

再谈SD卡分区与引导

准备

- **扇区大小**:SD卡的扇区大小一般为 512 字节, SD卡的分区都是以扇区为单位的

从之前的文章 [EasyArm IMX280A SD分区详解.pdf](#) 我们已经知道了, 要想从SD卡启动, 该SD卡必须使用 MBR 格式的分区表, MBR 记录了该SD上各个分区的地址, 大小和文件系统类型等信息。其中, 启动分区的文件系统标志 必须为 0x53。

在启动分区的第一个扇区的开始位置必须包含 BCB(Boot Control Block) 数据, 该 BCB 数据指明了启动镜像(u-boot.sb)的地址和大小。

在Windows下使用 `cfimager.exe` 制作SD卡启动盘的时候, 改工具自动帮我们在SD卡上创建了3个分区:

- 一个FAT32分区(10), 用于存放 uImage, zImage 和设备树文件;
- 一个 rootfs 分区(b), 用于存放根文件系统;
- 一个 uboot 分区(53), 用于存放 BCB 和 u-boot.sb 镜像, 该分区为启动引导分区(0x53);

那么我们可不可以自己手动给SD分区, 制作可启动的SD卡呢? 当然是可以的, 在linux系统上, 只需要使用 `fdisk`, `mkfs`, `dd`, `mxsboot`(uboot自带的tools) 这几个工具就可以制作一个可启动的SD卡。

如何分区

u-boot默认的启动参数如下(`printenv`,已忽略nand部分):

```
1  baudrate=115200
2  boot_fdt=try
3  bootcmd=mmc dev ${mmcdev}; if mmc rescan; then if run loadbootscript; then
    run bootscript; else if run loadimage; then run mmcboot; else run netboot;
    fi; fi; else run netboot; fi
4  bootscript=echo Running bootscript from mmc ...; source
5  console_mainline=ttyAMA0
6  fdt_addr=0x41000000
7  fdt_file=imx28-evk.dtb
8  image=zImage
9  loadaddr=0x42000000
10 loadbootscript=fatload mmc ${mmcdev}:${mmcpart} ${loadaddr} ${script};
11 loadfdt=fatload mmc ${mmcdev}:${mmcpart} ${fdt_addr} ${fdt_file}
12 loadimage=fatload mmc ${mmcdev}:${mmcpart} ${loadaddr} ${image}
13 mmcargs=setenv bootargs console=${console_mainline},${baudrate}
    root=${mmccroot}
14 mmcboot=echo Booting from mmc ...; run mmcargs; if test ${boot_fdt} = yes
    || test ${boot_fdt} = try; then if run loadfdt; then bootz ${loadaddr} -
    ${fdt_addr}; else if test ${boot_fdt} = try; then bootz; else echo WARN:
    Cannot load the DT; fi; fi; else bootz; fi;
15 mmcdev=0
16 mmcpart=2
17 mmccroot=/dev/mmcblk0p3 rw rootwait
18 script=boot.scr
```

把bootcmd展开如下:

```
1 mmc dev ${mmcdev};
2 if mmc rescan; then
3     if run loadbootscript; then
4         run bootscrip;
5     else
6         if run loadimage; then
7             run mmcboot;
8         else
9             run netboot;
10        fi;
11    fi;
12 else
13     run netboot;
14 fi
```

把mmcboot展开如下:

```
1 echo Booting from mmc ...;
2 run mmcargs;
3 if test ${boot_fdt} = yes || test ${boot_fdt} = try; then
4     if run loadfdt; then
5         bootz ${loadaddr} - ${fdt_addr};
6     else
7         if test ${boot_fdt} = try; then
8             bootz;
9         else
10            echo WARN: Cannot load the DT;
11        fi;
12    fi;
13 else
14     bootz;
15 fi;
```

loadbootscript表示从SD卡加载启动脚本 boot.scr,这里可以忽略.最关键的地方就是

loadimage和mmcboot

loadimage表示从SD卡的 mmcdev:mmcpart ==> 0:2 分区上加载 zImage 到内存的 0x42000000 地址上, 也就是说 zImage 保存在SD卡的 第二分区 上, fatload 表明了该分区是一个fat32分区

```
1 mmcdev=0
2 mmcpart=2
3 image=zImage
4 loadaddr=0x42000000
5 loadimage=fatload mmc ${mmcdev}:${mmcpart} ${loadaddr} ${image}
```

loadfdt 同样表示从SD卡的第二分区上加载设备树文件 imx28-evk.dtb 到内存的 0x41000000 地址上,最后通过 bootz 命令启动内核

```

1 mmcdev=0
2 mmcpart=2
3 fdt_addr=0x41000000
4 fdt_file=imx28-evk.dtb
5 loadfdt=fatload mmc ${mmcdev}:${mmcpart} ${fdt_addr} ${fdt_file}

```

由此可见,按照uboot默认的方式, SD卡的第二分区应该是FAT32分区.FAT32分区可以直接在Windows系统上读写,这样做的好处是,如果改变了uboot,zImage或设备树,不需要重新烧写SD卡,只要用新的文件替换SD卡上的文件即可.

kernel的 bootargs 启动参数如下:

```

1 run mmcargs
2 mmcargs=setenv bootargs console=${console_mainline},${baudrate}
  root=${mmccroot}
3 mmccroot=/dev/mmcblk0p3 rw rootwait

```

其中 root=/dev/mmcblk0p3 rw rootwait, 由此可见, rootfs应该放在SD卡的 第三分区 上.

mxsboot

剩下的, u-boot.sb 镜像自然应该放在SD卡的第一分区上,并且该分区应该是启动引导分区,必须包含 BCB 信息.

这个BCB信息应该怎么生成呢?这里就必须要用到官方提供的 mxsboot 这个工具了.

这个工具会生成一个分区镜像,在分区镜像的头部填充 BCB 信息,并且把 u-boot.sb 镜像放入该分区镜像的相应位置.

使用时,只需要用以下命令就可以生成一个包含BCB和u-boot镜像的启动分区镜像 u-boot.sd,可以用dd命令直接把 u-boot.sd 写入启动分区中.

```

1 $ ./tools/mxsboot sd u-boot.sb u-boot.sd
2 $ dd if=u-boot.sd of=/dev/mmcblk0p1

```

mxsboot的源码在 tools\mxsboot.c 中,其关键代码如下:

mxsboot.c

```

1  /*
2   * Sector on which the SigmaTel boot partition (0x53) starts.
3   */
4  static uint32_t sd_sector = 2048;
5
6  struct mx28_sd_drive_info {
7      uint32_t      chip_num;
8      uint32_t      drive_type;
9      uint32_t      tag;
10     uint32_t      first_sector_number;
11     uint32_t      sector_count;
12 };
13
14 struct mx28_sd_config_block {
15     uint32_t      signature;
16     uint32_t      primary_boot_tag;
17     uint32_t      secondary_boot_tag;

```

```

18     uint32_t          num_copies;
19     struct mx28_sd_drive_info  drv_info[1];
20 };
21
22 static int mx28_create_sd_image(int infd, int outfd)
23 {
24     int ret = -1;
25     uint32_t *buf;
26     int size;
27     off_t fsize;
28     ssize_t wr_size;
29     struct mx28_sd_config_block *cb;
30
31     fsize = lseek(infd, 0, SEEK_END);
32     lseek(infd, 0, SEEK_SET);
33     size = fsize + 4 * 512; //u-boot.sb的大小+4个扇区的BCB信息
34
35     buf = malloc(size);
36     if (!buf) {
37         printf("Can not allocate output buffer of %d bytes\n", size);
38         goto err0;
39     }
40
41     //把u-boot.sb放入分区镜像的第四个扇区之后的位置(从第五个扇区开始),这里是相对地址
42     ret = read(infd, (uint8_t *)buf + 4 * 512, fsize);
43     if (ret != fsize) {
44         ret = -1;
45         goto err1;
46     }
47
48     //前四个扇区用于存放BCB,这里是相对地址
49     cb = (struct mx28_sd_config_block *)buf;
50
51     cb->signature = cpu_to_le32(0x00112233);
52     cb->primary_boot_tag = cpu_to_le32(0x1);
53     cb->secondary_boot_tag = cpu_to_le32(0x1);
54     cb->num_copies = cpu_to_le32(1);
55     cb->drv_info[0].chip_num = cpu_to_le32(0x0);
56     cb->drv_info[0].drive_type = cpu_to_le32(0x0);
57     cb->drv_info[0].tag = cpu_to_le32(0x1);
58     //u-boot.sb的绝对扇区号 : 分区起始扇区号(sd_sector) + 偏移BCB大小(分区前4扇区)
59     cb->drv_info[0].first_sector_number = cpu_to_le32(sd_sector + 4);
60     cb->drv_info[0].sector_count = cpu_to_le32((size - 4) / 512);
61
62     wr_size = write(outfd, buf, size);
63     if (wr_size != size) {
64         ret = -1;
65         goto err1;
66     }
67
68     ret = 0;
69
70 err1:
71     free(buf);
72 err0:
73     return ret;
74 }

```

由上可知,默认情况下,mxsboot认为uboot启动分区的起始扇区号是 2048 ,而 u-boot.sb 镜像的位置会在此基础上再偏移 4个扇区 ,也就是 2052 号扇区.

实际上,我们在使用fdisk给SD卡分区的时候,默认的第一个分区也是从SD卡的第 2048 号扇区开始的.SD卡的前部预留了 2048 个扇区的大小(共 2048*512 字节).

uboot env

不知道你有没有注意到一个问题,为什么没有考虑uboot的环境变量呢?它也是保存再SD卡上的,但是我们为什么没有为它预留一部分空间呢?

是的没错,我们目前为止的确没考虑到这一点,那让我们看看uboot中是把环境变量保存在哪个位置的,在 configs/mx28evk_defconfig#L7 中可以找到如下定义:

```
1 CONFIG_ENV_SIZE=0x4000
2 CONFIG_ENV_OFFSET=0x40000
```

这里的单位是字节,而不是扇区了,由此可见,uboot中把环境变量保存在SD卡从0地址开始的第 0x40000 == 262,144 字节处,大小为 0x4000 == 16,384 字节.

上面我们说过,使用fdisk对SD卡分区的时候,会在SD前部预留2048个扇区,也就是 2048*512=1,048,576 字节,uboot的环境变量刚好就放在这部分预留的空间上

因此不必担心,uboot已经为我们考虑的很周到了.

分区表

通过以上分析,我们已经弄清楚了SD卡的分区情况,总结如下:

分区	ID	作用	大小
第一分区	0x53	UBOOT启动分区	1M
第二分区	0x10	FAT32分区	100M
第三分区	0x0b	ROOTFS分区	SD卡剩余空间

给SD卡分区

```
1 $ sudo fdisk /dev/${sdcard}
2 * o ..... create a clear partition table
3 * n ..... create new partition
4 * p ..... primary partition
5 * 1 ..... first partition(uboot)
6 * (default) ..... first sector is 2048
7 * +1M ..... make the partition 1Mb big
8 * n ..... create new partition
9 * p ..... primary partition
10 * 2 ..... second partition(fat32)
11 * (default) ..... first sector is 2048(default)
12 * +100M ..... make the partition 100Mb big
13 * n ..... create new partition
```

```

14      * p ..... primary partition
15      * 3 ..... third partition
16      * (default) ..... first sector is 2048(default)
17      * (default) ..... use all remaining space
18      * t ..... change partition ID
19      * 1 ..... change first partition ID
20      * 53 ..... change the ID to 0x53 (OnTrack DM6 Aux3)
21      * t ..... change partition ID
22      * 2 ..... change second partition ID
23      * 10 ..... change the ID to 0x10 (win95 FAT32)
24      * t ..... change partition ID
25      * 3 ..... change third partition ID
26      * b ..... change the ID to 0x0b (OPUS)
27      * w ..... write partition table to disk

```

烧写并启动

```

1 $ mkfs.fat /dev/${sdcard}2
2 $ mkdir /tmp/fat32
3 $ mount /dev/${sdcard}2 /tmp/fat32
4 $ cp uImage /tmp/fat32/
5 $ cp zImage /tmp/fat32/
6 $ cp imx28-evk.dtb /tmp/fat32/
7 $ dd if=u-boot.sd of=/dev/${sdcard}1
8 $ dd if=rootfs.full.img of=/dev/${sdcard}3

```

bootlog

```

1 U-Boot 2020.07-rc4 (Jul 04 2020 - 00:14:52 +0800)
2
3 CPU:   Freescale i.MX28 rev1.2 at 454 MHz
4 BOOT:  SSP SD/MMC #0, 3V3
5 DRAM:  64 MiB
6 NAND:  use legacy bch geometry
7 128 MiB
8 MMC:   MXS MMC: 0
9 Loading Environment from MMC... *** Warning - bad CRC, using default
  environment
10
11 In:    serial
12 Out:   serial
13 Err:   serial
14 Net:   FEC0 [PRIME]
15 Hit any key to stop autoboot:  0
16 switch to partitions #0, OK
17 mmc0 is current device
18 5300920 bytes read in 1600 ms (3.2 MiB/s)
19 Booting from mmc ...
20 22122 bytes read in 41 ms (526.4 KiB/s)
21 Kernel image @ 0x42000000 [ 0x000000 - 0x50e2b8 ]
22 ## Flattened Device Tree blob at 41000000
23    Booting using the fdt blob at 0x41000000
24    Loading Device Tree to 43b39000, end 43b41669 ... OK
25

```

```
26 Starting kernel ...
27
28 [ 0.000000] Booting Linux on physical CPU 0x0
29 [ 0.000000] Linux version 5.5.5-g7070cea-dirty (wnavy@wnavy-vm) (gcc
version 9.2.0 (crosstool-NG 1.24.0.105_5659366)) #1 Tue Jul 7 00:38:11 CST
2020
30 [ 0.000000] CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ),
cr=0005317f
31 [ 0.000000] CPU: VIVT data cache, VIVT instruction cache
32 [ 0.000000] OF: fdt: Machine model: Freescale i.MX28 Evaluation Kit
33 [ 0.000000] Memory policy: Data cache writeback
34 [ 0.000000] Built 1 zonelists, mobility grouping on. Total pages:
16256
35 [ 0.000000] kernel command line: console=ttyAMA0,115200
root=/dev/mmcblk0p3 rw rootwait
36 [ 0.000000] Dentry cache hash table entries: 8192 (order: 3, 32768
bytes, linear)
37 [ 0.000000] Inode-cache hash table entries: 4096 (order: 2, 16384
bytes, linear)
38 [ 0.000000] mem auto-init: stack:off, heap alloc:off, heap free:off
39 [ 0.000000] Memory: 47544K/65536K available (7092K kernel code, 635K
rwddata, 2648K rodata, 324K init, 6368K bss, 17992K reserved, 0K cma-
reserved)
40 [ 0.000000] SLUB: Hwalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
41 [ 0.000000] Running RCU self tests
42 [ 0.000000] NR_IRQS: 16, nr_irqs: 16, preallocated irq: 16
43 [ 0.000000] random: get_random_bytes called from
start_kernel+0x2b4/0x4e0 with crng_init=0
44 [ 0.000000] clocksource: mxs_timer: mask: 0xffffffff max_cycles:
0xffffffff, max_idle_ns: 79635851949 ns
45 [ 0.000038] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every
89478484971ns
46 [ 0.002257] Console: colour dummy device 80x30
47 [ 0.002429] Lock dependency validator: Copyright (c) 2006 Red Hat,
Inc., Ingo Molnar
48 [ 0.002491] ... MAX_LOCKDEP_SUBCLASSES: 8
49 [ 0.002544] ... MAX_LOCK_DEPTH: 48
50 [ 0.002592] ... MAX_LOCKDEP_KEYS: 8192
51 [ 0.002638] ... CLASSHASH_SIZE: 4096
52 [ 0.002685] ... MAX_LOCKDEP_ENTRIES: 32768
53 [ 0.002733] ... MAX_LOCKDEP_CHAINS: 65536
54 [ 0.002780] ... CHAINHASH_SIZE: 32768
55 [ 0.002828] memory used by lock dependency info: 3997 kB
56 [ 0.002877] memory used for stack traces: 2112 kB
57 [ 0.002925] per task-struct memory footprint: 1536 bytes
58 [ 0.003267] Calibrating delay loop... 226.09 BogoMIPS (lpj=1130496)
59 [ 0.071546] pid_max: default: 32768 minimum: 301
60 [ 0.072633] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes,
linear)
61 [ 0.072758] Mountpoint-cache hash table entries: 1024 (order: 0, 4096
bytes, linear)
62 [ 0.083465] CPU: Testing write buffer coherency: ok
63 [ 0.094288] Setting up static identity map for 0x40008400 - 0x40008458
64 [ 0.103232] devtmpfs: initialized
65 [ 0.178418] clocksource: jiffies: mask: 0xffffffff max_cycles:
0xffffffff, max_idle_ns: 19112604462750000 ns
66 [ 0.178695] futex hash table entries: 256 (order: 1, 11264 bytes,
linear)
```

```
67 [ 0.180643] pinctrl core: initialized pinctrl subsystem
68 [ 0.196613] NET: Registered protocol family 16
69 [ 0.199970] DMA: preallocated 256 KiB pool for atomic coherent
allocations
70 [ 0.428485] Serial: AMBA PL011 UART driver
71 [ 0.432586] 80074000.serial: ttyAMA0 at MMIO 0x80074000 (irq = 222,
base_baud = 0) is a PL011 rev2
72 [ 0.713981] printk: console [ttyAMA0] enabled
73 [ 0.889110] mxs-dma 80004000.dma-apbh: initialized
74 [ 0.908854] mxs-dma 80024000.dma-apbx: initialized
75 [ 0.946217] SCSI subsystem initialized
76 [ 0.953954] usbcore: registered new interface driver usbfs
77 [ 0.960612] usbcore: registered new interface driver hub
78 [ 0.966849] usbcore: registered new device driver usb
79 [ 0.984452] pps_core: LinuxPPS API ver. 1 registered
80 [ 0.989523] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo
Giometti <giometti@linux.it>
81 [ 0.999158] PTP clock support registered
82 [ 1.006525] Advanced Linux Sound Architecture Driver Initialized.
83 [ 1.024400] clocksource: Switched to clocksource mxs_timer
84 [ 2.655197] random: fast init done
85 [ 3.630761] NET: Registered protocol family 2
86 [ 3.641319] tcp_listen_portaddr_hash hash table entries: 128 (order: 0,
5120 bytes, linear)
87 [ 3.650374] TCP established hash table entries: 1024 (order: 0, 4096
bytes, linear)
88 [ 3.658496] TCP bind hash table entries: 1024 (order: 3, 36864 bytes,
linear)
89 [ 3.666357] TCP: Hash tables configured (established 1024 bind 1024)
90 [ 3.674116] UDP hash table entries: 256 (order: 2, 20480 bytes, linear)
91 [ 3.681431] UDP-Lite hash table entries: 256 (order: 2, 20480 bytes,
linear)
92 [ 3.690246] NET: Registered protocol family 1
93 [ 3.702827] RPC: Registered named UNIX socket transport module.
94 [ 3.709385] RPC: Registered udp transport module.
95 [ 3.714183] RPC: Registered tcp transport module.
96 [ 3.719191] RPC: Registered tcp NFSv4.1 backchannel transport module.
97 [ 3.742843] workingset: timestamp_bits=30 max_order=14 bucket_order=0
98 [ 3.852491] NFS: Registering the id_resolver key type
99 [ 3.860031] key type id_resolver registered
100 [ 3.864847] key type id_legacy registered
101 [ 3.869574] jffs2: version 2.2. (NAND) © 2001-2006 Red Hat, Inc.
102 [ 3.881311] Block layer SCSI generic (bsg) driver version 0.4 loaded
(major 248)
103 [ 3.889453] io scheduler mq-deadline registered
104 [ 3.894102] io scheduler kyber registered
105 [ 3.906141] pwm-backlight backlight: backlight supply power not found,
using dummy regulator
106 [ 3.927051] 8006a000.serial: ttyAPP0 at MMIO 0x8006a000 (irq = 220,
base_baud = 1500000) is a 8006a000.serial
107 [ 3.939733] mxs-auart 8006a000.serial: Found APPUART 3.1.0
108 [ 3.947635] 8006c000.serial: ttyAPP1 at MMIO 0x8006c000 (irq = 221,
base_baud = 1500000) is a 8006c000.serial
109 [ 3.960154] mxs-auart 8006c000.serial: Found APPUART 3.1.0
110 [ 8.032732] nand: device found, Manufacturer ID: 0xc2, Chip ID: 0xf1
111 [ 8.039446] nand: Macronix NAND 128MiB 3,3V 8-bit
112 [ 8.044253] nand: 128 MiB, SLC, erase size: 128 KiB, page size: 2048,
OOB size: 64
```



```
113 [ 8.052280] Scanning device for bad blocks
114 [ 8.612323] Bad eraseblock 850 at 0x000006a40000
115 [ 8.759365] gpmi-nand 8000c000.gpmi-nand: driver registered.
116 [ 8.797974] spi-nor spi1.0: unrecognized JEDEC id bytes: ff ff ff ff ff
117 [ 8.811453] libphy: Fixed MDIO Bus: probed
118 [ 8.967174] libphy: fec_enet_mii_bus: probed
119 [ 8.978260] usbcore: registered new interface driver asix
120 [ 8.984151] usbcore: registered new interface driver ax88179_178a
121 [ 8.991075] usbcore: registered new interface driver cdc_ether
122 [ 8.997805] usbcore: registered new interface driver smsc95xx
123 [ 9.004026] usbcore: registered new interface driver net1080
124 [ 9.010418] usbcore: registered new interface driver cdc_subset
125 [ 9.017062] usbcore: registered new interface driver zaurus
126 [ 9.023305] usbcore: registered new interface driver cdc_ncm
127 [ 9.029310] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
128 [ 9.036939] usbcore: registered new interface driver usb-storage
129 [ 9.068170] ci_hdrc ci_hdrc.0: EHCI Host Controller
130 [ 9.074195] ci_hdrc ci_hdrc.0: new USB bus registered, assigned bus
number 1
131 [ 9.104682] ci_hdrc ci_hdrc.0: USB 2.0 started, EHCI 1.00
132 [ 9.128549] hub 1-0:1.0: USB hub found
133 [ 9.133663] hub 1-0:1.0: 1 port detected
134 [ 9.158503] ci_hdrc ci_hdrc.1: EHCI Host Controller
135 [ 9.163730] ci_hdrc ci_hdrc.1: new USB bus registered, assigned bus
number 2
136 [ 9.194731] ci_hdrc ci_hdrc.1: USB 2.0 started, EHCI 1.00
137 [ 9.208329] hub 2-0:1.0: USB hub found
138 [ 9.212604] hub 2-0:1.0: 1 port detected
139 [ 9.232317] stmp3xxx-rtc 80056000.rtc: registered as rtc0
140 [ 9.240114] i2c /dev entries driver
141 [ 9.263847] stmp3xxx-rtc_wdt stmp3xxx-rtc_wdt: initialized watchdog
with heartbeat 19s
142 [ 9.280220] mxs-mmc 80010000.spi: Got WP GPIO
143 [ 9.316784] mxs-mmc 80010000.spi: initialized
144 [ 9.375456] mmc0: new high speed SDHC card at address 0007
145 [ 9.386819] usbcore: registered new interface driver usbhcd
146 [ 9.392495] usbhcd: USB HID core driver
147 [ 9.403055] mmcblk0: mmc0:0007 SD16G 14.5 GiB
148 [ 9.452132] mmcblk0: p1 p2 p3
149 [ 9.463335] imx28-pinctrl 80018000.pinctrl: pin SAIF0_MCLK already
requested by leds; cannot claim for 80042000.saif
150 [ 9.474274] imx28-pinctrl 80018000.pinctrl: pin-116 (80042000.saif)
status -22
151 [ 9.481850] imx28-pinctrl 80018000.pinctrl: could not request pin 116
(SAIF0_MCLK) from group saif0.0 on device 80018000.pinctrl
152 [ 9.493821] mxs-saif 80042000.saif: Error applying setting, reverse
things back
153 [ 9.523013] mxs-saif: probe of 80042000.saif failed with error -22
154 [ 9.534272] mxs-sgtl5000 sound: failed to get mclk
155 [ 9.539583] mxs-sgtl5000: probe of sound failed with error -22
156 [ 9.547234] NET: Registered protocol family 17
157 [ 9.552400] Key type dns_resolver registered
158 [ 9.560733] registered taskstats version 1
159 [ 9.712756] stmp3xxx-rtc 80056000.rtc: setting system clock to 1970-01-
01T13:08:19 UTC (47299)
160 [ 9.723350] ALSA device list:
161 [ 9.727181] No soundcards found.
```

```
162 [ 9.737929] uart-pl011 80074000.serial: no DMA platform data
163 [ 9.998528] EXT4-fs (mmcblk0p3): recovery complete
164 [ 10.021308] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data
mode. Opts: (null)
165 [ 10.030344] VFS: Mounted root (ext4 filesystem) on device 179:3.
166 [ 10.042110] devtmpfs: mounted
167 [ 10.047675] Freeing unused kernel memory: 324K
168 [ 10.052218] This architecture does not have kernel memory protection.
169 [ 10.058981] Run /sbin/init as init process
170 [ 10.418092] EXT4-fs (mmcblk0p3): re-mounted. Opts: (null)
171 mount: mounting /dev/mmcblk0p1 on /mnt/sdcard failed: Invalid argument
172 Starting syslogd: OK
173 Starting klogd: OK
174 Running sysctl: OK
175 Starting mdev... OK
176 modprobe: can't change directory to '/lib/modules': No such file or
directory
177 Initializing random number generator: OK
178 Saving random seed: [ 15.269711] random: dd: uninitialized urandom read
(512 bytes read)
179 OK
180 Starting network: [ 16.079916] Generic PHY 800f0000.ethernet-1:05:
attached PHY driver [Generic PHY] (mii_bus:phy_addr=800f0000.ethernet-
1:05, irq=POLL)
181 OK
182 Starting ntpd: OK
183
184 Welcome to EasyArm-i.MX280a
185 EasyArm-i.MX280a login: [ 19.228436] fec 800f0000.ethernet eth0: Link is
Up - 100Mbps/Full - flow control off
186 [ 40.266980] lcd-3v3: disabling
187 [ 40.270673] can-3v3: disabling
188 [ 40.273856] lcd-5v: disabling
189 [ 164.826088] random: crng init done
190
191 Welcome to EasyArm-i.MX280a
192 EasyArm-i.MX280a login: root
193 Password:
194 # cat /proc/version
195 Linux version 5.5.5-g7070cea-dirty (wnavy@wnavy-vm) (gcc version 9.2.0
(crosstool-NG 1.24.0.105_5659366)) #1 Tue Jul 7 00:38:11 CST 2020
196 # cat /proc/cpuinfo
197 processor : 0
198 model name : ARM926EJ-S rev 5 (v5l)
199 Bogomips : 226.09
200 Features : swp half thumb fastmult edsp java
201 CPU implementer : 0x41
202 CPU architecture: 5TEJ
203 CPU variant : 0x0
204 CPU part : 0x926
205 CPU revision : 5
206
207 Hardware : Freescale MXS (Device Tree)
208 Revision : 0000
209 Serial : 0000000000000000
210 # top
211 Mem: 21708K used, 26160K free, 40K shrd, 304K buff, 4712K cached
212 CPU: 0% usr 28% sys 0% nic 71% idle 0% io 0% irq 0% irq
```

```

213 Load average: 0.02 0.06 0.02 1/39 145
214      PID  PPID  USER      STAT   VSZ %VSZ %CPU  COMMAND
215      145    141  root       R      2640   5%  29%  top
216      139     1  root       S      5384  11%   0%  /usr/sbin/ntpd -g
217      104     1  root       S      3028   6%   0%  /sbin/mdev -df
218      141     1  root       S      2768   6%   0%  -sh
219       1     0  root       S      2640   5%   0%  init
220      88     1  root       S      2640   5%   0%  /sbin/syslogd -n
221      92     1  root       S      2640   5%   0%  /sbin/klogd -n
222      22     2  root       IW        0   0%   0%  [kworker/0:1-eve]
223      70     2  root       IW<        0   0%   0%  [kworker/0:1H-kb]
224       7     2  root       SW        0   0%   0%  [ksoftirqd/0]
225       5     2  root       IW        0   0%   0%  [kworker/u2:0-ev]
226       8     2  root       SW        0   0%   0%  [kdevtmpfs]
227      71     2  root       SW        0   0%   0%  [jbd2/mmcblk0p3-]
228       2     0  root       SW        0   0%   0%  [kthreadd]
229       3     2  root       IW        0   0%   0%  [kworker/0:0-pm]
230       4     2  root       IW<        0   0%   0%  [kworker/0:0H-kb]
231       6     2  root       IW<        0   0%   0%  [mm_percpu_wq]
232       9     2  root       SW        0   0%   0%  [khungtaskd]
233      10     2  root       SW        0   0%   0%  [oom_reaper]
234      11     2  root       IW<        0   0%   0%  [writeback]
235      #

```

参考文献

<u-boot/doc/imx/common/mxs.txt>