(Internal)3921 uboot与kernel之间参数的传递

Kevin Tian
田伍红（7211）
to: Barry Wu, Simon Ho, Youri Zhang, Will Jiang
吴汉; 何聪; 张亚辉; 姜维维

2014/06/05 18:48

3921有3个分区保存启动参数

```
[PARTITION2]
NAME = bootargs
SIZE = 0x800000
FILE = bootargs.abs

[PARTITION3]
NAME = deviceinfo
SIZE = 0x800000
FILE = deviceinfo.abs

[PARTITION4]
NAME = baseparams
SIZE = 0x800000
FILE = baseparams.abs

[PARTITION5]
NAME = bootlogo
SIZE = 0x800000
FILE = bootlogo.ubo
```

这3个分区文件详细解析参考这几个头文件struct定义：
\PDK\alicommon\alidefinition\adf_boot.h
\PDK\u-boot\board\ALi\ali_3921\bootargs.h
\PDK\u-boot\board\ALi\ali_3921\deviceinfo.h
\PDK\u-boot\board\ALi\ali_3921\baseparams.h

示例:

ali nand分区信息进化路线图: pmi -> partition_tbl分区 -> bootargs.abs

bootargs.abs:分区信息的表示: mtdparts=

```
00003000h: E4 00 00 00 20 69 6E 69 74 3D 2F 69 6E 69 74 20 ; ?.. init=/init
00003010h: 61 6E 64 72 6F 69 64 62 6F 6F 74 2E 63 6F 6E 73 ; androidboot.cons
00003020h: 6F 6C 65 3D 74 74 79 53 30 20 6D 74 64 70 61 72 ; ole=ttyS0 mtdpar
00003030h: 74 73 3D 61 6C 69 5F 6E 61 6E 64 3A 38 4D 40 30 ; ts=ali_nand:8M@0
00003040h: 4D 28 62 6F 6F 74 29 2C 38 4D 40 38 4D 28 62 6F ; M(boot),8M@8M(bo
00003050h: 6F 74 62 61 6B 29 2C 38 4D 40 31 36 4D 28 62 6F ; otbak),8M@16M(bo
00003060h: 6F 74 61 72 67 73 29 2C 38 4D 40 32 34 4D 28 64 ; otargs),8M@24M(d
00003070h: 65 76 69 63 65 69 6E 66 6F 29 2C 38 4D 40 33 32 ; eviceinfo),8M@32
00003080h: 4D 28 62 61 73 65 70 61 72 61 6D 73 29 2C 38 4D ; M(baseparams),8M
00003090h: 40 34 30 4D 28 62 6F 6F 74 6C 6F 67 6F 29 2C 38 ; @40M(bootlogo),8
000030a0h: 4D 40 34 38 4D 28 62 6F 6F 74 6D 65 64 69 61 29 ; M@48M(bootmedia)
000030b0h: 2C 38 4D 40 35 36 4D 28 73 65 65 29 2C 31 36 4D ; ,8M@56M(see),16M
000030c0h: 40 36 34 4D 28 6B 65 72 6E 65 6C 29 2C 38 4D 40 ; @64M(kernel),8M@
```

```
000030d0h: 38 30 4D 28 61 65 29 2C 35 30 30 4D 40 38 38 4D ; 80M(ae),500M@88M
000030e0h: 28 72 6F 6F 74 66 73 29 00 00 00 00 00 00 00 00  ; (rootfs)........



struct boot_args * get_bootargs()
{


        args_off =  nand->writesize*2048;   //-->16M(8K*2048)  /  8M(4K*2048)
get_bootargs从这里开始，找4个block

        for (i=0;i<4;i++)
        {
                offset = args_off+i*nand->erasesize;
                if (nand_block_isbad(nand, offset&~(nand->erasesize - 1)))
                        continue;

                if (nand_read(nand, offset, &len, datbuf))
                /* check block data */
                if (_sector_crc_check(datbuf, nand->erasesize)>0) //校验正确，退出
                {
                        break;
                }
        }


}


int set_mtdparts(void)
{
        struct boot_args *bootargs = get_bootargs();
        char *cmdline = bootargs->cmdline_rcv;
        char *mtdparts = strstr(cmdline,"mtdparts="); //找到mtdparts=起始位置

        setenv("mtdparts", mtdparts);       //分区信息通过标准cmd line传入kernel

//mtdparts=ali_nand:8M@0M(boot),8M@8M(bootbak),8M@16M(bootargs),8M@24M(deviceinfo),8M@32M
(baseparams),8M@40M(bootlogo),8M@48M(bootmedia),8M@56M(see),16M@64M(kernel),8M@80M(ae),50
0M@88M(rootfs)


        setenv("mtdids", mtdids);    // mtdids = ali_nand

        return 0;
}


bootargs.abs完整内容解析

struct boot_args
{
        char magic[16];       // bootargs
        int registerinfo_size;
        unsigned char registerinfo [1024*8-12-16-4];
        int cmdline_size;
        char cmdline[1024*4-4];      // nand分区信息mtdpart=放在这里
        int cmdline_size_rcv;
        char cmdline_rcv[1024*4-4];
        int meminfo_size;
```

```
        unsigned char meminfo[1024*4-4];
        int meminfo_size_rcv;
        unsigned char meminfo_rcv[1024*4-4];
        int tveinfo_size;
        unsigned char tveinfo[1024*28-4];
};
```

```
[PARTITION3]
NAME = deviceinfo
SIZE = 0x800000
FILE = deviceinfo.abs
```

deviceinfo.abs

```
struct device_info
{
        /* 0 bytes */
        u_char magic[16];     // deviceinfo
        /* 16 bytes */
        HDMI_INFO hdmi;
        /* 304 bytes */
        FIRMWARE_INFO firmware;
};


typedef struct
{
        u_char hdcp[286];
        u_short hdcp_disable;
}HDMI_INFO;    /* 288 bytes */

#define STB_FIRMINFO_SN_LEN                64
#define STB_FIRMINFO_MAC_LEN                        8
typedef struct
{
        u_char sn[STB_FIRMINFO_SN_LEN];
        u_char mac[STB_FIRMINFO_MAC_LEN];
        u_char mac2[STB_FIRMINFO_MAC_LEN];
        u_char mac3[STB_FIRMINFO_MAC_LEN];
        u_char mac4[STB_FIRMINFO_MAC_LEN];
        u_int oui;
        u_char hw_ver[128];
        u_char rsv[1820];
}FIRMWARE_INFO;        /* 2048 bytes */

u_char magic[16];     // deviceinfo
00000000h: 3C 09 00 00 C3 F6 FF FF 39 6F FE 96 64 65 76 69 ; <...闽    9o敷devi


HDMI_INFO hdmi;
00000010h: 63 65 69 6E 66 6F 00 00 00 00 00 00 00 00 00 00  ; ceinfo..........


u_char sn[STB_FIRMINFO_SN_LEN];
00000130h: 00 00 00 00 00 00 00 00 00 00 01 00 73 6 E 00 00 ; ............sn..
00000140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ; ................
00000150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ; ................
00000160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ; ................
```

```
        u_char mac[STB_FIRMINFO_MAC_LEN];
        u_char mac2[STB_FIRMINFO_MAC_LEN];
        u_char mac3[STB_FIRMINFO_MAC_LEN];
        u_char mac4[STB_FIRMINFO_MAC_LEN];
        u_int oui;
        u_char hw_ver[128];
        u_char rsv[1820];
00000170h: 00 00 00 00 00 00 00 00 00 00 00 00  DE AD BE EF ; ............薪撅
00000180h: 01 01 61 61 12 22 33 44 55 66 65 33 14 22 33 44  ; ..aa."3DUfe3."3D
00000190h: 55 66 63 33 16 22 33 44 55 66 61 33 08 10 00 00  ; Ufc3."3DUfa3....
000001a0h: 68 77 30 2E 31 2E 30 00 00 00 00 00 00 00 00 00  ; hw0.1.0.........
000001b0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ; ................
000001c0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ; ................
000001d0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ; ................


int set_mac(void)
{
        struct device_info *deviceinfo = get_deviceinfo();
        u_char *mac = deviceinfo->firmware.mac;

        setenv("ethaddr", macStr);   // mac地址通过cmd line传入kernel
}



[PARTITION4]
NAME = baseparams
SIZE = 0x800000
FILE = baseparams.abs
```

baseparams.abs

```
struct base_params {
        char magic[16];
        ADF_BOOT_AVINFO avinfo;
        struct SysInfo sysinfo;
        unsigned char rsv[1024*6];
};



typedef struct
{
        unsigned char tvSystem;
        unsigned char progressive;
        unsigned char tv_ratio;
        unsigned char display_mode;

        unsigned char scart_out;
        unsigned char vdac_out[6];
        unsigned char video_format;

        unsigned char audio_output;
        unsigned char brightness;
        unsigned char contrast;
        unsigned char saturation;
```

```
        unsigned char sharpness;
        unsigned char hue;
        unsigned char resv[10];
}ADF_BOOT_AVINFO;

struct SysInfo
{
    char sw_ver[128];
    unsigned char sw_md5[128];
    unsigned char sw_private[1024];
};
```

uboot怎么向kernel传递以上信息:

除了mtdparts= mac地址等通过标准kernel cmd line传入解析外:

```
#define PRE_DEFINED_ADF_BOOT_START                  (0x84000000 - 0x20000)

int set_boardinfo(int isRecovery)
{
        unsigned int size;
        unsigned char *p;

        ADF_BOOT_BOARD_INFO *boardinfo = (ADF_BOOT_BOARD_INFO *)(
PRE_DEFINED_ADF_BOOT_START); // uboot把board_info写到这个地址

        size = get_hdmiinfo(&p);
        memcpy((u_char*)&boardinfo->hdcp, p, size);

        size = get_avinfo(&p);
        memcpy((u_char*)&boardinfo->avinfo, p, size);

        size = get_mac(&p);
        memcpy((u_char*)&boardinfo->macinfo, p, size);

        size = get_memmap(&p,isRecovery);
        memcpy((u_char*)&boardinfo->memmap_info, p, size);

        //dump_print("memmap:",p, 32);

        size = get_tveinfo(&p);
        memcpy((u_char*)&boardinfo->tve_info, p, size);

        size = get_registerinfo(&p);
        memcpy((u_char*)&boardinfo->reg_info, p, size);

        return 0;
}



typedef struct
{
        ADF_BOOT_HDCP hdcp;
        ADF_BOOT_AVINFO avinfo;
        ADF_BOOT_MEMMAPINFO memmap_info;
        ADF_BOOT_TVEINFO tve_info;
        ADF_REGISTERINFO reg_info;
```

```
        ADF_BOOT_MAC_PHYADDR macinfo;
        ADF_BOOT_MEDIAINFO    media_info;
        ADF_BOOT_GMA_INFO gma_info;
ADF_SEE_HEART_BEAT heart_beat;

 }ADF_BOOT_BOARD_INFO; // the maximun size is  128K
```

kernel在各个地方读取PRE_DEFINED_ADF_BOOT_START，获取board_info

```
Board_config.c
(y:\pdk_release\pdk1.5.5\linux\kernel\boards\c3921_alidroid\boardfiles):433:
ADF_BOOT_BOARD_INFO *board_info = (ADF_BOOT_BOARD_INFO *)((unsigned
long)(phys_to_virt)PRE_DEFINED_ADF_BOOT_START);


Hdmi_proc.c (y:\pdk_release\pdk1.5.5\linux\kernel\alidrivers\modules\alihdmi):1847:
ADF_BOOT_BOARD_INFO *board_info = (ADF_BOOT_BOARD_INFO
*)(ALI_VIRT(PRE_DEFINED_ADF_BOOT_START));


Ali_rpc_dev_init.c
(y:\pdk_release\pdk1.5.5\linux\kernel\alidrivers\modules\alirpcng\rpchld):186:
ADF_BOOT_BOARD_INFO *board_info = (ADF_BOOT_BOARD_INFO *)((unsigned
long)(phys_to_virt)PRE_DEFINED_ADF_BOOT_START);
Ali_rpc_dev_init.c
(y:\pdk_release\pdk1.5.5\linux\kernel\alidrivers\modules\alirpcng\rpchld):191:
tve_adjust_table_total_reconstruct[i].index =
board_info->tve_info.tve_adjust_table_info[i].index;
Ali_rpc_dev_init.c
(y:\pdk_release\pdk1.5.5\linux\kernel\alidrivers\modules\alirpcng\rpchld):192:
tve_adjust_table_total_reconstruct[i].pTable =
(T_TVE_ADJUST_ELEMENT*)(board_info->tve_info.tve_adjust_table_info[i].field_info);
```

bootargs.abs/deviceinfo.abs/baseparams.abs 这3个文件又从哪里来？

是编译时host tool解析\PDK\linux\kernel\boards\C3921\flashfiles\下面的xml文件自动生成的

\PDK\linux\kernel\boards\C3921\prebuild.sh

```
#!/bin/sh
# use shell funcs
. $BUILD_DIR/funcs.sh

$BOARD_DIR/flashfiles/genrawpart.sh
```

\PDK\linux\kernel\boards\C3921\flashfiles\genrawpart.sh

```
HOSTTOOLS_DIR=$LINUX_SRC_DIR/../tools/hosttools

$BOARD_DIR/flashfiles/genftoolini.sh
$BOARD_DIR/flashfiles/genbootargs.sh
$BOARD_DIR/flashfiles/gendeviceinfo.sh
$BOARD_DIR/flashfiles/genbaseparams.sh
$BOARD_DIR/flashfiles/genbootmedia.sh
```

这些sh会调用\PDK\tools\hosttools\python\下面的脚本解析flashfiles目录里xml生成.abs

Best Regards,

Kevin Tian

****
ALi(Shanghai) Corp.
6F-A, Building 3, No. 7 Guiqing Road, Shanghai 200233
Tel: +86 21-64855058 ext. 7211
FAX: +86 21-64951498
Mail: kevin.tian@alitech.com
****