Linux on NAND

From Linux Rockchip

Contents

Important considerations

Why NAND?

Please be aware NAND memory is NOT the fastest! Please see this Memory Speed Test.

Using NAND is OK in case of:

- minimal power consumption is important
- no external connections are allowed and micro SD card is not preferred
- for academical reason

Using NAND should be avoided because:

- likely to be slower than micro SD class10 card
- backups are uncomfortable
- in case of problem with data storage -device is not bootable anymore- the recovery procedure is nearly impossible (for your data) without backup

General NAND partitioning

Before one gets down to the task of bringing the SD card system on NAND, you should think about the future disposition of the NAND memory. This corresponds to the partition considerations before installing a Linux system on the hard disk. Mandatory are always areas for the *parameter* file (4MB) for "boot" (16MB) and the kernel (8- 16MB). The remaining area can be divided freely.

A 1GB swap partition may be used and the rest (about 6.5 GB) as a Linux root filesystem partition ("rfs"). Another option is to use the whole remaining space for the "rfs" and use a swap file within it (http://www.cyberciti.biz/faq/linux-add-a-swap-file-howto/) . The advantage of this is that you can easily and immediately change its size at any time, as opposed to a partition.

Open Source rknand driver

Note that as of 2014-03-01, `drivers/mtd/rknand` driver in apparently all open source Rockchip kernels is incomplete; a self-built kernel using this driver will be unable to further make use of the NAND after boot. As a workaround, the proprietary `rk30xxnand.ko` can be extracted from your stick's Android image and inserted by your kernel's ramdisk. Galland's rk30_linux_initramfs repository (https://github.com/Galland/rk30_linux_initramfs) contains such a ramdisk ready made.

parameter file format

The boot loader needs to know the partitions, their locations and sizes in NAND. Those settings are specified in the *parameter* file. The following example assumes a 16MiB kernel partition.

```
FIRMWARE_VER:4.0.5
MACHINE_MODEL:rk30sdk
MACHINE_ID:007
MANUFACTURER:RK30SDK
MAGIC: 0x5041524B
ATAG: 0x60000800
MACHINE: 3066
CHECK_MASK: 0x80
KERNEL_IMG: 0x60408000
#RECOVER_KEY: 1,1,0,20,0
CMDLINE:initrd=0x62000000,0x00800000 root=LABEL=linuxroot init=/sbin/init \
mtdparts=rk29xxnand:0x00008000@0x00002000(boot),0x00008000@0x0000A000(kernel),\
0x00200000@0x00012000(swap),-@0x00212000(linux)
```

Since this information is flashed in the first area of the NAND memory, all future flash operations can determine the correct destination (start address and block size) for the respective flash partition.

Note

This section:

```
root=LABEL=linuxroot
```

means that the "rfs" can be found on a partition with the name (label) "linuxroot". But it seems that the *parameter* file does not get analyzed here. This means it has a fallback mechanism: "look for "rfs" at the specified start address (0x00212000) or just use the

kernel entry". Reasoning: If you use an SD card with "rfs" this is the desired behavior. In NAND "rfs" things are different, but this same *parameter* file will still be working - it does not have to be overwritten again just for NAND "rfs" usage.

Kernel cmdline configuration

All partitions in the NAND have to be configured in the *parameter* file and at the same time on the kernel cmdline. The kernel cmdline syntax is shown in the following example: it specifies the same entries/partitions as given in the above *parameter* file.

```
 CONFIG_CMDLINE="root=/dev/mtdblock3 init=/sbin/init loglevel=8 rootfstype=ext4 rootwait \
mtdparts=rk29xxnand:0x00008000@0x00002000(boot),0x00008000@0x0000A000(kernel),\
0x00200000@0x00012000(swap),-@0x00212000(linux)"
```

The cmdline could also look like the the following example. Some partitions are omitted. One reason might be that the Linux system just does not need access to the other partitions. Another reason might be that access/knowledge to those partitions shall be prevented for the Linux system.

```
 CONFIG_CMDLINE="root=/dev/mtdblock1 init=/sbin/init loglevel=8 rootfstype=ext4 rootwait \
mtdparts=rk29xxnand:0x00200000@0x00012000(swap),-@0x00212000(linux)"
```

Preparations

Linux *parameter* file

Copy an existing file and adjust / replace the *CMDLINE* entry (or use the one described above).

```
CMDLINE:initrd=0x62000000,0x00800000 root=LABEL=linuxroot init=/sbin/init \
mtdparts=rk29xxnand:0x00008000@0x00002000(boot),0x00008000@0x0000A000(kernel),\
0x00200000@0x00012000(swap),-@0x00212000(linux)
```

Note:

This *parameter* file is tailored to the sdcard boot type, but it can equally be used to boot from NAND (no further adjustment necessary).

In this case the name of the *parameter* file is **parameter_1GBSwap.txt**.

Linux kernel cmdline configuration

First, build the sdcard kernel. Then configure the kernel cmdline:

```
CONFIG_CMDLINE="root=LABEL=linuxroot init=/sbin/init loglevel=8 rootfstype=ext4 rootwait \
mtdparts=rk29xxnand:0x00008000@0x00002000(boot),0x00008000@0x0000A000(kernel),\
0x00200000@0x00012000(swap),-@0x00212000(linux)"
```

(Untested: can it possibly be flashed later in NAND? Edit: what is meant with this note?)

The name for the kernel image is now **kernel_00_1GBswap_SD.img**

Flash Linux using Windows

The following files are needed using a Windows flash tool (f.e. ROM Flash Tool v1.3.7).

- "Loader" (RK3188Loader(L)_V1.20.bin) --> extracted from existing image or downloaded
- *parameter* file (parameter_1GBSwap.txt) --> your own
- boot-image (i.e. linuxium-boot.img) --> downloaded
- sdcard-kernel (kernel_00_1GBswap_SD.img) --> your own

How to flash:

1. Copy the files to an NTFS partition or a USB stick. Boot into Windows and make the NTFS partition or USB stick available.
2. Connect the TV stick (i.e. rk3188 device) in "flash mode".
3. Open the flash tool. The bottom line should say: "Found RKAndroid rock usb" or similar.
4. Press the "EraseIDB" button and wait.
5. Select the files and press "run":

```
* "Loader"    (RK3188Loader (L ) _V1.20.bin)
* parameter (parameter_1GBSwap.txt)
* "Boot"      (linuxium-boot.img)
* "Kernel"    (kernel_00_1GBswap_SD.img)
```

Once the flash process finishes successfully the stick should automatically boot. The boot process will stop in "emergency mode".

Flash Linux using Linux

See Flash Linux using Linux (http://radxa.com/Rock/flash_the_image#Linux%7C) .

Copy Linux kernel and root filesystem from sdcard to NAND

Boot TV stick from sdcard

Shut down TV stick, insert sdcard into the TV stick's sdcard slot and boot from sdcard. The boot process should not get stuck in "emergency mode" now, but continue to boot from sdcard.

Now you can boot the PC back into Linux.

Check MTD devices..

Either use a monitor and a keyboard or connect via ssh to your TV stick. Log in and execute:

```
fdisk -l /dev/mtdblock0 # boot (4MiB)
fdisk -l /dev/mtdblock1 # kernel (16MiB)
fdisk -l /dev/mtdblock2 # swap (1GiB)
```

```
fdisk -l /dev/mtdblock3 # for the Linux root filesystem (6-7GiB)
swapon -s # show active swap
```

If these commands terminate successfully the partitions for linux and swap can be created.

.. and create partitions on them

Create the Linux root filesystem partition and swap:

```
mkfs.ext4 /dev/mtdblock3
mkswap /dev/mtdblock2
swapon /dev/mtdblock2
```

Prepare swap

Before you start copying delete any swap file that might exist (copy process is faster then) and instead provide the new NAND swap partition immediately:

```
swapoff -a
rm -f /swapfile1
```

Edit */etc/fstab*:

```
vi /etc/fstab

#/swapfile1 swap swap defaults 0 0
/dev/mtdblock2 swap swap defaults 0 0
```

Copy over the sdcard root filesystem to NAND

Creating a mount point on the sdcard and mount blank NAND partition:

```
mkdir /mnt/rfs/
mount /dev/mtdblock3 /mnt/rfs
```

Copy the "rfs" from sdcard to NAND:

```
cd /
cp -ax $(ls | egrep -v "proc|run|sys$") /mnt/rfs
mkdir /mnt/rfs/{proc,run,sys}
chmod 555 /mnt/rfs/proc
umount /mnt/rfs
```

Now shut down the TV stick and remove the sdcard.

Finally create and flash the NAND kernel

```
CONFIG_CMDLINE="root=/dev/mtdblock3 init=/sbin/init loglevel=8 rootfstype=ext4 rootwait \
mtdparts=rk29xxnand:0x00008000@0x00002000(boot),0x00008000@0x0000A000(kernel),\
```

```
0x00200000@0x00012000(swap),-@0x00212000(linux)"
```

The name of your NAND kernel is now **kernel_00_1GBswap_NAND.img**.

Start TV stick in "flash mode" (Linux). Run "rkflashkit -master" and flash your new NAND kernel and reboot afterwards.

Alternative: Flash a complete system image file

Using Windows, it is also possible to flash a complete system image (.img) to the system partition (Edit: which is the correct partition to use? Make sure to set the correct start address.

When flashing a complete system image file to NAND we have to consider that the size of the image file does correspond to the size of the root filesystem contained within. The size of the image file is (hopefully) smaller than the available free NAND memory. This means that the resulting root filesystem will only use it's original capacity leaving available NAND capacity unused. Also, the swap partition of file has to be checked/created.

A very nice solution to this situation can be found in the "initramfs" (inside *boot.img*) of the *DX05* system image file dx05_nand_ubuntu_1.1.1.7z (http://www.freaktab.com/showthread.php?6803-Linux-for-RK3188-based-TV-boxes%7C) : create a file named */firstrun* and the system will enlarge the NAND partition to the maximum available after first boot up.

See the following script excerpt:

```
if [ -f /new_root/firstrun ] ; then
  echo "Running for first time, preparing filesystem"
  umount /new_root
  /sbin/e2fsck -f -y $rootfs
  /sbin/resize2fs -p $rootfs
  /bin/mount -t ext4 $rootfs /new_root
  rm /new_root/firstrun
  # create swap partition
  swap_mtd=`cat /proc/mtd | grep swap | cut -c4`
  if [ "$swap_mtd" != "" ] ; then
    swappart="/dev/mtdblock"$swap_mtd
    /bin/mknod -m 660 $swappart b 31 $swap_mtd
    mkswap $swappart
    cat /new_root/etc/fstab | grep -v swap > /new_root/etc/fstab.1
    rm /new_root/etc/fstab
    mv /new_root/etc/fstab.1 /new_root/etc/fstab
    echo "$swappart none swap defaults 0 0" >> /new_root/etc/fstab
  fi
fi
```

For using this you might have to modify your *boot.img* (i.e. *linuxium-boot.img*). Some system image files already have this functionality included.

Addendum

- See also http://hwswbits.blogspot.de/2013/11/your-own-official-linux-distro-in-sd.html (http://hwswbits.blogspot.de/2013/11/your-own-official-linux-distro-in-sd.html%7C)

Category: Tutorial
