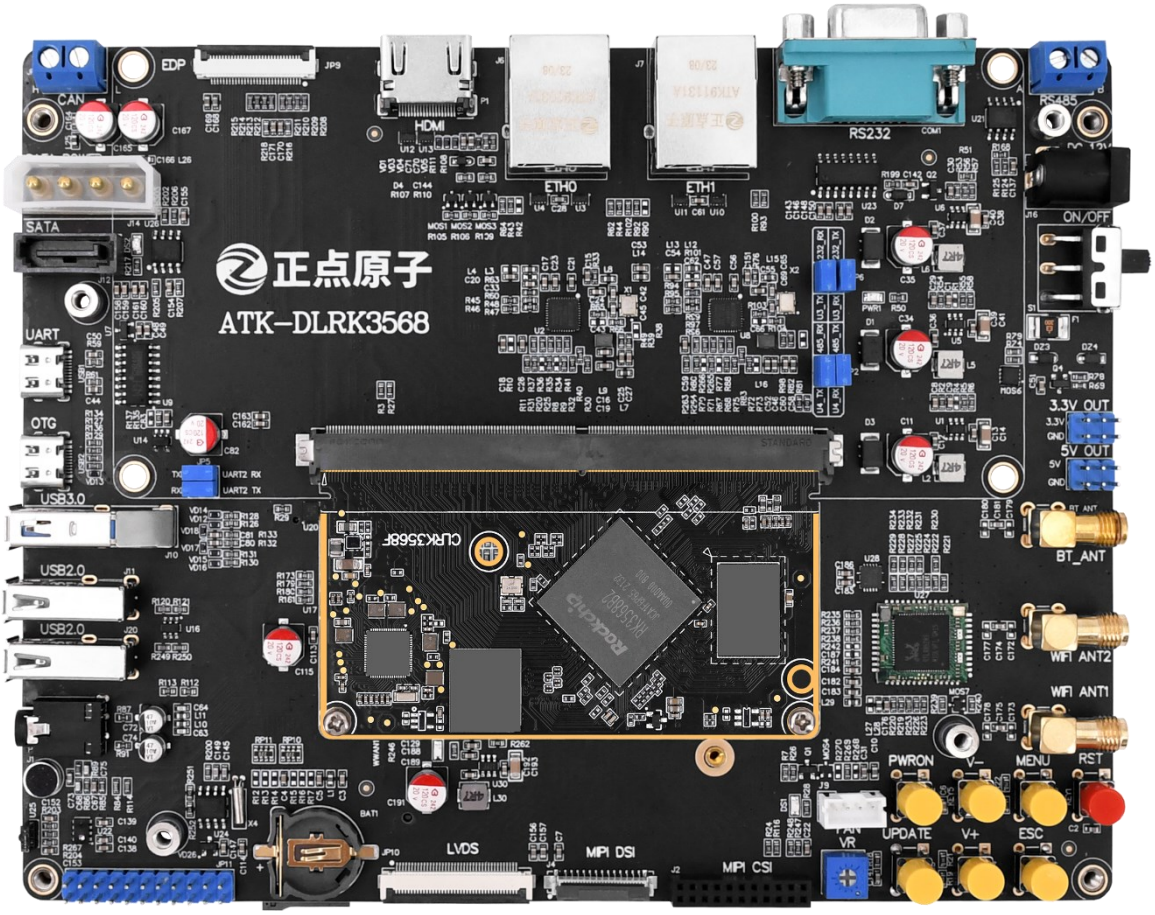
ATK-DLRK3568

Android12 SDK Compilation Instructions

V1.2



ALIENTEK

1. Shopping:

TMALL: https://zhengdianyuanzi.tmall.com

TAOBAO: https://openedv.taobao.com

2. Download

Address: http://www.openedv.com/docs/index.html

3. FAE

Website : www.alientek.com

Forum : http://www.openedv.com/forum.php

Videos : www.yuanzige.com

Fax : +86 - 20 - 36773971

Phone : +86 - 20 - 38271790

Disclaimer

The product specifications and instructions mentioned in this document are for reference only and subject to update without prior notice; Unless otherwise agreed, this document is intended as a product guide only, and none of the representations made herein constitutes a warranty of any kind. The copyright of this document belongs to Guangzhou Xingyi Electronic Technology Co., LTD. Without the written permission of the company, any unit or individual shall not be used for profit-making purposes in any way of dissemination.

In order to get the latest version of product information, please regularly visit the download center or contact the customer service of Taobao ALIENTEK flagship store. Thank you for your tolerance and support.

Revision History:

Version	Version Update Notes	Responsible person	Proofreading	Date
V1.0	release officially	ALIENTEK	ALIENTEK	2023.08.01
V1.1	Add a note: Instructions for the Ubuntu terminal	ALIENTEK	ALIENTEK	2023.10.26
V1.2	Change to the repo checkout method	ALIENTEK	ALIENTEK	2024.07.10

Catalogue

Brief 1
Chapter 1. Android 12 SDK compilation 2
1.1 Copy the SDK to the Ubuntu system 2
1.2 Compiling SDK 3
1.3 Compilation Error 9
1.4 Package as update.img 11

Brief

This document is the compilation instruction manual for the ATK-DLRK3568 Android12 SDK. Before reading this document, please first refer to the contents of Chapter 1 and Chapter 2 in the document: <Development Board CD-ROM A Disk - **Basic Materials→10_user_manual→02, Development Documents→ [ALIENTEK] ATK-DLRK3568_Android System Development Manual.pdf>** which cover the installation of the Ubuntu system and the setup of the development environment. And follow the instructions in Section 4.1.1 of this document to install the required software packages.

Chapter 1. Android 12 SDK compilation

ATK-DLRK3568 Android 12 SDK Compilation Instructions.

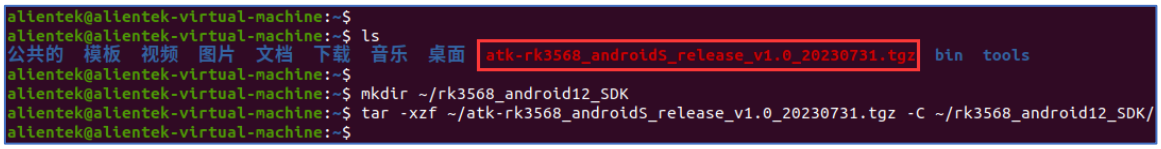
1.1 Copy the SDK to the Ubuntu system

The path of the Android 12 SDK is: Development board CD-ROM B drive - Development environment and SDK→02, ATK-DLRK3568 development board SDK atk-rk3568_androidS_release_v1.0_20230731.tgz. As the version updates, the name of the SDK compressed file will also change, but they are all named in the format of atk-rk3568_androidS_release_ version _ release date.tgz.

Note: If the current terminal has already compiled the RK3568 Android 11 SDK, you need to open a new terminal first, and then compile the Android 12 SDK in this new terminal. Otherwise, it may cause compilation errors!!!

Copy the **atk-rk3568_androidS_release_v1.0_20230731.tgz** compressed file to the home directory of the user on the Ubuntu system, and execute the following command to decompress it:

```
mkdir ~/rk3568_android12_SDK
tar -xzf ~/atk-rk3568_androidS_release_v1.0_20230731.tgz -C ~/rk3568_android12_SDK/
```



The decompression process will take a long time. Please be patient and wait!

After decompression is completed, execute the following command to check out the source code:

```
cd ~/rk3568_android12_SDK/
.repo/repo/repo sync -l -j10
```

This process will also last for a long time. Please be patient and wait!

After the checkout is completed, the entire source code directory of Android 12 SDK is obtained, as shown below:

```
tgg@tgg-virtual-machine:~$
tgg@tgg-virtual-machine:~$ cd rk3568_android12_SDK/
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$ ls
Android.bp      build.sh        external      libcore           packages            rkst        tools
art             compatibility   frameworks    libnativehelper   pdk                 RKTools     u-boot
bionic          cts             hardware      libOpenCL.so      platform_testing    rockdev     vendor
bootable        dalvik          javaenv.sh    Makefile          prebuilts           sdk         WORKSPACE
bootstrap.bash  developers      kernel        mkcombinedroot    restore_patches.sh  system
build           development     kernel-4.19   mkimage_ab.sh     rkbin               test
BUILD           device          kernel-5.10   mkimage.sh        RKDocs              toolchain
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
```

Android 12 SDK source code directory

1.2 Compiling SDK

Enter the rk3568_android12_SDK directory. First, execute the following command to initialize the compilation environment:

```
source build/envsetup.sh
```

```
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$ source build/envsetup.sh
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
```

Initialize the compilation environment

Note: Before opening a new terminal to compile the Android source code each time, you need to execute the "source build/envsetup.sh" command to initialize the compilation environment.

Then execute the "lunch" command to select a product (that is, to choose a compilation target and compilation options):

```
lunch
```

```
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$ lunch

You're building on Linux

Lunch menu... pick a combo:
    1. PX30_Android12-user
    2. PX30_Android12-userdebug
    3. aosp_arm-eng
    4. aosp_arm64-eng
    5. aosp_blueline-userdebug
    6. aosp_blueline_car-userdebug
    7. aosp_bonito-userdebug
    8. aosp_bonito_car-userdebug
    9. aosp_bramble_car-userdebug
    10. aosp_cf_arm64_auto-userdebug
    11. aosp_cf_arm64_phone-userdebug
    12. aosp_cf_x86_64_foldable-userdebug
    13. aosp_cf_x86_64_pc-userdebug
```

14. aosp_cf_x86_64_phone-userdebug
15. aosp_cf_x86_64_tv-userdebug
16. aosp_cf_x86_auto-userdebug
17. aosp_cf_x86_phone-userdebug
18. aosp_cf_x86_tv-userdebug
19. aosp_coral_car-userdebug
20. aosp_crosshatch-userdebug
21. aosp_crosshatch_car-userdebug
22. aosp_crosshatch_vf-userdebug
23. aosp_flame_car-userdebug
24. aosp_oriole-userdebug
25. aosp_oriole_car-userdebug
26. aosp_raven-userdebug
27. aosp_raven_car-userdebug
28. aosp_redfin_car-userdebug
29. aosp_sargo-userdebug
30. aosp_sargo_car-userdebug
31. aosp_slider-userdebug
32. aosp_sunfish_car-userdebug
33. aosp_whitefin-userdebug
34. aosp_x86-eng
35. aosp_x86_64-eng
36. arm_krait-eng
37. arm_v7_v8-eng
38. armv8-eng
39. armv8_cortex_a55-eng
40. armv8_kryo385-eng
41. beagle_x15-userdebug
42. beagle_x15_auto-userdebug
43. fuchsia_arm64-eng
44. fuchsia_x86_64-eng
45. hikey-userdebug
46. hikey64_only-userdebug
47. hikey960-userdebug
48. hikey960_tv-userdebug
49. hikey_tv-userdebug
50. qemu_trusty_arm64-userdebug
51. rk3288_Android10-user
52. rk3288_Android10-userdebug
53. rk3288_Android11-user
54. rk3288_Android11-userdebug
55. rk3288_Android12-user
56. rk3288_Android12-userdebug

```
    57. rk3326_pie-user
    58. rk3326_pie-userdebug
    59. rk3326_q-user
    60. rk3326_q-userdebug
    61. rk3326_r-user
    62. rk3326_r-userdebug
    63. rk3326_s-user
    64. rk3326_s-userdebug
    65. rk3326_sgo-user
    66. rk3326_sgo-userdebug
    67. rk3399_Android10-user
    68. rk3399_Android10-userdebug
    69. rk3399_Android11-user
    70. rk3399_Android11-userdebug
    71. rk3399_Android12-user
    72. rk3399_Android12-userdebug
    73. rk3399_mid-user
    74. rk3399_mid-userdebug
    75. rk3566_32bit-user
    76. rk3566_32bit-userdebug
    77. rk3566_eink-user
    78. rk3566_eink-userdebug
    79. rk3566_einkw6-user
    80. rk3566_einkw6-userdebug
    81. rk3566_r-user
    82. rk3566_r-userdebug
    83. rk3566_s-user
    84. rk3566_s-userdebug
    85. rk3566_sgo-user
    86. rk3566_sgo-userdebug
    87. rk3568_s-user
    88. rk3568_s-userdebug
    89. sdk_car_arm-userdebug
    90. sdk_car_arm64-userdebug
    91. sdk_car_portrait_x86_64-userdebug
    92. sdk_car_x86-userdebug
    93. sdk_car_x86_64-userdebug
    94. silvermont-eng
    95. uml-userdebug
    96. yukawa-userdebug
    97. yukawa_sei510-userdebug

Which would you like? [aosp_arm-eng]
```

After executing the "lunch" command (without any parameters), it will list all the products (compilation targets). For the rk3568 platform, we can choose either **rk3568_s-user** or **rk3568_s-userdebug**; rk3568_s-user will compile the user version of the Android 12 system image, while rk3568_s-userdebug will compile the userdebug version of the Android 12 system image.

Here, taking the userdebug version as an example, after the string "**Which would you like? [aosp_arm-eng]**" enter "**rk3568_s-userdebug**" or the corresponding number 88, and then press Enter:

```
     83. rk3566_s-user
     84. rk3566_s-userdebug
     85. rk3566_sgo-user
     86. rk3566_sgo-userdebug
     87. rk3568_s-user
     88. rk3568_s-userdebug
     89. sdk_car_arm-userdebug
     90. sdk_car_arm64-userdebug
     91. sdk_car_portrait_x86_64-userdebug
     92. sdk_car_x86-userdebug
     93. sdk_car_x86_64-userdebug
     94. silvermont-eng
     95. uml-userdebug
     96. yukawa-userdebug
     97. yukawa_sei510-userdebug

Which would you like? [aosp_arm-eng] rk3568_s-userdebug

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=12
TARGET_PRODUCT=rk3568_s
TARGET_BUILD_VARIANT=userdebug
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=cortex-a55
TARGET_2ND_ARCH=arm
TARGET_2ND_ARCH_VARIANT=armv8-2a
TARGET_2ND_CPU_VARIANT=cortex-a55
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-5.15.0-53-generic-x86_64-Ubuntu-20.04.2-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=SQ3A.220605.009.B1
OUT_DIR=out
============================================
```

Enter the configuration name or the corresponding number

Execute the "lunch" command

It is also possible to directly specify the product name when executing the "lunch" command, as follows:

```
lunch rk3568_s-userdebug
```

Now we can proceed with the compilation. Use the compiled script build.sh provided by RK for compilation.

build.sh is a compiled script packaged by RK. By using this script, users can conveniently quickly build the entire Android system image file and perform packaging operations on the image. It can automatically compile the entire Android SDK with one click, and also separately compile U-Boot, Linux Kernel, Android 12 source code, etc. It is very convenient!

You can view the usage method of the build.sh script by using the following command:

```
./build.sh -h
```

```
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$ ./build.sh -h
./build.sh: 非法选项 -- h
USAGE: [-U] [-CK] [-A] [-p] [-o] [-u] [-v VERSION_NAME]
No ARGS means use default build option
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
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
```

As shown in the following table:

Parameter	Explanation	Example
-U	Compile the U-Boot source code	./build.sh -U
-C	Compile the Linux Kernel source code using the clang compiler; if GMS certification is required, Google mandates the use of the clang compiler for compiling the kernel source code	./build.sh -C
-K	Compile the Linux Kernel source code	./build.sh -K
-A	Compile the Android source code	./build.sh -A
-p	Package the compiled images into the IMAGE directory	./build.sh -p
-o	Compile the OTA upgrade package	./build.sh -o
-u	Package all the images into update.img	./build.sh -u
-J	Specify the number of compilation threads	./build.sh -J16

Introduction to Common Parameters of the build.sh Script

In the root directory of the SDK source code, execute the following command to compile the entire Android SDK:

```
./build.sh -UKA -J16
```

```
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$ ./build.sh -UKA -J16
will build u-boot
will build kernel
will build android
-------------------KERNEL_VERSION:4.19
-------------------KERNEL_DTS:rk3568-atk-evb1-ddr4-v10

============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=12
TARGET_PRODUCT=rk3568_s
TARGET_BUILD_VARIANT=userdebug
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=cortex-a55
TARGET_2ND_ARCH=arm
TARGET_2ND_ARCH_VARIANT=armv8-2a
TARGET_2ND_CPU_VARIANT=cortex-a55
HOST_ARCH=x86_64
HOST_2ND_ARCH=x86
HOST_OS=linux
HOST_OS_EXTRA=Linux-5.15.0-53-generic-x86_64-Ubuntu-20.04.2-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=SQ3A.220605.009.B1
OUT_DIR=out
============================================
start build uboot
  CLEAN   dts/../arch/arm/dts
  CLEAN   dts
  CLEAN   examples/standalone
  CLEAN   tools
  CLEAN   tools/lib tools/common
  CLEAN   spl/arch spl/board spl/cmd spl/common spl/disk spl/drivers spl/dts spl/env spl/fs spl/lib spl/u-boot.c
pl.lds spl/u-boot-spl.map spl/u-boot-spl-nodtb.bin spl/u-boot-spl.sym tpl/arch tpl/board tpl/common tpl/disk tpl
 tpl/u-boot-tpl.map tpl/u-boot-tpl-nodtb.bin tpl/u-boot-tpl.sym
  CLEAN   u-boot-dtb.bin u-boot.lds u-boot.dtb u-boot.map u-boot-nodtb.bin u-boot.srec u-boot.bin u-boot u-boot.
 bl31_0xfdcd0000.bin bl31_0x00040000.bin bl31_0xfdcce000.bin bl31_0x0006a000.bin u-boot-nodtb.bin bl31_0xfdcc100

#### build completed successfully (1 seconds) ####
```

One-click automatic compilation of Android SDK

-U: Indicates compiling U-Boot;

-K: Indicates compiling Linux Kernel;

-A: Indicates compiling Android;

-J16: Specifies the number of compilation threads to be 16.

The entire compilation process will take a considerable amount of time, approximately 3 to 4 hours or longer, depending on the configuration of the personal computer (CPU and memory).

If there are no unexpected issues, the compilation will be successful, as shown in the following image:

```
[100% 127408/127408] Target super fs image for debug: out/target/product/rk3568_s/super.img
2023-08-05 13:29:27 - build_super_image.py - INFO    : Building super image from info dict...
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 239610 4096-byte output blocks in 18 input chunks.
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 37419 4096-byte output blocks in 6 input chunks.
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 83310 4096-byte output blocks in 9 input chunks.
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 2271 4096-byte output blocks in 2 input chunks.
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 169 4096-byte output blocks in 2 input chunks.
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 64 4096-byte output blocks in 4 input chunks.
2023-08-05 13:29:27 - sparse_img.py - INFO    : Total of 61252 4096-byte output blocks in 5 input chunks.
2023-08-05 13:30:25 - build_super_image.py - INFO    : Done writing image out/target/product/rk3568_s/super.img

#### build completed successfully (03:11:08 (hh:mm:ss)) ####

Build android ok!
make and copy android images
TARGET_PRODUCT=rk3568_s
TARGET_BASE_PARAMETER_IMAGE==device/rockchip/common/baseparameter/v2.0/baseparameter.img
HIGH_RELIABLE_RECOVERY_OTA=
BOARD_AVB_ENABLE=false
system filesysystem is ext4
create dtbo.img...
done.
create rockdev/Image-rk3568_s/resource.img...
done.
create rockdev/Image-rk3568_s/boot.img...
done.
create rockdev/Image-rk3568_s/boot-debug.img...
done.
skip copy images: /home/tgg/rk3568_android12_SDK/out/target/product/rk3568_s/vendor_boot.img
skip copy images: /home/tgg/rk3568_android12_SDK/out/target/product/rk3568_s/vendor_boot-debug.img
create rockdev/Image-rk3568_s/recovery.img...
done.
create rockdev/Image-rk3568_s/super.img...
done.
skip copy images: /home/tgg/rk3568_android12_SDK/out/target/product/rk3568_s/userdata.img
create vbmeta.img...
BOARD_AVB_ENABLE is false, use default vbmeta.img
create misc.img.... done.
create uboot.img...
u-boot/trust.img not fount! Please make it from u-boot first!
create loader...
create config.cfg...
create baseparameter...done.
Make image ok!
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
```

The compilation took 3 hours and 11 minutes.

Compilation successful.

After the compilation is completed, the generated image file will be packaged into the **<SDK>/rockdev/Image-rk3568_s/** directory, as shown below:

```
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$
tgg@tgg-virtual-machine:~/rk3568_android12_SDK$ cd rockdev/Image-rk3568_s/
tgg@tgg-virtual-machine:~/rk3568_android12_SDK/rockdev/Image-rk3568_s$ ls
baseparameter.img  boot.img    dtbo.img          misc.img       pcba_small_misc.img  recovery.img  super.img  vbmeta.img
boot-debug.img     config.cfg  MiniLoaderAll.bin  parameter.txt  pcba_whole_misc.img  resource.img  uboot.img
tgg@tgg-virtual-machine:~/rk3568_android12_SDK/rockdev/Image-rk3568_s$
```

The compiled generated image file

Is exactly the same as the image file generated after compiling the Android 11 SDK.

1.3 Compilation Error

If the following error message appears during the compilation process:

```
>>>>>>>>>>>>>>>>>>>>> rk356x
libeptz enabled on RK356X
libeptz enabled on RK356X
libeptz enabled on RK356X
librknnrt enabled on RK356X
librknnrt enabled on RK356X
librknnrt enabled on RK356X
librockx enabled on RK356X
librockx enabled on RK356X
librockx enabled on RK356X
error: hardware/rockchip/libmpimmz/Android.bp:24:1: module "gen_mmz_version": module source path "hardware/rockchip/libmpimmz/.git/HEAD" does not exist
error: vendor/rockchip/hardware/interfaces/codec2/Android.bp:1:1: module "c2_version": module source path "vendor/rockchip/hardware/interfaces/codec2/.gi
error: hardware/rockchip/libhwjpeg/Android.bp:1:1: module "gen_hwjpeg_version": module source path "hardware/rockchip/libhwjpeg/.git/HEAD" does not exist
10:10:52 soong bootstrap failed with: exit status 1

#### failed to build some targets (51 seconds) ####
```

Compile error message

Please handle it according to the following method:

① , Open the file "**hardware/rockchip/libmpimmz/Android.bp**", and delete "**.git/HEAD**":

```
24 genrule {
25     name: "gen_mmz_version",
26     srcs: libmpimmz_src + [
27         "version.sh",
28         "version.h.template",
29         ".git/HEAD"
30     ],
31     out: ["version.h"],
32     cmd: "bash $(location version.sh) < $(in) > $(out)",
33 }
34
```

Delete this line

Delete, then save and exit.

② Open the "**vendor/rockchip/hardware/interfaces/codec2/Android.bp**" file and delete "**.git/HEAD**":

```
 1 genrule {
 2     name: "c2_version",
 3     srcs: [
 4         "version.h.template",
 5         "version.sh",
 6         ".git/HEAD",
 7     ],
 8     cmd: "bash $(location version.sh) > $(out)",
 9     out: ["C2RKVersion.h"],
10 }
~
```

Delete this line

Delete, then save and exit.

③ . Open the "**hardware/rockchip/libhwjpeg/Android.bp**" file and delete "**.git/HEAD**":

```
 1 genrule {
 2     name: "gen_hwjpeg_version",
 3     srcs: [
 4         "version.h.template",
 5         "genversion.sh",
 6         ".git/HEAD",
 7     ],
 8     cmd: "bash $(location genversion.sh) > $(out)",
 9     out: ["version.h"],
10 }
11
```

Delete this line

Delete, then save and exit.

After modification, simply recompile!

1.4 Package as update.img

Execute the following command in the root directory of the SDK to package the image in the rockdev/Image-rk3568_s/ directory into an update.img firmware, which is convenient for users to flash:

```
./build.sh -u
```