AM335x UBI Filesystem 制作以及 NandFlash 烧录手册

V1.1

Mile

2012/9/26

目录

UBI Filesystem 制作

编译 mtd-utils

软件环境：

Ubuntu 11.04
am335x-evm-sdk-05.05.00.00

获取源码：

可通过以下链接下载 MTD Utils：
http://download.chinaunix.net/download.php?id=33753&ResourceID=13056

或通过 Git 获取源码：

```
git://git.infradead.org/mtd-utils.git
```

推荐使用 mtd-utils 1.4.8 以上

编译 mtd-utils 之前，需先安装以下工具：

1. zlib
2. lzo
3. e2fsprogs

zlib 和 e2fsprogs Ubuntu 中已经自带了，所以只需安装 lzo。注意，安装时需连接网络。

打开终端，输入：

```
sudo apt-get install liblzo2-2
sudo apt-get install liblzo2-dev
sudo apt-get install uuid-dev
```

安装完毕后，开始编译 mtd-utils。

打开终端，输入：

```
cd [your base directory]/mtd-utils-1.5.0
make WITHOUT_XATTR=1
```

至此，mtd-utils 编译完成。

制作 UBI Filesystem

打开终端，输入：

```
cd [your base dirctory]/mtd-utils-1.5.0
mkfs.ubifs/mkfs.ubifs -r [your filesystem directory]/ -F -o ubifs.img -m 2048 -e 126976 -c 1580
```

其中，[your filesystem directory]/ 换成你的文件系统文件夹，建议先使用 TI 官方的 base-rootfs-am335x-evm.

各个参数的详细说明(来自 TI Wiki: http://processors.wiki.ti.com/index.php/UBIFS_Support#Creating_UBIFS_file_system)：

-m 2KiB (or 2048)

The minimum I/O size of the underlying UBI and MTD devices. In our case, we are running the flash with no sub-page writes, so this is a 2KiB page.

-e 124KiB (or 126976)

Erase Block Size: UBI requires 2 minimum I/O units out of each Physical Erase Block (PEB) for overhead: 1 for maintaining erase count information, and 1 for maintaining the Volume ID information. The PEB size for the XO flash is 128KiB, so this leads to each Logical Erase Block (LEB) having 124KiB available for data.

-c 1580

The maximum size, in LEBs, of this file system. See calculation below for how this number is determined.

-r filesystem

Use the contents of the 'filesystem/' directory to generate the initial file system image.

-F

File-system free space has to be fixed up on first mount (http://www.linux-mtd.infradead.org/faq/ubifs.html#L_free_space_fixup)

-o ubifs.img

Output file.

执行完毕后会在当前目录下生成 ubifs.img.

在终端继续输入：

```
gedit ubinize.cfg
```

在打开的编辑器中输入以下内容：

```
[ubifs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=192MiB
vol_type=dynamic
vol_name=rootfs
```

```
vol_flags=autoresize
```

保存后退出，在终端输入：

```
ubi-utils/ubinize -o ubi.img -m 2048 -p 128KiB -s 512 -O 2048 ubinize.cfg
```

参数介绍：

Where:

-o ubi.img

Output file

-m 2KiB (or 2048)

Minimum flash I/O size of 2KiB page

-p 128KiB

Size of the physical eraseblock of the flash this UBI image is created for

-O 2048 offset if the VID header from start of the physical eraseblock

The output of the above command, '**ubi.img**' is the required image.

完成后，会在当前目录下生成 ubi.img，这个就是我们需要烧录到 Flash 中的文件。

烧录 Nandflash

配置 Source Code

U-Boot：

虽然官方说支持 NandFlash 启动，但在 U-Boot 04.06.00.08 的 SPL 代码中没有添加对 boot mode 的选择，仅仅默认从 SD 卡启动，因此需修改 [your u_boot base directory]\arch\arm\cpu\armv7\omap-common\boot-common.c 文件，从第 38 行改为：

```
/* Modified by WPI Mile Tang */
u32 omap_bootmode[] = {
		MMCSD_MODE_FAT,
		NAND_MODE_HW_ECC,
	};
/* End */
u32 omap_boot_device(void)
```

```
{
        return (u32) (boot_params.omap_bootdevice);
}
/* Modified by WPI Mile Tang */
u32 omap_boot_mode(void)
{
        u32 boot_device = omap_boot_device();
        switch(boot_device){
        case BOOT_DEVICE_NAND:
                return omap_bootmode[1];
        case BOOT_DEVICE_MMC1:
        case BOOT_DEVICE_MMC2:
        default:
                return omap_bootmode[0];
        }
}
/* End */
```

修改完后重新编译。

Linux Kernel:

由于 SDK 是 3.2 的内核，需要打以下补丁，关于该补丁的描述参见 https://patchwork.kernel.org/patch/1245721/，并且该补丁已融入 3.4.7 内核中 http://www.kernel.org/pub/linux/kernel/v3.x/ChangeLog-3.4.7：

```
--- a/fs/ubifs/sb.c
+++ b/fs/ubifs/sb.c
@@ -718,8 +718,12 @@ static int fixup_free_space(struct ubifs_info *c)
                lnum = ubifs_next_log_lnum(c, lnum);
        }

-        /* Fixup the current log head */
-        err = fixup_leb(c, c->lhead_lnum, c->lhead_offs);
+         /*
+          * Fixup the log head which contains the only a CS node at the
+          * beginning.
+          */
+         err = fixup_leb(c, c->lhead_lnum,
+                         ALIGN(UBIFS_CS_NODE_SZ, c->min_io_size));
        if (err)
                goto out;
```

如果你使用的是 WPI 的 EVM，在 Linux kernel 中需修改文件 [your kernel base directory]\arvh\arm\mach-omap2\board-am335xevm.c：

```
/* General Purpose EVM */
static struct evm_dev_cfg gen_purp_evm_dev_cfg[] = {
        {enable_ecap0,       DEV_ON_DGHTR_BRD, (PROFILE_0 | PROFILE_1 |
                                        PROFILE_2 | PROFILE_7) },
        {lcdc_init,DEV_ON_BASEBOARD, (PROFILE_0 | PROFILE_1 |
                                        PROFILE_2 | PROFILE_7) },
        {tsc_init,  DEV_ON_BASEBOARD, (PROFILE_0 | PROFILE_1 |
                                        PROFILE_2 | PROFILE_7) },
        {rgmii1_init,    DEV_ON_BASEBOARD, PROFILE_ALL},
        {rgmii2_init,    DEV_ON_DGHTR_BRD, (PROFILE_1 | PROFILE_2 |
                                        PROFILE_4 | PROFILE_6) },
        {usb0_init,      DEV_ON_BASEBOARD, PROFILE_ALL},
        {usb1_init,      DEV_ON_BASEBOARD, PROFILE_ALL},
        {evm_nand_init, DEV_ON_DGHTR_BRD,
                (PROFILE_ALL & ~PROFILE_2 & ~PROFILE_3)},
        {i2c1_init,        DEV_ON_BASEBOARD, (PROFILE_ALL & ~PROFILE_2)},
        {mcasp1_init,  DEV_ON_BASEBOARD, (PROFILE_0 | PROFILE_3 | PROFILE_7)},
        {mmc1_init,    DEV_ON_BASEBOARD, PROFILE_2},
        {mmc2_wl12xx_init,      DEV_ON_BASEBOARD, (PROFILE_0 | PROFILE_3 |
                                                PROFILE_5)},
        {mmc0_init,    DEV_ON_BASEBOARD, (PROFILE_ALL & ~PROFILE_5)},
        {mmc0_no_cd_init,DEV_ON_BASEBOARD, PROFILE_5},
        {spi0_init,      DEV_ON_DGHTR_BRD, PROFILE_2},
        {uart1_wl12xx_init,DEV_ON_BASEBOARD, (PROFILE_0 | PROFILE_3 |
                                                PROFILE_5)},
        {wl12xx_init,  DEV_ON_BASEBOARD, (PROFILE_0 | PROFILE_3 | PROFILE_5)},
        {d_can_init,   DEV_ON_DGHTR_BRD, PROFILE_1},
        {matrix_keypad_init, DEV_ON_BASEBOARD, PROFILE_0},
        {volume_keys_init,    DEV_ON_BASEBOARD, PROFILE_0},
        {uart2_init,     DEV_ON_DGHTR_BRD, PROFILE_3},
        {haptics_init,  DEV_ON_DGHTR_BRD, (PROFILE_4)},
        {NULL, 0, 0},
};
```

高亮处修改为

```
        {evm_nand_init, DEV_ON_BASEBOARD,
                (PROFILE_ALL & ~PROFILE_2 & ~PROFILE_3)},
```

修改之后重新编译 Kernel。

烧录文件：

这里介绍通过 SD 卡烧录 NandFlash 的方法：
首先制作 SD 启动卡，请参考
http://software-dl.ti.com/dsps/dsps_public_sw/am_bu/sdk/AM335xSDK/latest/exports/sitara-linuxsdk-sdg-05.05.00.00.pdf

然后将 ubi.img 也拷贝到 SD 卡 boot 分区，启动开发板，停在 u-boot 阶段，在串口终端中输入：

```
U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 MLO
U-Boot# nandecc hw 2
U-Boot# nand erase 0x0 0x20000
U-Boot# nand write.i 0x82000000 0x0 0x20000
U-Boot# mw.b 0x82000000 0 0x20000

U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 u-boot.img
U-Boot# nandecc hw 2
U-Boot# nand erase 0x80000 0x40000
U-Boot# nand write.i 0x82000000 0x80000 0x40000
U-Boot# mw.b 0x82000000 0 0x40000

U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 uImage
U-Boot# nandecc hw 2
U-Boot# nand erase 0x00280000 0x00500000
U-Boot# nand write.i 0x82000000 0x00280000 0x500000
U-Boot# mw.b 0x82000000 0 0x500000

U-Boot# mmc rescan
U-Boot# fatload mmc 0 0x82000000 ubi.img
U-Boot# nandecc hw 2
U-Boot# nand erase 0x780000 0xf880000
U-Boot# nand write.i 0x82000000 0x780000 0x1200000(此处参数根据你的
                                                  ubi.img 大小设置)

U-Boot# setenv bootcmd ‘run nand_boot’
U-Boot# boot
```

此时即可成功启动 Linux。

参考链接：
AM335x U-Boot User’s Guide：
http://processors.wiki.ti.com/index.php/AM335x_U-Boot_User%27s_Guide

Creating UBIFS file system：
http://processors.wiki.ti.com/index.php/UBIFS_Support#Creating_UBIFS_file_system