Dual Image on EnGenius Designed ARM-based Product Families

Chin-Ting Chu | Software Engineer

ABSTRACT—This document contains information on the changes required to support dual images in the bootloader level. Dual image is one workable idea to provide alternating firmware between two firmware partitions. When the current firmware is broken, we reset the kernel entry and boot it up. Most importantly, we do the firmware upgrade in the inactive partition. If there is nothing wrong with it, the upgrade process will swap the kernel entry. Therefore, we can keep one bootable firmware in our device all the time.

I - Introduction

The bootloader does some low-level hardware initialization, such as processor, memory, ethernet, flash, PCI, UART, and so on. After setting system configurations, the bootloader loads the kernel and passes startup information including serial port speeds, clock rates, the table of partitions and other hardware configuration data.

When the kernel is loaded, it configures the hardware, allocates the memory, loads the drivers, inserts the modules, mounts the root filesystem, and runs the pre-init processes. The final step is spawning the init process, the first user space process. The three types on the ARM-based OpenWRT are NOR-only, NAND-only and NOR plus-NAND.

Fig.1. NOR only

Fig.2. NAND only

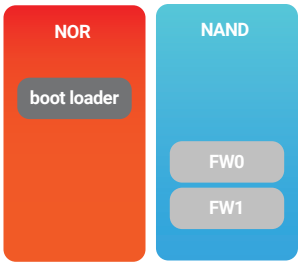

Fig.3. NOR and NAND

II - Procedure

I. NOR flash only

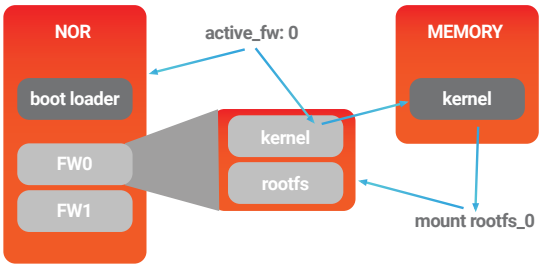

Fig.4. Use the 1st firmware – NOR only

1. Bootloader : When the power on, the bootloader will check the dual image flag, “active_fw”

    1.1. Append additional partitions to mtdparts : kernel_1 and rootfs_1

    1.2. Load the target kernel to the memory

    1.3. Assign a active root filesystem partition by setting the bootargs.

    1.3.1. If “active_fw=0” => root=rootfs

    1.3.2. If “active_fw=1” => root=rootfs_1

    1.4. If boot kernel fail, swap active firmware partition

2. Kernel Space : parses the bootargs assigned by bootloader, and mount the root filesystem.

3. User space : when system upgrade

    3.1. If “active_fw=1” =>

    3.1.1. flash kernel and rootfs

    3.1.2. set active_fw=0

    3.1.3. reboot

    3.2. If “active_fw=0” =>

3.2.1. flash kernel_1 and rootfs_1

3.2.2. set active_fw=0

3.2.3. reboot

II.. NAND flash only

The procedure is the same as 'NOR flash only'.

III.. NOR + NAND

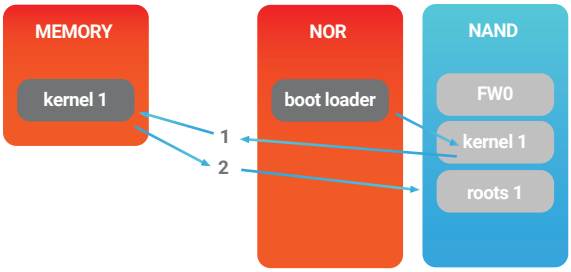

Fig.5. Use the 2nd firmware – NOR+NAND

1. Bootloader : When the power on, the bootloader will check the dual image flag, "active_fw"

1.1. Append two NAND partitions named rootfs and fs2

1.1.1 If "active_fw=0" =>

The upper half of NAND is named rootfs and the another one is named fs2.

1.1.2 If "active_fw=1" =>

The upper half of NAND is named fs2 and the another one is named rootfs.

1.2. Load the target kernel to the memory

1.3. If boot kernel fail, swap active firmware partition

2. Kernel Space : Under the NOR+NAND structure, we always use the partition named rootfs as the root filesystem.

3. User space : when system upgrade
The procedure is the same as 'NOR flash only'.

III - Implement

MTD partitions: two equal parts for two firm-wares

II. Dual image flag: set a active firmware partition

III. System upgrade: Always upgrade the inactive firmware partition

IV. Auto recovery: If the uboot finds the boot fail from this firmware partition, active the another firmware partition.

A. NOR flash only

```
[    1.439237] 0x000000000000-0x000000020000 : "SBL1"
[    1.446547] 0x000000020000-0x000000040000 : "MIBIB"
[    1.452827] 0x000000040000-0x000000080000 : "SBL2"
[    1.459169] 0x000000080000-0x000000100000 : "SBL3"
[    1.465823] 0x000000100000-0x000000110000 : "DDRCONFIG"
[    1.472446] 0x000000110000-0x000000120000 : "SSD"
[    1.478225] 0x000000120000-0x0000001a0000 : "TZ"
[    1.483880] 0x0000001a0000-0x000000220000 : "RPM"
[    1.489721] 0x000000220000-0x0000002a0000 : "APPSBL"
[    1.495813] 0x0000002a0000-0x0000002e0000 : "APPSBLENV"
[    1.502155] 0x0000002e0000-0x000000320000 : "ART"
[    1.507966] 0x000000320000-0x000000520000 : "kernel"      FIRMWARE 1
[    1.513995] 0x000000520000-0x000001020000 : "rootfs"
[    1.520181] mtd: partition "rootfs" set to be root filesystem
[    1.525335] mtd: partition "rootfs_data" created automatically, ofs=C90000, len=390000
[    1.532927] 0x000000c90000-0x000001020000 : "rootfs_data"
[    1.540549] 0x000001020000-0x000001220000 : "kernel_1"    FIRMWARE 2
[    1.546766] 0x000001220000-0x000001d20000 : "rootfs_1"
```

Fig.9. MTD partition table - NOR only

1. Create two firmware partitions

1.1. modify the nor-partition.xml to tell the bootloader the new partitions info

1.2. modify the sys_flash_map.mk to tell the kernel mtd partition table

2. Dual image flag

2.1. Modify board.c to add "active_fw" to uboot de-fault env

2.2. Modify the cmd_bootipq.c to swap the boot args(root= rootfs or root= rootfs_1).

3. System upgrade

3.1. check "active_fw" and flash the target firmware partition@ipq806x.sh

3.1.1. "active_fw=1" => flash kernel and rootfs

3.1.2. "active_fw=0" => flash kernel_1 and rootfs_1

4. Auto recovery

4.1. Modify the board.c to make sure the uboot catch-es the return value from the boot function

B. NAND flash only

The procedure is the same as 'NOR flash only'.

C. NOR + NAND

```
[    1.420524] Creating 2 MTD partitions on "msm_nand":
[    1.424679] 0x000000000000-0x000004000000 : "rootfs"
[    1.579318] mtd: partition "rootfs" set to be root filesystem
[    1.584879] split_squashfs: no squashfs found in "msm_nand"
[    1.589659] 0x000004000000-0x000008000000 : "fs2"
[    1.695095] ata1: SATA link down (SStatus 0 SControl 300)
[    1.749859] m25p80 spi5.0: found s25fl256s1, expected s25fl512s
[    1.754764] m25p80 spi5.0: s25fl256s1 (32768 Kbytes)
[    1.760387] Creating 11 MTD partitions on "m25p80":
[    1.764573] 0x000000000000-0x000000020000 : "SBL1"
[    1.771977] 0x000000020000-0x000000040000 : "MIBIB"
[    1.778381] 0x000000040000-0x000000080000 : "SBL2"
[    1.784661] 0x000000080000-0x000000100000 : "SBL3"
[    1.790940] 0x000000100000-0x000000110000 : "DDRCONFIG"
[    1.797250] 0x000000110000-0x000000120000 : "SSD"
[    1.802999] 0x000000120000-0x0000001a0000 : "TZ"
[    1.808716] 0x0000001a0000-0x000000220000 : "RPM"
[    1.814495] 0x000000220000-0x0000002a0000 : "APPSBL"
[    1.820618] 0x0000002a0000-0x0000002e0000 : "APPSBLENV"
[    1.827054] 0x0000002e0000-0x000000320000 : "ART"
```

Fig.11. MTD partition table when active_fw=0 - NOR+NAND

1. Create two firmware partitions
    1.1. Modify bootipq.c to add a partitions named fs2
    1.2. Modify bootm.c to pass the atags partition info. to kernel
2. Dual image flag
    2.1. add “active_fw” to uboot default env
3. System upgrade
    3.1. check “active_fw” and flash the target firmware
    partition@ipq806x.sh
    3.1.1. Flash the fs2 partition
4. Auto recovery
    4.1. Modify the board.c to make sure the uboot catch-es the return value from the boot function