

Rockchip

Recovery Developer Guide

Release version: 1.0.4

Date: 2018.12

Warranty Disclaimer

This document is provided by "status" and Fuzhou Rockchip Electronics Co., Ltd ("our company ", the same below) does not provide any express or implied statement or warranty of any accuracy, reliability, integrity, merchantability, specific purpose and non-infringement of any statement, information and content of this document. This document is intended as a guide only for use.

Due to product version upgrades or other reasons, this document may be updated or modified from time to time without notice.

Trademarks

Rockchip and “瑞芯微”、“瑞芯” are trademarks of Fuzhou Rockchip Electronics Co., Ltd. and are exclusively owned by Fuzhou Rockchip Electronics Co., Ltd.

References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

Copyright © 2018 Fuzhou Rockchip Electronics Co., Ltd.

Exceeding the scope of reasonable use, No part of this publication may be copied or reproduced without the written permission of our company, and may not be transmitted in any form.

Fuzhou Rockchip Electronics Co., Ltd

Address: No.18 Building, A District, No.89 Software Boulevard, FuZhou, FuJian, PRC

Website: www.rock-chips.com

Tel: +86-591-83991906

Fax: +86-591-83951833

Mail: service@rock-chips.com

Preface

Overview

This document mainly introduces recovery development process and technical details for OTA upgrading of Rockchip processors.

The document introduces the detailed development process and notices of the solution.

Product version

Chipset name	Kernel version
RK3308, RK3226, RK3399, RK3288, etc.	4.4

Applicable object

This document (the guide) is mainly suitable for below engineers:

- Field application engineers
- Software development engineers

Revision history

Date	Version	Author	Revision Description
2018.09.18	1.0.0	Chad Ma	Initial version release
2018.10.16	1.0.1	Chad Ma	Add reset to factory mode chapter
2018.11.06	1.0.2	Chad Ma	Add SD card boot up upgrading chapter Add frequently asked questions summary in Appendix
2018.11.28	1.0.3	Chad Ma	Modify chapter 1.2 Add chapter 4.3.3 Add chapter 4.3.4
2018.12.18	1.0.4	Chad Ma	Modify chapter 1.3 Add chapter 2.2

Table of content

Chapter 1.	OTA Upgrading	1-6
1.1	Overview	1-6
1.2	Compilation	1-6
1.3	Upgrading process.....	1-7
1.4	Reset to factory mode.....	1-8
1.5	Notices	1-9
Chapter 2.	Debugging	2-10
2.1	Analyze log of Recovery mode.....	2-10
2.2	Device with/without panel	2-10
Chapter 3.	Upgrade through SD card boot up disk.....	3-12
3.1	Recovery supports SD card boot up upgrading.....	3-12
3.2	Make SD card boot up disk	3-12
Chapter 4.	Appendix.....	4-15
4.1	misc partition introduction.....	4-15
4.2	Usage of recovery in different cases.....	4-16
4.3	FAQ.....	4-18

Table of figures

Figure 1-1 recovery upgrading flow chart.....	1-8
Figure 2-1 create hidden file in recovery	2-10
Figure 2-2 recovery.mk configuration	2-11
Figure 3-1 make SD card boot up disk.....	3-12
Figure 4-2 the content of misc.img file	4-16
Figure 4-3 Recovery Makefile related.....	4-19
Figure 4-4 mkfirmware.sh	4-22
Figure 4-5 misc partition commonly used images	4-22

Chapter 1. OTA Upgrading

1.1 Overview

OTA (Over-the-Air) is space download technology. OTA upgrading is the standard software upgrading method provided by Android system. It is powerful and can upgrade system without loss, mainly through network such as WIFI, 3G/4G to automatically download OTA package and automatically upgrade, also support to download OTA package to SD card or U disk to upgrade. OTA package is very small, usually several M or more than a dozen M.

This document mainly introduces the local upgrading program recovery execution process and technical details when using OTA technology to upgrade, to help customers understand the upgrading process and notices during development.

1.2 Compilation

rootfs main system

Rootfs should enable recoverySystem, choose BR2_PACKAGE_RECOVERYSYSTEM=y in configs file, or configure BR2_PACKAGE_UPDATE=y.

If below error occurs after configuring BR2_PACKAGE_RECOVERYSYSTEM=y:

```
external/recoverySystem does not exist
package/pkg-generic.mk:194: recipe for target
rk3399/build/recoverySystem-develop/.stamp_rsynced' failed
make[1]: * [rk3399/build/recoverySystem-develop/.stamp_rsynced] Error 1
rk3399/Makefile:16: recipe for target '_all' failed
make: * [_all] Error 2
```

It means recoverySystem is not used in the project code, check if BR2_PACKAGE_UPDATE=y is configured or not.

```
rk3399/buildroot/configs/rockchip$ ag BR2_PACKAGE_RKUPDATE
recovery.config
3:BR2_PACKAGE_RKUPDATE=y
```

It means update package is already configured here.

Note:

Currently there are two set of code available to invoke upgrade function in main system, recoverySystem and update, and both can enter recovery mode using the same parameters to do OTA upgrade.

In the future we will unify to use update.

Recovery

Execute in system root directory

```
$ ./build.sh recovery
```

It will generate the file: buildroot/output/rockchip_rk3308_recovery/images/recovery.img.

```
$ ./mkfirmware.sh
```

And it will copy the generated image to the directory of rockdev/.

The path of the main source code related with recovery:

external/recovery/: mainly generate recovery binary files, the key program of recovery mode.

external/rkupdate/: mainly generate rkupdate binary files, parse each partition data of update.img and execute the key program upgrading each partition.

If the source files in above two directories are changed, the compiling method is:

1. Source envsetup.sh
2. select the recovery configuration for some platform
3. make recovery-rebuild
4. make rkupdate-rebuild
5. ./build.sh recovery
6. ./mkfirmware.sh
7. flash recovery.img

1.3 Upgrading process

Image upgrading preparation

Modify tools/linux/Linux_Pack_Firmware/rockdev/package-file

```
$ Vim tools/linux/Linux_Pack_Firmware/rockdev/package-file
```

Modify the package-file file according to the configuration of the partition to be upgraded.

Save the modification and exit to root directory to execute below command:

```
$ ./build.sh updateimg
```

After the command is executed successfully, it will put the generated update.img to the directory of rockdev/. Use this update.img to upgrade.

```
E:
=== umount userdata fail ===
>>>rkflash will update from /userdata/update.img
start with main.
OK
librkupdate_Start to upgrade firmware...
CRKImage :Error! imageHead.uiTag != 0x57464B52
librkupdate_ERROR:do_rk_firmware_upgrade-->new CRKImage failed!
librkupdate_Fail to upgrade firmware!
mkdir: can't create directory '/sys/kernel/config/usb_gadget/rockchip/functions/mass_storage.0': No such file or directory
/etc/init.d/S50usbdevice: line 46: can't create /sys/kernel/config/usb_gadget/rockchip/functions/mass_storage.0: No such file or directory
ln: /sys/kernel/config/usb_gadget/rockchip/configs/b.1/f1: No such file or directory
[ 3.992165] file system registered
install_listener('tcp:5037','*smartsocket*')
[ 4.002725] read descriptors
[ 4.002772] read strings
[ 4.745849] vendor_storage:20160801 ret = -1
```

If the error "Error! imageHead.uiTag !=0x57464B52" occurs during upgrading, it means there are errors with image package. Please follow the above steps to regenerate update.img and retry.

Upgrading process

- Put the upgrading image update.img in SD card or U disk root directory or /userdata directory of the device.
- Execute the upgrading program recoverySystem ota /xxx/update.img in normal system, the device will enter recovery mode and start upgrading.

The usable paths are as below:

U disk mount path: /udisk

sdcard mount path: /mnt/sdcard/ or /sdcard

flash mount path: /userdata/

- After upgrading successfully, the device will reboot to normal system.

Upgrading flow chart

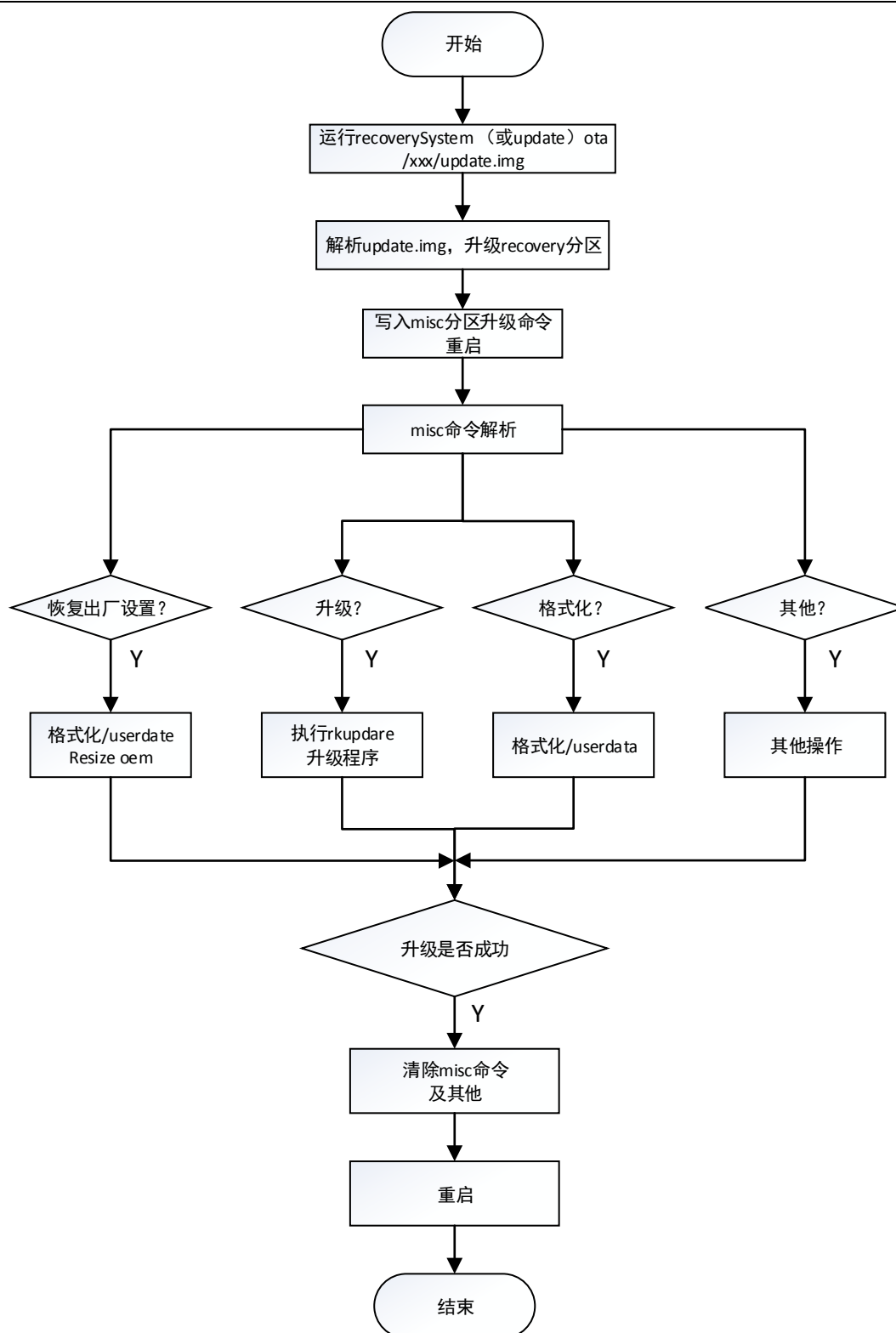


Figure 1-1 recovery upgrading flow chart

For more details about upgrading process, developers can refer to the source code of upgrading solution according to the flow chart. We will give no more elaboration here.

1.4 Reset to factory mode

We save the W/R configuration files in userdata partition. There are some default configuration parameters with factory image and users may generate or modify the configuration files after using for a period of time. Sometimes users want to erase these data,

and then we will need to recover it to factory configuration.

- Solution: generate userdata.img when packing image, and format userdata partition when user requests to reset to factory configuration.
- SDK implementation:
Function key RECOVERY + VOLUMEUP will trigger reset to factory mode. Code refers to:
buildroot/board/rockchip/rk3308/fs-overlay/etc/input-event-daemon.conf
board/rockchip/rk3308/fs-overlay/usr/sbin/factory_reset_cfg

Directly run recoverySystem (or update) without adding any parameter or adding factory/reset parameter can enter recovery to reset to factory mode.

1.5 Notices

- When packing update.img, need to pay attention to that, full partition upgrading is not necessary for image upgrading, you can change the package-file to remove the partition no need to upgrade, and in this way you can reduce the size of upgrading package (update.img).
- If recovery.img is packed in package-file, it will not upgrade in recovery mode, in order to prevent that other partitions cannot upgrade normally due to power down during recovery.img upgrading, this partition upgrading is done in normal system, that is, it will check if recovery.img is packed in update.img package before executing recoverySystem command, if yes, it will upgrade recovery partition and then enter recovery mode to upgrade other partition images.
- **Recommend not to pack misc partition into update.img.** Even if it is packed, it will also be checked and ignored while loading the upgrading program. The reason is: even if misc partition is upgraded, after upgrading successfully recovery program will still clear up all the commands and parameters in misc partition, as a result, the expected results cannot be achieved.
- If put update.img package in userdata partition of flash, need to ensure that userdata.img is not packed in package-file. Because it may cause the damage of the file system, and make oem or userdata partition fail to mount after upgrading successfully. If upgrading from SD card or U disk, userdata.img can be packed to update userdata partition. It will resize userdata partition after upgrading.

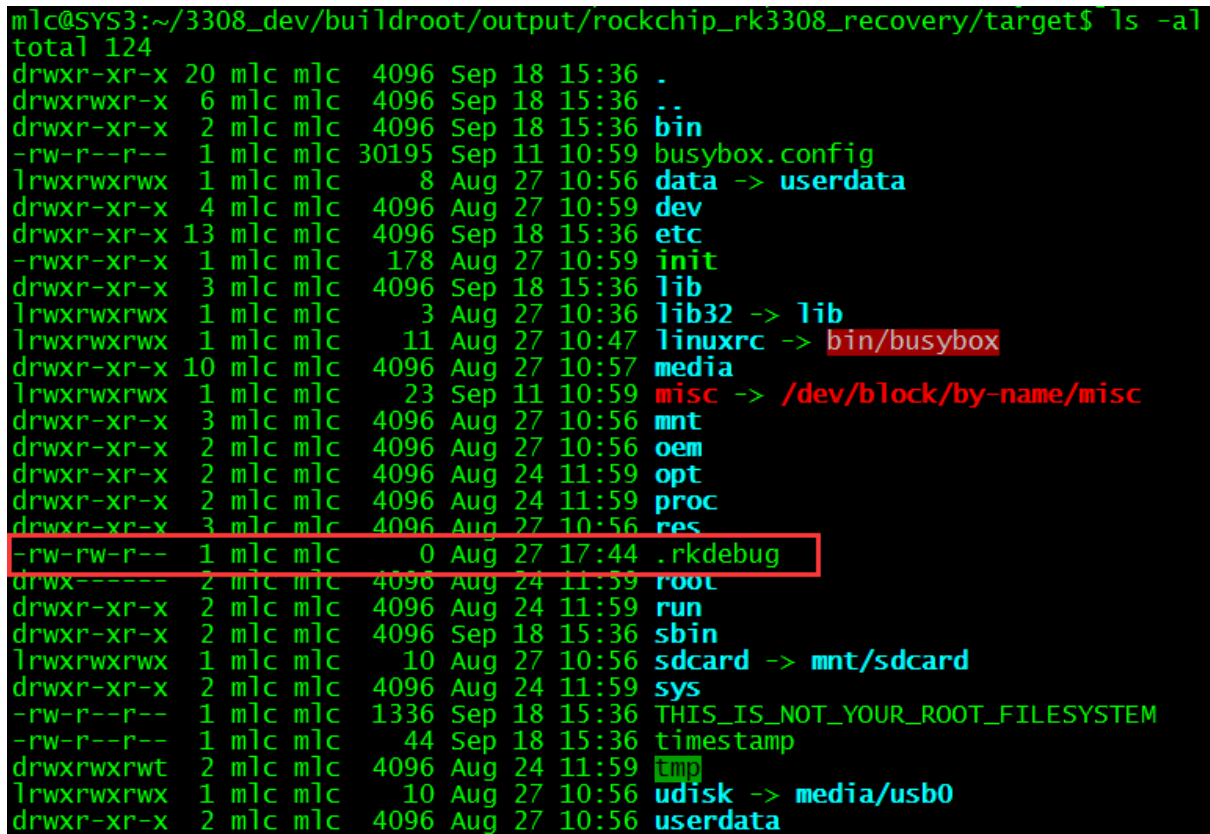
Chapter 2. Debugging

2.1 Analyze log of Recovery mode

- buildroot/output/rockchip_rk3308_recovery/target directory

```
$ touch .rkdebug
```

Create this hidden file can print out the upgrading log of recovery mode through serial port.



```
mlc@SYS3:~/3308_dev/buildroot/output/rockchip_rk3308_recovery/target$ ls -al
total 124
drwxr-xr-x 20 mlc mlc 4096 Sep 18 15:36 .
drwxrwxr-x 6 mlc mlc 4096 Sep 18 15:36 ..
drwxr-xr-x 2 mlc mlc 4096 Sep 18 15:36 bin
-rw-r--r-- 1 mlc mlc 30195 Sep 11 10:59 busybox.config
lrwxrwxrwx 1 mlc mlc 8 Aug 27 10:56 data -> userdata
drwxr-xr-x 4 mlc mlc 4096 Aug 27 10:59 dev
drwxr-xr-x 13 mlc mlc 4096 Sep 18 15:36 etc
-rwxr-xr-x 1 mlc mlc 178 Aug 27 10:59 init
drwxr-xr-x 3 mlc mlc 4096 Sep 18 15:36 lib
lrwxrwxrwx 1 mlc mlc 3 Aug 27 10:36 lib32 -> lib
lrwxrwxrwx 1 mlc mlc 11 Aug 27 10:47 linuxrc -> bin/busybox
drwxr-xr-x 10 mlc mlc 4096 Aug 27 10:57 media
lrwxrwxrwx 1 mlc mlc 23 Sep 11 10:59 misc -> /dev/block/by-name/misc
drwxr-xr-x 3 mlc mlc 4096 Aug 27 10:56 mnt
drwxr-xr-x 2 mlc mlc 4096 Aug 27 10:56 oem
drwxr-xr-x 2 mlc mlc 4096 Aug 24 11:59 opt
drwxr-xr-x 2 mlc mlc 4096 Aug 24 11:59 proc
drwxr-xr-x 3 mlc mlc 4096 Aug 27 10:56 res
-rw-rw-r-- 1 mlc mlc 0 Aug 27 17:44 .rkdebug
drwx----- 2 mlc mlc 4096 Aug 24 11:59 root
drwxr-xr-x 2 mlc mlc 4096 Aug 24 11:59 run
drwxr-xr-x 2 mlc mlc 4096 Sep 18 15:36 sbin
lrwxrwxrwx 1 mlc mlc 10 Aug 27 10:56 sdcard -> mnt/sdcard
drwxr-xr-x 2 mlc mlc 4096 Aug 24 11:59 sys
-rw-r--r-- 1 mlc mlc 1336 Sep 18 15:36 THIS_IS_NOT_YOUR_ROOT_FILESYSTEM
-rw-r--r-- 1 mlc mlc 44 Sep 18 15:36 timestamp
drwxrwxrwt 2 mlc mlc 4096 Aug 24 11:59 tmp
lrwxrwxrwx 1 mlc mlc 10 Aug 27 10:56 udisk -> media/usb0
drwxr-xr-x 2 mlc mlc 4096 Aug 27 10:56 userdata
```

Figure 2-1 create hidden file in recovery

- Check through userdata/recovery/Log file

After upgrading, check log files in the directory of userdata/recovery of the device.

```
$ cat userdata/recovery/Log
```

2.2 Device with/without panel

If failed during recovery execution and prompt below log:

```
failed to read font: res=-1, fall back to the compiled-in font
cannot find/open a drm device: No such file or directory
```

The log means cannot find or open a drm device, if the device is with panel, need to connect the panel, if not, need to do the following configurations.

It is the device without panel in recovery default configuration of SDK code.

```
buildroot/package/rockchip/recovery$ vim recovery.mk
```

Open recovery Makefile of buildroot/package/rockchip/recovery as shown below:

```
#####
#
# Rockchip Recovery For Linux
#
#####

RECOVERY_VERSION = develop
RECOVERY_SITE = $(TOPDIR)/../external/recovery
RECOVERY_SITE_METHOD = local

RECOVERY_LICENSE = Apache V2.0
RECOVERY_LICENSE_FILES = NOTICE
CC="$(TARGET_CC)"
PROJECT_DIR="$(@D)"
RECOVERY_BUILD_OPTS+=-I$(PROJECT_DIR) -I$(STAGING_DIR)/usr/include/libdrm \
    --sysroot=$(STAGING_DIR) \
    -fPIC \
    -lpthread

ifeq ($(BR2_PACKAGE_RK3308),y)
    TARGET_MAKE_ENV += RecoveryNoUi=true
else
    RECOVERY_BUILD_OPTS += -lz -lpng -ldrm
    RECOVERY_DEPENDENCIES += libzlib libpng libdrm
endif
```

Figure 2-2 recovery.mk configuration

As shown above, configure the target environment according to the specific chipset hardware platform.

Only RK3308 defines `RecoveryNoUi=true` by default, that means no display device is used. Other platforms use display device by default.

If the product of current chipset platform doesn't support display, need to add judgment according to below code, define `RecoveryNoUi=true`. Then it will compile the code without display in recovery code according to the macro definition of `RecoveryNoUi`.

```
ifeq ($(BR2_PACKAGE_RK3308),y)
    TARGET_MAKE_ENV += RecoveryNoUi=true
else
    RECOVERY_BUILD_OPTS += -lz -lpng -ldrm
    RECOVERY_DEPENDENCIES += libzlib libpng libdrm
endif
```

For example, if using RK3326 platform, modify `ifeq ($(BR2_PACKAGE_RK3308),y)` to `ifeq ($(BR2_PACKAGE_RK3326),y)`.

Compiling after modification:

1. source `envsetup.sh rockchip_xxxx_recovery` (xxxx is the specific chipset platform)
2. `make recovery-dirclean`
3. `make recovery-reconfigure`
4. `make recovery-rebuild`
5. `./build.sh recovery`
6. `./build.sh firmware`
7. flash `rockdev/recovery.img`

Chapter 3. Upgrade through SD card boot up disk

This chapter mainly describes how to make SD card boot up disk and relative upgrading issues in order to meet the upgrading requirement of bare chip using SD card to boot up.

3.1 Recovery supports SD card boot up upgrading

1. Firstly make sure the codes supporting SD card boot up upgrading are already added in the directories of external/recovery and external/rkupdate.

If the codes are not added, need to integrate the following patches:

Under the directory of external/recovery:

0001-recovery-add-support-sdcard-boot-update.patch

Under the directory of external/rkupdate:

0001-rkupdate-add-support-sdcard-boot-update.patch

2. Recompile recovery and rkupdate

```
$ make recovery-rebuild
```

```
$ make rkupdate-rebuild
```

3. Compile recovery

```
$ ./build.sh recovery
```

4. Generate the image

```
$ ./build.sh
```

3.2 Make SD card boot up disk

As shown in figure 3-1, use the tool in the project directory of tools\windows\SDDiskTool to make SD card boot up disk.

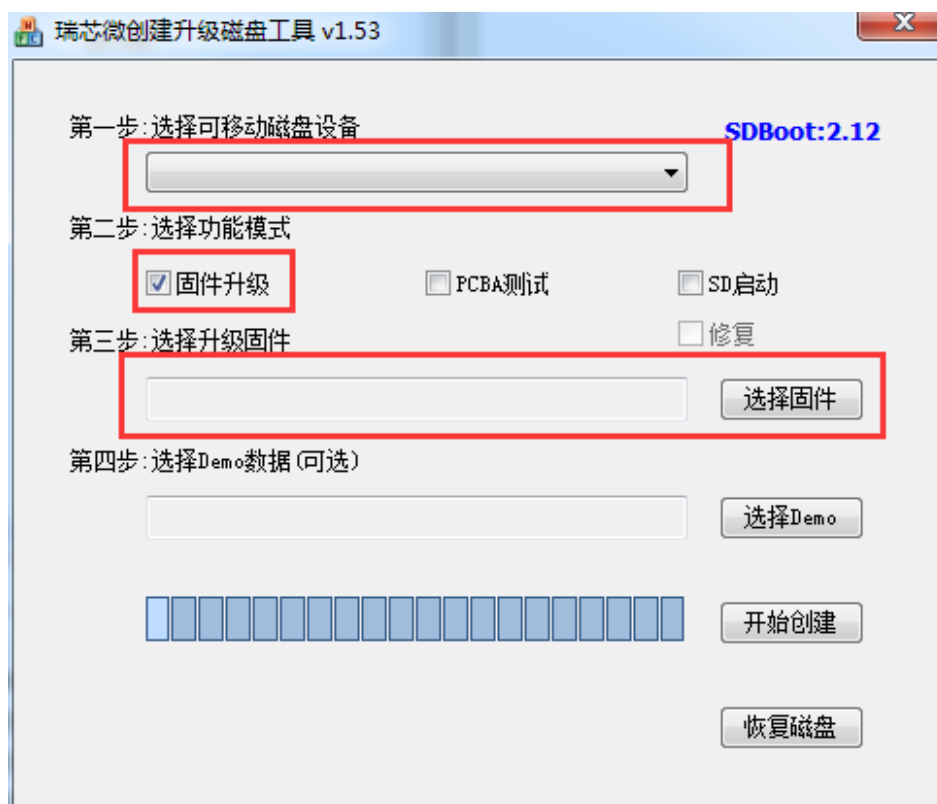


Figure 3-1 make SD card boot up disk

Select the packed update.img file in images choosing to upgrade.

After all the preparation work is done, click start to create, and it will prompt if creation is successful.

Now there are two files in the root directory of SD card, and the image choosing to upgrade update.img is renamed as sduupdate.img.

After all the preparation work is done, insert SD card to the device and then power on again.

If below information occurs in Log, it means the device is booted up by SD card successfully.

```
U-Boot 2017.09-g1bee468 (Oct 11 2018 - 16:53:06 +0800) V1.000
Model: USM-110 a102-1
Board:Advantech usm110_rk3288 Board,HW version:0
DRAM:  2 GiB
Relocation Offset is: 7ff5a000
PMIC:  RK808
vdd_arm 1100000 uV
vdd_gpu 1100000 uV
vcc_io 3300000 uV
regulator(LDO_REG2) init 3300000 uV
regulator(LDO_REG3) init 1100000 uV
regulator(LDO_REG4) init 1800000 uV
regulator(LDO_REG5) init 3300000 uV
regulator(LDO_REG6) init 1100000 uV
regulator(LDO_REG7) init 1800000 uV
regulator(LDO_REG8) init 1800000 uV
MMC:  dwmmc@ff0c0000: 1, dwmmc@ff0f0000: 0
SF: Detected w25q32bv with page size 256 Bytes, erase size 4 KiB, total 4 MiB
*** Warning - bad CRC, using default environment
In:    serial
Out:   serial
Err:   serial
switch to partitions #0, OK
mmc1 is current device
do_rking_test found IDB in SDcard
Boot from SDcard
enter Recovery mode!
SF: Detected w25q32bv with page size 256 Bytes, erase size 4 KiB, total 4 MiB
Skipped ethaddr assignment due to invalid,using default!
Net:   No ethernet found.
Hit any key to stop autoboot:  0
ANDROID: reboot reason: "recovery"
FDT load addr 0x10f00000 size 263 KiB
Booting kernel at 0x3575c70 with fdt at 42cf470...
```

If below log is printed out through serial port, it means the recovery image upgrading process of bare chip through SD card boot up is ongoing.

```
firmware update will from SDCARD.
is_sdcard_update out
```

```
sdupdate_package = /mnt/sdcard/sdupdate.img
Command: "/usr/bin/recovery"
>>>sdboot update will update from /mnt/sdcard/sdupdate.img
start with main.
librkupdate_Start to upgrade firmware...
librkupdate_INFO:is emmc devices...
librkupdate_INFO:CRKUsbComm-->is emmc.
librkupdate_INFO:CRKUsbComm-->/dev/vendor_storage=24
librkupdate_INFO:CRKUsbComm-->/dev/mmcblk2=26
librkupdate_uid: 52 4F 43 4B 43 48 49 50 45 EB C3 5B 6C 87 6F 6D
87 DA 4E 76 A1 FB 38 28 57 98 5C 8F B3 23
librkupdate_Get FlashInfo...
librkupdate_INFO:FlashInfo: 00 F8 E8 00 01 00 00 00 9C 51 E6
GetFlashInfo: 266 info.uiFlashSize = 15267840 total uiBlockNum = 30535680
GetFlashInfo: 267 FlashSize = 14910 MB
##### update bootloader start#####
librkupdate_IDBlock Preparing...
##### IDBlock Preparing...
librkupdate_ERROR:PrepareIDB-->New IDblock offset=0 1 2 3 4 .
librkupdate_IDBlock Writing...
##### IDBlock Writing...
librkupdate_INFO:MakeIDBlockData in
librkupdate_INFO:MakeIDBlockData out
librkupdate_INFO:WriteIDBlock in
librkupdate_INFO:-----
librkupdate_dwSectorNum=180
librkupdate_uiTotal=92160
librkupdate_INFO:WriteIDBlock out
##### update bootloader Suceess#####
librkupdate_##### RKA_Gpt_Download #####
librkupdate_##### Download trust ... #####
librkupdate_INFO:Start to download trust,offset=0x6000,size=4194304
librkupdate_##### Download uboot ... #####
librkupdate_INFO:Start to download uboot,offset=0x4000,size=4194304
librkupdate_##### Download misc ... #####
librkupdate_INFO:Start to download misc,offset=0x8000,size=49152
librkupdate_##### Download boot ... #####
librkupdate_INFO:Start to download boot,offset=0xa000,size=7774208
librkupdate_##### Download rootfs ... #####
librkupdate_INFO:Start to download rootfs,offset=0x5a000,size=2548436992
.....
```

Chapter 4. Appendix

4.1 misc partition introduction

Misc actually is the first four letters of miscellaneous, which means sundry, mixture and mess.

The concept of misc partition comes from Android system. It is usually used for system upgrading or resetting to factory configuration in Linux system.

The write/read of misc partition: misc partition will be written/read in the following cases:

1) Uboot: when the device is powered on, it will firstly boot up uboot, and read the content of misc partition in uboot. Then decide will enter normal system or recovery mode according to the command of misc partition.

If command is boot-recovery, it will enter recovery mode.

If command is empty, it will enter normal system.

2) Recovery: when the device enters recovery mode, it can read recovery part contents of misc partition, and then execute different actions such as upgrading or erasing user data and so on.

The structure and content of misc partition:

The structure of misc partition is shown as below:

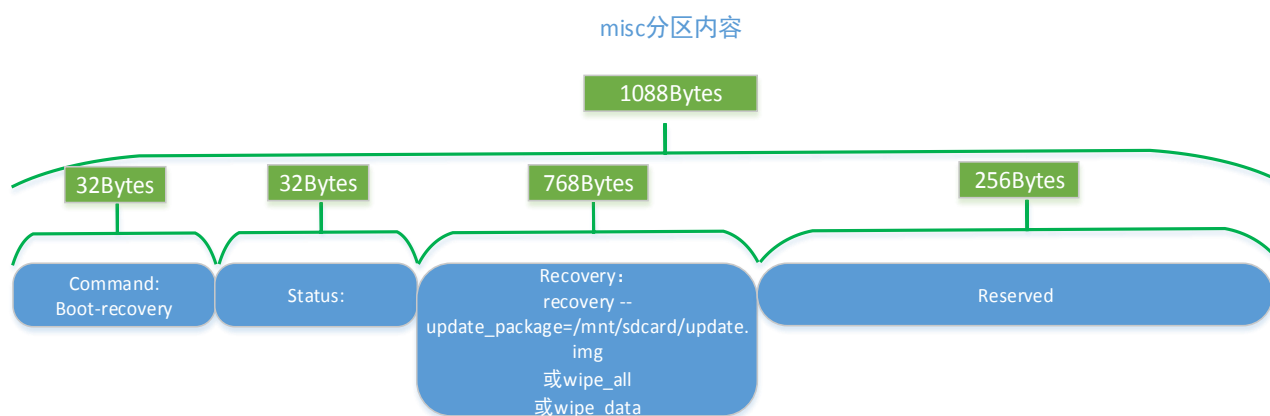


Figure 4-1 the structure of misc partition

Below take RK3308 misc partition as an example, use the tool such as winhex or ultraEdit etc. to open misc.img file with binary format, and save the content of struct BootLoader Msg starting at 16K (16384 Byte) byte offset from file start position

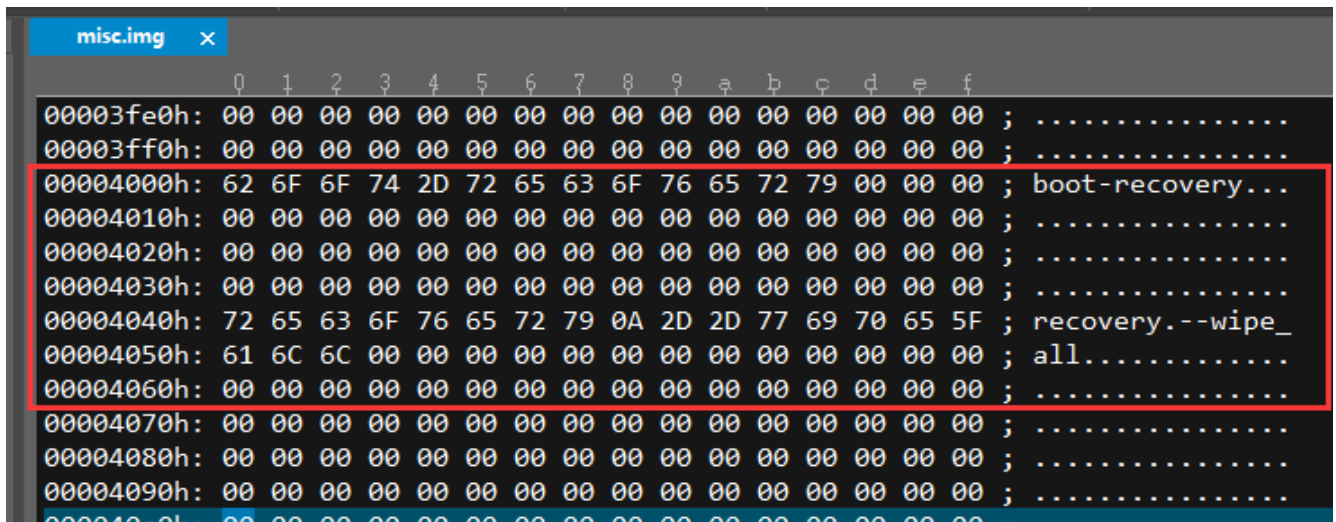


Figure 4-2 the content of misc.img file

For the supported commands in Recovery, you can refer to the contents of struct OPTIONS in external/recovery/recovery.c.

4.2 Usage of recovery in different cases

● First power up

The device with misc.img and recovery.img will enter the first power up process.

The serial port will print below log:

I:Boot command: boot-recovery

I:Got arguments from boot message

Command: "recovery" "--wipe_all"

format '/dev/block/by-name/userdata' to ext2 filesystem

executing '/sbin/mke2fs'

executed '/sbin/mke2fs' done

executed '/sbin/mke2fs' return 0

executing '/sbin/e2fsck'

e2fsck 1.43.9 (8-Feb-2018)

Pass 1: Checking inodes, blocks, and sizes

Pass 2: Checking directory structure

Pass 3: Checking directory connectivity

Pass 4: Checking reference counts

Pass 5: Checking group summary information

/dev/block/by-name/userdata: 11/2304 files (0.0% non-contiguous), 82/2299 blocks

executed '/sbin/e2fsck' done

executed '/sbin/e2fsck' return 0

executing '/usr/sbin/e2fsck'

e2fsck 1.43.9 (8-Feb-2018)

Pass 1: Checking inodes, blocks, and sizes

Pass 2: Checking directory structure

Pass 3: Checking directory connectivity

Pass 4: Checking reference counts

Pass 5: Checking group summary information

/dev/block/by-name/oem: 18/2448 files (0.0% non-contiguous), 513/16384 blocks


```

executed '/usr/sbin/e2fsck' done
executed '/usr/sbin/e2fsck' return 1
executing '/usr/sbin/resize2fs'
resize2fs 1.43.9 (8-Feb-2018)
The filesystem is already 16384 (1k) blocks long. Nothing to do!
executed '/usr/sbin/resize2fs' done
executed '/usr/sbin/resize2fs' return 0

```

- Reset to factory configuration

Run recoverySystem program in Command line, the device will enter recovery, start formatting, and it will automatically enter normal system after formatting is done.

```
$ recoverySystem
```

The serial port will print below log:

```

I:Boot command: boot-recovery
I:Got arguments from boot message
Command: "recovery" "--wipe_data"
format '/dev/block/by-name/userdata' to ext2 filesystem
executing '/sbin/mke2fs'
[ 4.692437] vendor storage:20160801 ret = -1
[ 6.030842] phy phy-ff008000.syscon:usb2-phy@100.0: charger =
USB_SDP_CHARGER
[ 10.891460] random: nonblocking pool is initialized
executed '/sbin/mke2fs' done
executed '/sbin/mke2fs' return 0
executing '/sbin/e2fsck'
e2fsck 1.43.9 (8-Feb-2018)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/block/by-name/userdata: 11/2304 files (0.0% non-contiguous), 82/2299 blocks
executed '/sbin/e2fsck' done
executed '/sbin/e2fsck' return 0
[ 11.141033] cpu0 limit freq=816000 min=816000 max=816000
[ 11.141773] rkndand_shutdown...
[ 11.142139] nand th quited
[ 11.142484] rk_ftl_de_init 0
[ 11.150824] rkndand_shutdown:OK
[ 11.152054] reboot: Restarting system

```

- upgrading

Run recoverySystem ota /xxx/update.img in command line, the device will enter recovery and start upgrading.

```
$ recoverySystem ota /udisk/update.img
```

Take upgrading from U disk as example:

The serial port may print below log:

I:Boot command: boot-recovery

I:Got arguments from boot message

Command: "recovery" "--update_package=/udisk/update.img"

o o o o

librkupdate_ui_print = parameter writing....

RKA_Gpt_Download

librkupdate_##### Download trust ... #####

ui_print = trust writing....

librkupdate_##### Download uboot ... #####

ui_print = uboot writing....

librkupdate_##### Download boot ... #####

librkupdate_ui_print = boot writing....

librkupdate_##### Download rootfs ... #####

librkupdate_ui_print = rootfs writing....

librkupdate_#####Ignore recovery download #####

librkupdate_##### Download oem ... #####

ui_print = oem writing....

librkupdate_##### Download userdata:grow ... #####

ui_print = parameter checking....

ui_print = trust checking....

ui_print = uboot checking....

ui_print = boot checking....

ui_print = rootfs checking....

ui_print = oem checking....

ui_print = userdata:grow checking....

librkupdate_##### Ignore recovery Check #####

librkupdate_Finish to upgrade firmware.

[38.655387] EXT2-fs (rknd0p8): warning: mounting unchecked fs, running e2fsck is recommended

[38.663735] cpu0 limit freq=816000 min=816000 max=816000

[38.664552] rknd_shutdown...

[38.664865] nand th quited

[38.665127] rk_ftl_de_init 0

[38.676436] rknd_shutdown:OK

[38.678078] reboot: Restarting system

4.3 FAQ

4.3.1 “cannot find/open a drm device”

For Non RK3308 platforms, after the device enters recovery mode, it is common to see below log printed from serial port:

we are in recovery, skip init oem/userdata

[start debug recovery...](#)

Starting recovery on Fri Jan 18 09:19:51 2013

failed to read font: res=-1, fall back to the compiled-in font

Starting network: cannot find/open a drm device: No such file or directory

In this case, the solution is to connect supported display panel or HDMI device to the device.

Analysis: log hints cannot find or open a drm device. Because for non RK3308 platforms the compiling of recovery program enables UI display by default, if it cannot open the display device after entering recovery mode, it will lead to recovery execution failure.

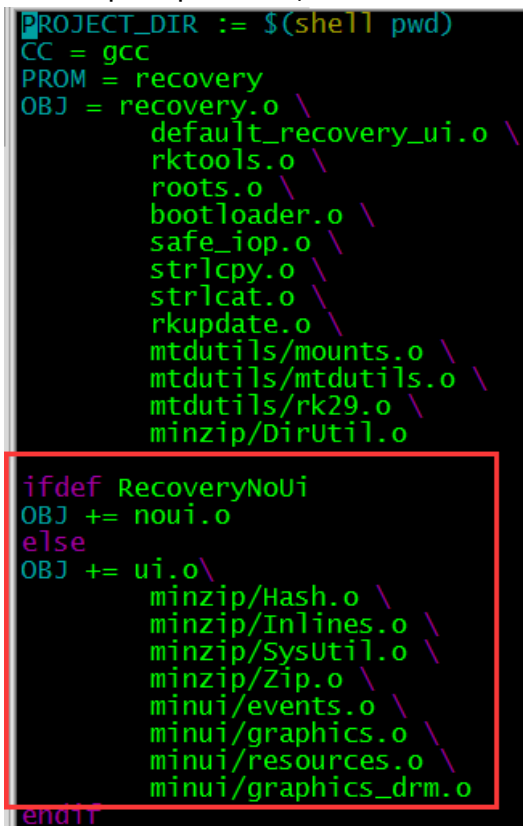
If you want to support recovery upgrading feature without display for non RK3308 platforms such as RK3399, RK3288 etc., users can operate according to below method:

1、Modify mk file of recovery

```
vim buildroot/package/rockchip/recovery/? recovery.mk
```

```
ifeq ($(BR2_PACKAGE_RK3308),y)
    TARGET_MAKE_ENV += RecoveryNoUi=true
else
    RECOVERY_BUILD_OPTS += -lz -lpng -ldrm
    RECOVERY_DEPENDENCIES += libzlib libpng libdrm
endif
```

Referring to above, per its own chipset platform, define RecoveryNoUi=true.



```
PROJECT_DIR := $(shell pwd)
CC = gcc
PROM = recovery
OBJ = recovery.o \
    default_recovery_ui.o \
    rktools.o \
    roots.o \
    bootloader.o \
    safe_iop.o \
    strlcpy.o \
    strlcat.o \
    rkupdate.o \
    mtdutils/mounts.o \
    mtdutils/mtdutils.o \
    mtdutils/rk29.o \
    minzip/DirUtil.o

ifdef RecoveryNoUi
OBJ += noui.o
else
OBJ += ui.o \
    minzip/Hash.o \
    minzip/Inlines.o \
    minzip/SysUtil.o \
    minzip/Zip.o \
    minui/events.o \
    minui/graphics.o \
    minui/resources.o \
    minui/graphics_drm.o
endif
```

Figure 4-3 Recovery Makefile related

Then it will not compile display related code in external/recovery/Makefile as shown in Figure 4-3.

2、Recompile recovery

```
make recovery-rebuild
```

3、Regenerate recovery image

```
./build.sh recovery
4、Generate image.
./mkfirmware.sh
```

4.3.2 “E:Can't open /dev/block/by-name/misc”

When use recovery function to upgrade image, the serial port may print below log while using emmc flash:

```
start debug recovery...
Starting recovery on Thu Jan  1 00:00:01 1970

recovery filesystem table
=====
0 (null) /tmp ramdisk (null) (null) (null)
1 /dev/root / ext2 rw,noauto 0 1
2 proc /proc proc defaults 0 0
3 devpts /dev/pts devpts defaults,gid=5,mode=620 0 0
4 tmpfs /dev/shm tmpfs mode=0777 0 0
5 tmpfs /tmp tmpfs mode=1777 0 0
6 tmpfs /run tmpfs mode=0755,nosuid,nodev 0 0
7 sysfs /sys sysfs defaults 0 0
Starting network:  8 debug /sys/kernel/debug debugfs defaults 0 0
9 pstore /sys/fs/pstore pstore defaults 0 0
10 /dev/block/by-name/misc /misc emmc defaults 0 0
11 /dev/block/by-name/oem /oem ext2 defaults 0 2
12 /dev/block/by-name/userdata /userdata ext2 defaults 0 2

emmc_point is
sd_point is (null)
sd_point_2 is (null)
buf = /dev/block/by-name/misc
### get mount_point = /dev/block/by-name/misc ###
===path = /misc, v-mount_point = /misc ===
E:Can't open /dev/block/by-name/misc
(No such file or directory)
No link to path!!!
===path = /userdata/recovery/command, v-mount_point = /userdata ===
E:failed to mount /userdata (No such file or directory)
E:Can't mount /userdata/recovery/command
buf = /dev/block/by-name/misc
### get mount_point = /dev/block/by-name/misc ###
===path = /misc, v-mount_point = /misc ===
E:Can't open /dev/block/by-name/misc
(No such file or directory)
```

Command: "/usr/bin/recovery"

In this case, the solution is to integrate below patch.

Analysis: the log hints cannot open misc partition after entering recovery mode. Because misc partition is soft linked to by-name corresponding partition after udev async initialization finishes, mmc recognition is asynchronous, if mmc initialization is not finished when recovery starts, it will fail to open by-name of msic.

Only need to add wait_for_device operation for the function get_bootloader_message_block in Recovery/bootloader.c.

```
diff --git a/bootloader.c b/bootloader.c
index fbf01ab2..7681507c 100644
--- a/bootloader.c
+++ b/bootloader.c
@@ -30,7 +30,7 @@ static int set_bootloader_message_block(const struct
bootloader_message *in, con

int get_bootloader_message(struct bootloader_message *out) {
    Volume* v = volume_for_path("/misc");
-
+
    if(!v) return -1;
    if (strcmp(v->fs_type, "mtd") == 0) {
        return get_bootloader_message_mtd(out, v);
@@ -133,9 +133,26 @@ static int set_bootloader_message_mtd(const struct
bootloader_message *in,
    // -----
    // for misc partitions on block devices
    // -----
+static void wait_for_device(const char* fn) {
+    int tries = 0;
+    int ret;
+    struct stat buf;
+    do {
+        ++tries;
+        ret = stat(fn, &buf);
+        if (ret) {
+            printf("stat %s try %d: %s\n", fn, tries, strerror(errno));
+            sleep(1);
+        }
+    } while (ret && tries < 10);
+    if (ret) {
+        printf("failed to stat %s\n", fn);
+    }
+}

static int get_bootloader_message_block(struct bootloader_message *out,
```

```

const Volume* v) {
+   wait_for_device(v->device);
   FILE* f = fopen(v->device, "rb");
   if (f == NULL) {
       LOGE("Can't open %s\n(%s)\n", v->device, strerror(errno));
   }
--

```

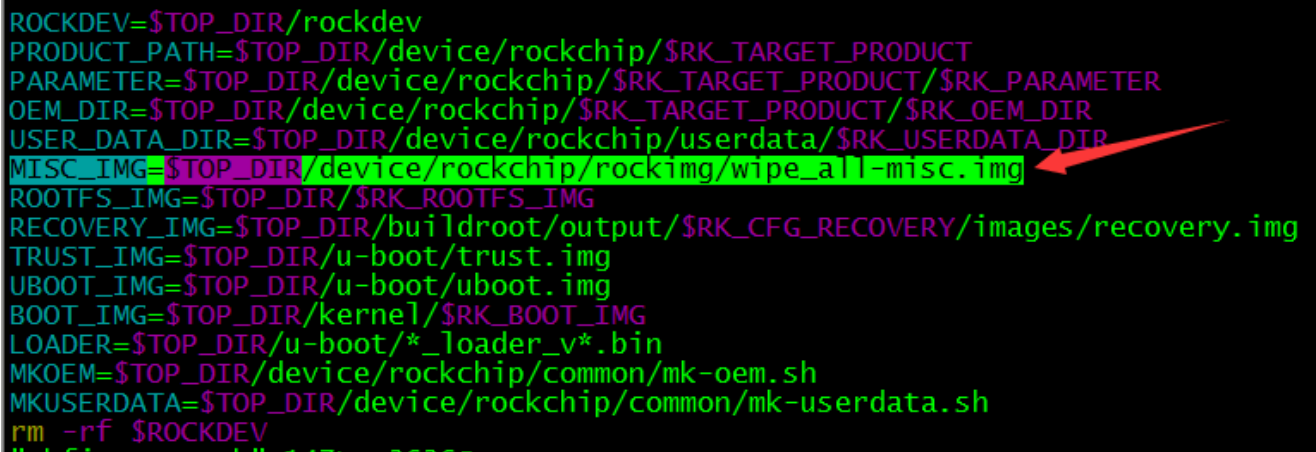
4.3.3 Change default command of misc partition image

If want to pack the different recovery commands in changing misc image, or use empty misc partition image.

You can change in the following way:

Modify mkfirmware.sh in the root directory of project.

```
$ vim mkfirmware.sh
```



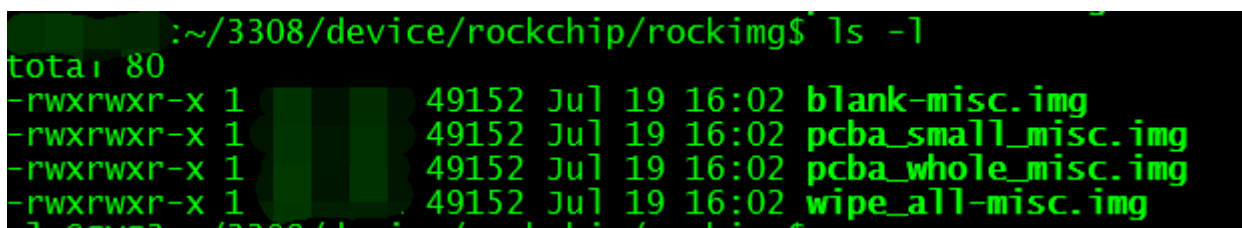
```

ROCKDEV=$TOP_DIR/rockdev
PRODUCT_PATH=$TOP_DIR/device/rockchip/$RK_TARGET_PRODUCT
PARAMETER=$TOP_DIR/device/rockchip/$RK_TARGET_PRODUCT/$RK_PARAMETER
OEM_DIR=$TOP_DIR/device/rockchip/$RK_TARGET_PRODUCT/$RK_OEM_DIR
USER_DATA_DIR=$TOP_DIR/device/rockchip/userdata/$RK_USERDATA_DIR
MISC_IMG=$TOP_DIR/device/rockchip/rockimg/wipe_all-misc.img
ROOTFS_IMG=$TOP_DIR/$RK_ROOTFS_IMG
RECOVERY_IMG=$TOP_DIR/buildroot/output/$RK_CFG_RECOVERY/images/recovery.img
TRUST_IMG=$TOP_DIR/u-boot/trust.img
UBOOT_IMG=$TOP_DIR/u-boot/uboot.img
BOOT_IMG=$TOP_DIR/kernel/$RK_BOOT_IMG
LOADER=$TOP_DIR/u-boot/*_loader_v*.bin
MKOEM=$TOP_DIR/device/rockchip/common/mk-oem.sh
MKUSERDATA=$TOP_DIR/device/rockchip/common/mk-userdata.sh
rm -rf $ROCKDEV

```

Figure 4-4 mkfirmware.sh

There are commonly used misc partition image files in the directory of device/rockchip/rockimg.



```

~/3308/device/rockchip/rockimg$ ls -l
total 80
-rwxrwxr-x 1  49152 Jul 19 16:02 blank-misc.img
-rwxrwxr-x 1  49152 Jul 19 16:02 pcba_small_misc.img
-rwxrwxr-x 1  49152 Jul 19 16:02 pcba_whole_misc.img
-rwxrwxr-x 1  49152 Jul 19 16:02 wipe_all-misc.img

```

Figure 4-5 misc partition commonly used images

4.3.4 set userdata partition as vfat file system

SDK default userdata partition is ext2/ext4 file system, if want to change it to vfat32 file system, you can do as below:

1. Change `board/rockchip/rk3308/fs-overlay-recovery/etc/fstab`

Before change:

/dev/block/by-name/userdata	/userdata	ext2	defaults0	0
-----------------------------	-----------	------	-----------	---

After change:

/dev/block/by-name/userdata	/userdata	vfat	defaults0	0
-----------------------------	-----------	------	-----------	---

2. Change `configs/rockchip_rk3308_release_defconfig`

Add below configuration:

```
BR2_PACKAGE_DOSFSTOOLS=y
BR2_PACKAGE_DOSFSTOOLS_FATLABEL=y
BR2_PACKAGE_DOSFSTOOLS_FSCK_FAT=y
BR2_PACKAGE_DOSFSTOOLS_MKFS_FAT=y
```

3. Change `package/rockchip/usbdevice/S50usbdevice`

start)

```
mkdir /dev/usb-ffs -m 0770
mkdir /dev/usb-ffs/adb -m 0770
mount -t configfs none /sys/kernel/config
mkdir /sys/kernel/config/usb_gadget/rockchip -m 0770
echo 0x2207 > /sys/kernel/config/usb_gadget/rockchip/idVendor
echo "ums_adb" >
```

```
/sys/kernel/config/usb_gadget/rockchip/configs/b.1/strings/0x409/configuration
```

```
mount -t functionfs adb /dev/usb-ffs/adb
mount -t vfat /dev/disk/by-partlabel/userdata /media/usb0
export service_adb_tcp_port=5555
adb &
sleep 1
```

4. Make sure below patch is already updated.

0001-common-mk-userdata-Fix-wrong-FS_TYPE-check.patch

```
case $FS_TYPE in
    ext[2-4])
        $COMMON_DIR/mke2img.sh $USERDATA_DIR $USERDATA_IMG
        ;;
    fat|vfat)
        SIZE=$(du -h -BM --max-depth=1 $USERDATA_DIR|awk '{print int($1)}')
        # echo "create image size=${SIZE}M"
        dd if=/dev/zero of=$USERDATA_IMG bs=1M count=$SIZE >/dev/null 2>&1
        mkfs.vfat $USERDATA_IMG >/dev/null 2>&1
        mcopy -i $USERDATA_IMG $USERDATA_DIR/* ::/ >/dev/null 2>&1
        ;;
    *)
        echo "file system: $FS_TYPE not support."
        exit 1
        ;;
Esac
```

5. Change `common/mk-userdata.sh`

```
SIZE=$(du -h -BM --max-depth=1 $USERDATA_DIR|awk '{print int($1)}')
```

```
# echo "create image size=${SIZE}M"
dd if=/dev/zero of=$USERDATA_IMG bs=1M count=$SIZE >/dev/null 2>&1
mkfs.vfat -F 32 $USERDATA_IMG >/dev/null 2>&1
mcopy -i $USERDATA_IMG $USERDATA_DIR/* ::/ >/dev/null 2>&1
;;
*)
```

6. Change **rk3308/BoardConfig.mk**

```
# Set userdata partition type, including ext2, fat
export RK_USERDATA_FS_TYPE=ext2
export RK_USERDATA_FS_TYPE=vfat
```