的组合。

Linux 系统的镜像启动顺序如下图所示：



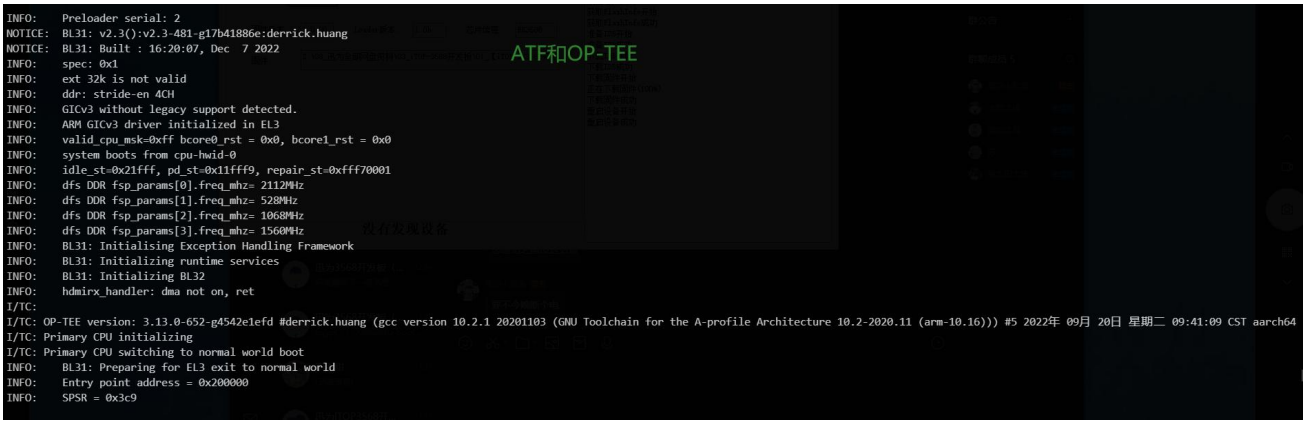
各个部分在开发板启动打印信息如下：

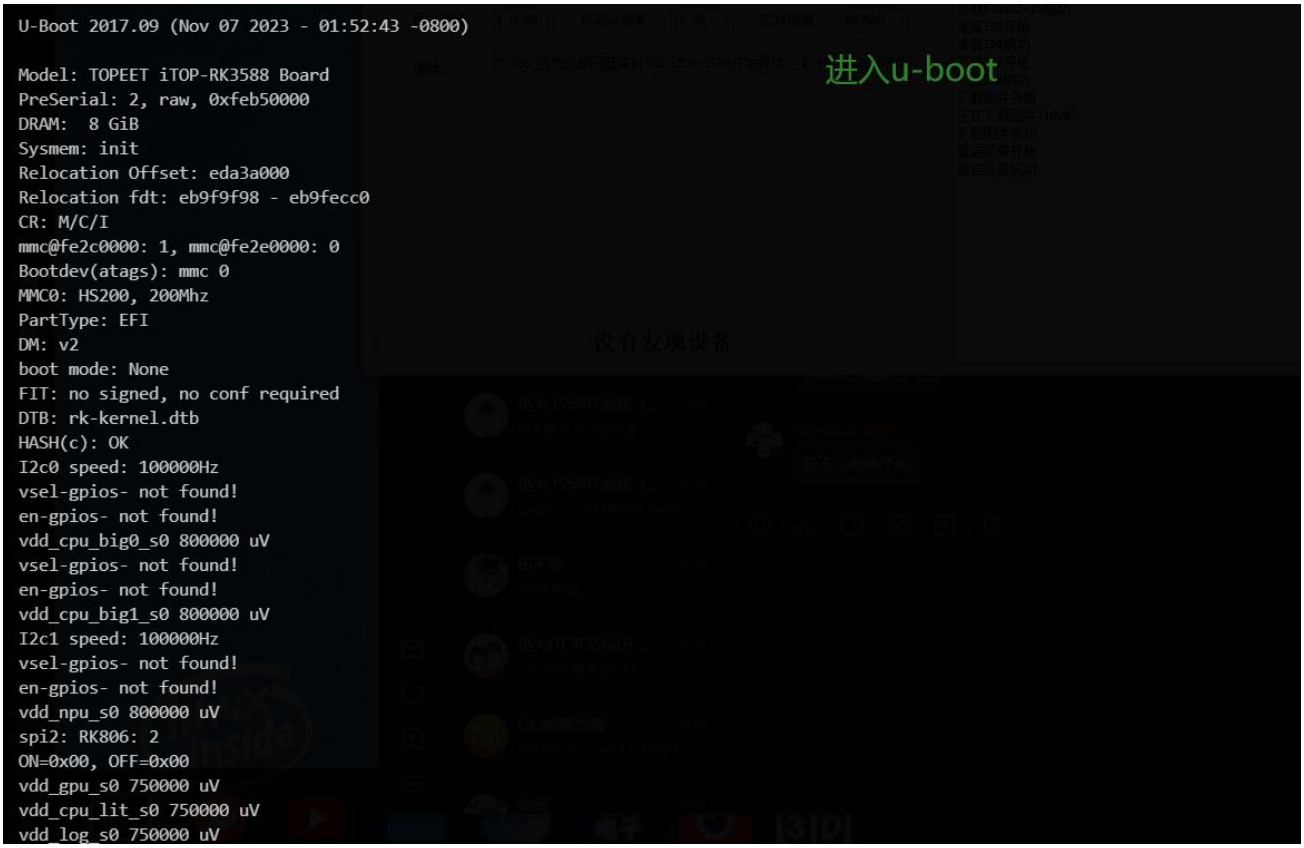
```
root@topeet:/# DDR V1.09 a930779e06 typ 22/11/21-17:50:56
LPDDR4X, 2112MHz
channel[0] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[1] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[2] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
channel[3] BW=16 Col=10 Bk=8 CS0 Row=16 CS1 Row=16 CS=2 Die BW=16 Size=2048MB
Manufacturer ID:0x6
CH0 RX Vref:30.7%, TX Vref:21.8%,21.8%
CH1 RX Vref:30.7%, TX Vref:21.8%,21.8%
CH2 RX Vref:29.7%, TX Vref:21.8%,20.8%
CH3 RX Vref:31.7%, TX Vref:22.8%,21.8%
change to F1: 528MHz
change to F2: 1068MHz
change to F3: 1560MHz
change to F0: 2112MHz
out
```

TPL(ddr bin)

```
U-Boot SPL board init
U-Boot SPL 2017.09-gc060f28d70-220414 #zyf (Apr 18 2022 - 18:13:34)
Failed to set cpub01
Failed to set cpub23
unknown raw ID phN
unrecognized JEDEC id bytes: 00, 00, 00
Trying to boot from MMC2
MMC: no card present
mmc_init: -123, time 2
spl: mmc init failed with error: -123
Trying to boot from MMC1
Trying fit image at 0x4000 sector
## Verified-boot: 0
## Checking atf-1 0x00040000 ... sha256(045b2cef29...) + OK
## Checking uboot 0x00200000 ... sha256(91161ac802...) + OK
## Checking fdt 0x0033d2d8 ... sha256(a4d6ae0a3e...) + OK
## Checking atf-2 0x000f0000 ... sha256(30812190d0...) + OK
## Checking atf-3 0xff100000 ... sha256(cb7bdbec2b...) + OK
## Checking optee 0x08400000 ... sha256(fde0860845...) + OK
Jumping to U-Boot(0x00200000) via ARM Trusted Firmware(0x00040000)
Total: 177.165 ms
```

SPL(miniloader)


ATF和OP-TEE
INFO: Preloader serial: 2
NOTICE: BL31: v2.3():v2.3-481-g17b41886e:derrick.huang
NOTICE: BL31: Built : 16:20:07, Dec 7 2022
INFO: spec: 0x1
INFO: ext 32k is not valid
INFO: ddr: stride-en 4CH
INFO: GICv3 without legacy support detected.
INFO: ARM GICv3 driver initialized in EL3
INFO: valid_cpu_msk=0xff bcore0_rst = 0x0, bcore1_rst = 0x0
INFO: system boots from cpu-hwid-0
INFO: idle_st=0x21fff, pd_st=0x11fff9, repair_st=0xfff70001
INFO: dfs DDR fsp_params[0].freq_mhz= 2112MHz
INFO: dfs DDR fsp_params[1].freq_mhz= 528MHz
INFO: dfs DDR fsp_params[2].freq_mhz= 1068MHz
INFO: dfs DDR fsp_params[3].freq_mhz= 1560MHz
INFO: BL31: Initialising Exception Handling Framework
INFO: BL31: Initializing runtime services
INFO: BL31: Initializing BL32
INFO: hdmirx_handler: dma not on, ret
I/TC:
I/TC: OP-TEE version: 3.13.0-652-g4542e1efd #derrick.huang (gcc version 10.2.1 20201103 (GNU Toolchain for the A-profile Architecture 10.2-2020.11 (arm-10.16))) #5 2022年 09月 20日 星期二 09:41:09 CST aarch64
I/TC: Primary CPU initializing
I/TC: Primary CPU switching to normal world boot
INFO: BL31: Preparing for EL3 exit to normal world
INFO: Entry point address = 0x200000
INFO: SPSR = 0x3c9


进入u-boot
U-Boot 2017.09 (Nov 07 2023 - 01:52:43 -0800)
Model: TOPEET iTOP-RK3588 Board
PreSerial: 2, raw, 0xfeb50000
DRAM: 8 GiB
Sysmem: init
Relocation Offset: eda3a000
Relocation fdt: eb9f9f98 - eb9fecc0
CR: M/C/I
mmc@fe2c0000: 1, mmc@fe2e0000: 0
Bootdev(atags): mmc 0
MMC0: HS200, 200Mhz
PartType: EFI
DM: v2
boot mode: None
FIT: no signed, no conf required
DTB: rk-kernel.dtb
HASH(c): OK
I2c0 speed: 100000Hz
vsel-gpios- not found!
en-gpios- not found!
vdd_cpu_big0_s0 800000 uV
vsel-gpios- not found!
en-gpios- not found!
vdd_cpu_big1_s0 800000 uV
I2c1 speed: 100000Hz
vsel-gpios- not found!
en-gpios- not found!
vdd_npu_s0 800000 uV
spi2: RK806: 2
ON=0x00, OFF=0x00
vdd_gpu_s0 750000 uV
vdd_cpu_lit_s0 750000 uV
vdd_log_s0 750000 uV


进入内核
Starting kernel ...
[1.523214] TOPEET BOARD:iTOP-RK3588
[1.523214] http://www.topeetboard.com
[1.523238] Linux version 5.10.110 (chai@topeet) (aarch64-none-linux-gnu-gcc (GNU Toolchain for the A-profile Architecture 10.3-2021.07 (arm-10.29)) 10.3.1 20210621, GNU ld (GNU Toolchain for the A-profile Architecture 10.3-2021.07 (arm-10.29)) 2.36.1.20210621) #1 SMP Fri Jan 12 09:42:29 CST 2024
[1.535077] Machine model: Rockchip RK3588 EVB7 LP4 V10 Board
[1.535159] earlycon: uart8250 at MMIO32 0x00000000feb50000 (options '')
[1.626194] printk: bootconsole [uart8250] enabled
[1.634039] efi: UEFI not found.
[1.643814] OF: fdt: Reserved memory: failed to reserve memory for node 'drm-cubic-lut@00000000': base 0x0000000000000000, size 0 MiB
[1.656811] Reserved memory: created CMA memory pool at 0x0000000010000000, size 128 MiB
[1.665372] OF: reserved mem: initialized node cma, compatible id shared-dma-pool
[1.819890] Zone ranges:
[1.822605] DMA [mem 0x0000000000200000-0x00000000ffffffff]
[1.829221] DMA32 empty
[1.832315] Normal [mem 0x0000000100000000-0x00000002ffffffff]
[1.838954] Movable zone start for each node

1.5.4 Kernel 开发

RK3588 平台 Linux 系统使用的 boot.img 是一种 FIT 格式镜像，它由多个镜像合并而成，对于 RK3588 平台来说，烧录到开发板 boot 分区的 boot.img 包含了内核镜像 Image、内核 DTB、resource.img 这三部分。具体如下表所示：

文件名	说明
rk3588-evb7-lp4-v10-linux.dtb	内核 DTB。对于 RK 平台来说，这份 DTB 是无效的，它只是一个摆设，RK 平台并未使用这份 DTB，而实际生效的是 resource.img 中的 DTB，因为 RK U-Boot 是直接从 resource.img 中去读取内核 DTB、并将其加载到内存
Image	内核镜像
resource.img	RK 平台资源镜像，resource.img 中包含了内核 DTB、logo 图片。RK U-Boot 会从 resource.img 中读取内核 DTB、而不是直接打包进 boot.img 的这份 DTB，这个一定要搞清楚！

如果改动了设备树文件，执行“./build.sh kernel”命令生成 boot.img 单独烧写 boot.img 镜像即可。

1.5.5 oem 和 userdata

oem.img 包含了厂家的 APP 或数据，该镜像会烧录到开发板的 oem 分区，系统启动之后会被挂载到根文件系统/oem 目录。userdata.img 包含了最终用户的 APP 或数据，该镜像会烧录到开发板的 userdata 分区，系统启动之后会被挂载到根文件系统/userdata 目录。

oem.img 和 userdata.img 镜像是由 mkfirmware.sh 脚本制作而成，具体可看下此脚本。两个镜像的大小是自动的，如需配置，可在板级配置文件中定义 RK_OEM_PARTITION_SIZE 或 RK_USERDATA_PARTITION_SIZE 来指定大小。

在板级配置文件中可以看到这两个镜像均是 ext2 格式文件系统镜像，并且对应目录分别为 device/rockchip/common/images/oem/oem_normal 和 device/rockchip/common/images/userdata/userdata_normal，如下图所示：

```
# <dev>:<mount point>:<fs type>:<mount flags>:<source dir>:<image size(M|K|auto)>:[options]
export RK_EXTRA_PARTITIONS="oem:/oem:ext2:defaults:oem_normal:auto:resize@userdata:/userdata:ext2:defaults:userdata_normal:auto:resize"
```

```
topeet@ubuntu:~/Linux/3588-linux$ ls device/rockchip/common/images/oem/oem_normal/
200frames_count.h264            game_test.gba        retroarch.cfg              SampleVideo_1280x720_5mb.mp4
belle-nuit-testchart-1080p.png  piano2-CoolEdit.mp3  SampleJPGImage_500kbmb.jpg
topeet@ubuntu:~/Linux/3588-linux$ ls device/rockchip/common/images/userdata/userdata_normal/
200frames_count.h264  belle-nuit-testchart-1080p.png  piano2-CoolEdit.mp3
topeet@ubuntu:~/Linux/3588-linux$
```

如果修改了 oem_normal 和 userdata_normal 目录下的文件，则需删除 rockdev 目录下的 oem.img 和 userdata.img 镜像，使用./build.sh firmware 或./mkfirmware.sh 命令重新生成镜像。

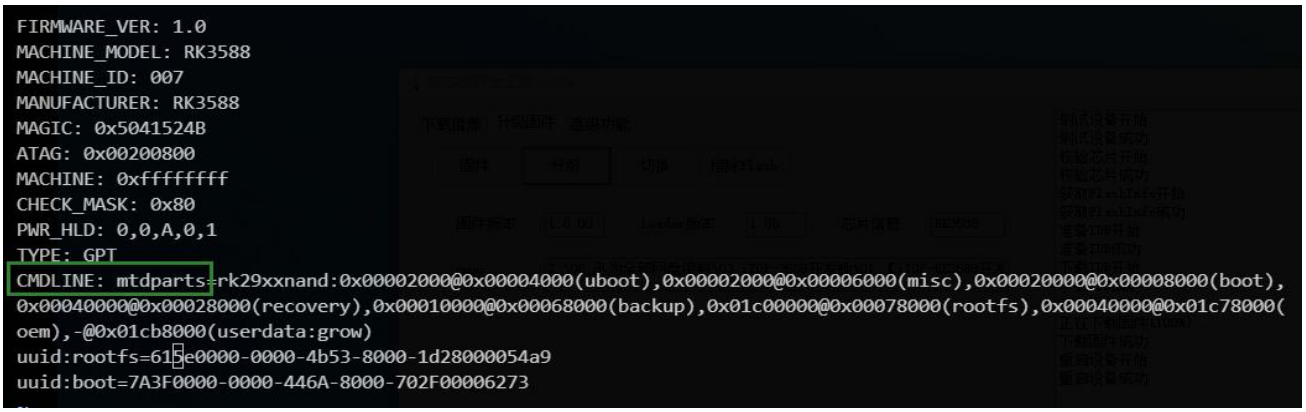
1.5.6 parameter.txt 分区表文件

烧写镜像时，会读取 parameter.tx 的信息来定义存储器的物理分区，各分区的分区名、分区的起始地址以及分区大小。

在板级配置文件中定义了默认使用的分区表文件，如下图所示：

使用以下命令查看其内容，如下所示：

```
vi device/rockchip/rk3588/parameter.txt
```



这里跟分区有关的信息为“CMDLINE: mtdparts=**”。下面我们将对该部分分析

首先我们对该行内容进行分割，分割内容如下

```
CMDLINE: mtdparts=rk29xxnand:
0x00002000@0x00004000(uboot),
0x00002000@0x00006000(misc),
0x00020000@0x00008000(boot),
0x00040000@0x00028000(recovery),
0x00010000@0x00068000(backup),
0x01c00000@0x00078000(rootfs),
0x00040000@0x01c78000(oem),
-@0x01cb8000(userdata:grow)
```

@符号前是分区的大小
@符号后是分区的起始地址
括号中是分区的名字
单位都是 sector（512Bytes）

我们以 uboot 的分区信息为例子进行讲解，0x00002000@0x00004000(uboot)。首先对 0x00002000 sector 化成十进制为 8192 sector，8192*512Bytes=4096KB=4MB

另外 flash 最大的 block 是 4M（0x2000 sectors），所以每个分区需要 4MB 对齐，即每个分区必须为 4MB 的整数倍。可根据自己需求对分区文件进行修改。