Linux U-Boot
开发指南

版本号: 3.5
发布日期: 2025.5.6

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.7.19	AWA1538	添加基础模板
1.1	2020.7.21	AWA1538	添加快速编译 boot0 及 U-Boot
2.0	2020.11.10	AWA1538	添加 U-Boot 配置参数文件介绍，重点介绍内部 fdt 使用
3.0	2021.05.24	AWA1538	增加 LICHEE 配置宏信息
3.1	2023.11.27	XAA0249	仅修改标点
3.2	2024.9.1	AWA2256	仅修改部分表述
3.3	2025.1.14	XAA0329	增加组合按键烧录功能描述
3.4	2025.3.18	XAA0312	更新文档描述
3.5	2025.5.6	XAA0329	1. 合并 dts defconfig 两个表格 2. 改正 dts 表格标点符号问题 3. 改正.buildconfig 文件内容的描述，更新 LICHEE 类环境变量描述为表格 4. 添加编译方法介绍部分的声明 5. 删除 defconfig 的手动配置方式 6. 增加 LICHEE_BOARD_CONFIG_DIR 的具体实例 7. 改正 fastboot 命令说明问题部分的字体问题 8. 改正 fastboot 命令使用步骤的双引号问题 9. 改正 fastboot 基本使用命令部分的英文冒号问题 10. 改正 FDT 部分的小节换行问题 11. 改正烧 key 图中的非法 mac 12. 改正进入烧写方法部分的标点符号问题 13. 改正 FAQ 的风格不一致问题

目　录

插　图

1 前言

1.1 编写目的

介绍 U-Boot 的编译打包、基本配置、常用命令的使用、基本调试方法等，为 U-BOOT 的移植及应用开发提供了基础。

1.2 适用范围

本文档适用于 brandy-2.0，即 U-Boot-2018 平台。

1.3 相关人员

U-Boot 开发/维护人员，内核开发人员。

2 相关术语介绍

表 2-1: 术语介绍

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台
U-Boot	Universal Boot Loader
FDT	flatted device tree，扁平设备树

3 LICHEE 类宏关键字解释

longan 目录下的.buildconfig 文件里定义了 LICHEE 环境变量，LICHEE 环境变量具体的解释如下:

环境变量	解释
LICHEE_IC	IC 名
LICHEE_CHIP	平台名
LICHEE_BOARD	板级名
LICHEE_ARCH	所属架构
LICHEE_BOARD_CONFIG_DIR	板级目录
LICHEE_BRANDY_OUT_DIR	bin 文件所在目录
LICHEE_PLAT_OUT	平台临时 bin 所在目录
LICHEE_CHIP_CONFIG_DIR	IC 目录

4 编译方法介绍

注意：以下示例以 android sdk 为例，介绍编译方法

4.1 准备编译工具链

准备编译工具链接执行步骤如下：

```
1）cd longan/brandy/brandy-2.0/
2）./build.sh -t
```

若有以下警告打印，则说明当前编译版本不支持通过命令解压工具链，会在编译前自动解压工具链。

```
This command is not supported, the toolchain will be unpacked during building
```

4.2 快速编译 boot0 及 U-Boot

在 longan/brandy/brandy-2.0/目录下，执行./build.sh -p 平台名称，可以快速完成整个 boot 编译动作。这个平台名称是指，LICHEE_CHIP。

```
./build.sh -p {LICHEE_CHIP}   //快速编译spl/U-Boot
./build.sh -o spl-pub -p {LICHEE_CHIP} //快速编译spl-pub
./build.sh -o uboot -p {LICHEE_CHIP}   //快速编译U-Boot
```

4.3 编译 U-Boot

cd longan/brandy/brandy-2.0/u-boot-2018/进入 u-boot-2018 目录。以 {LICHEE_CHIP} 为例，依次执行如下操作即可。

```
1）make {LICHEE_CHIP}_defconfig
2）make -j
```

4.4 编译 boot0/fes/sboot

cd longan/brandy/brandy-2.0/spl-pub 进入 spl-pub 目录，需设置平台和要编译的模块参数。以 {LICHEE_CHIP} 为例，编译 nand/emmc 的方法如下：

1. 编译 **boot0**

```
make distclean
make p={LICHEE_CHIP} m=nand
make boot0

make distclean
make p={LICHEE_CHIP} m=emmc
make boot0
```

2. 编译 **fes**

```
make distclean
make p={LICHEE_CHIP} m=fes
make fes
```

3. 编译 **sboot**

```
make distclean
make p={LICHEE_CHIP} m=sboot
make sboot
```

注：目前我司部分平台已不提供 spl-pub，而是直接提供镜像，此章节仅做参考。

5 U-Boot 功能及其配置方法/文件介绍

5.1 U-Boot 功能介绍

在嵌入式操作系统中，BootLoader/U-Boot 是在操作系统内核运行之前运行。可以初始化硬件设备、建立内存空间映射图，从而将系统的软硬件环境带到一个合适状态，以便为最终调用操作系统内核准备好正确的环境。在 sunxi 平台中，除了必须的引导系统启动功能外，BOOT 系统还提供烧写、升级等其它功能。

U-Boot 主要功能可以分为以下几类

1. 引导内核
   能从存储介质（nand/mmc/spinor）上加载内核镜像到 DRAM 指定位置并运行

2. 量产 & 升级
   包括卡量产，USB 量产，私有数据烧录，固件升级

3. 开机提示信息
   开机能显示启动 logo 图片（BMP 格式)

4. Fastboot 功能
   实现 fastboot 的标准命令，能使用 fastboot 刷机

5.2 U-Boot 功能配置方法介绍

U-Boot 中的各项功能可以通过配置菜单 menuconfig 进行开启或关闭，具体配置方法步骤如下:

1. cd brandy/brandy-2.0/u-boot-2018/
2. 执行 make menuconfig 命令，会弹出 menuconfig 配置菜单窗口，如下图所示。此时即可对各模块功能进行配置，配置方法 menuconfig 配置菜单窗口中有说明。
3. 修改后配置已经生效，直接 make 即可生成对应 bin。如果重新运行 make xxx_defconfig，通过 menuconfig 方式修改的配置会在运行 make xxx_defconfig 后被 xxx_defconfig 中的配置覆盖。

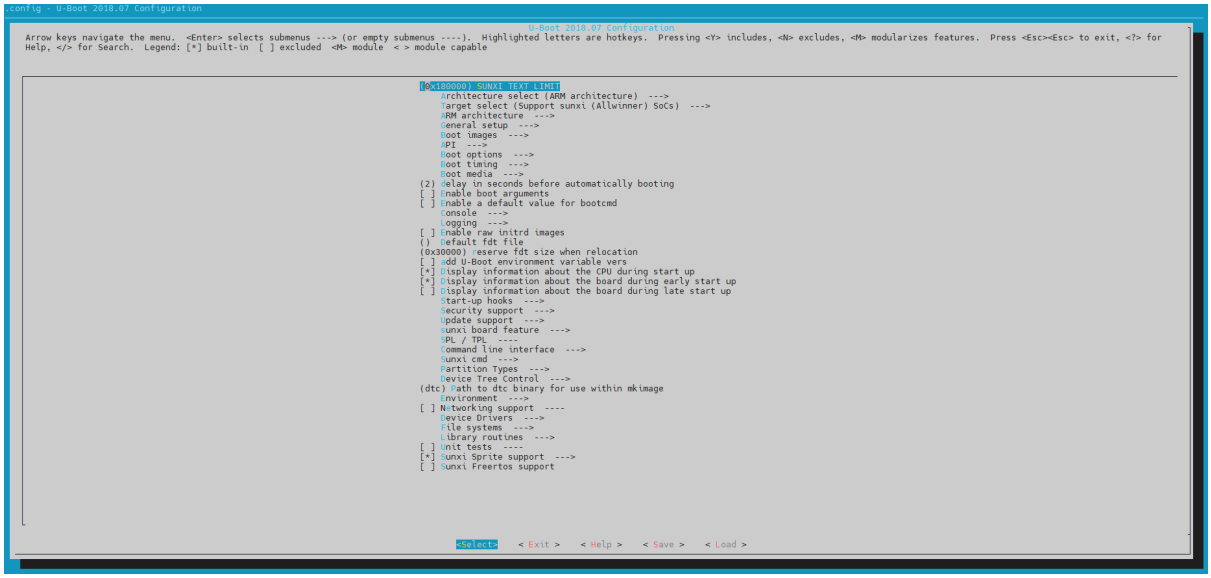

图 5-1: menuconfig

5.3 U-Boot 配置参数文件介绍

U-Boot 自 linux-5.4 以后不再使用 sysconfig 和内核 dts 作为配置文件，而是使用 U-Boot 自带的 dts 来配置参数。kernel-dts 与 U-Boot-dts 完全独立。

5.3.1 U-Boot-dts 路径

U-Boot-dts 路径为：longan/brandy/brandy-2.0/u-boot-2018/arch/arm/dts

5.3.2 U-Boot-dts 相关 defconfig 配置

配置项	配置项含义	选项
CONFIG_OF_SEPARATE	uboot dts 编入 uboot 镜像	y
CONFIG_OF_BOARD	关闭使用外部 dts	n
CONFIG_DEFAULT_DEVICE_TREE	选择构建的 dts 文件文件名	{LICHEE_CHIP}-soc-system
CONFIG_SUNXI_NECESSARY_REPLACE_FDT	内部 dts 换成外部 dts	y

5.3.3 U-Boot-dts 注意事项

5.3.3.1 编译注意事项

1.dts 分为板级 dts，和系统 dts

- 系统 dts 由 CONFIG_DEFAULT_DEVICE_TREE 决定，可以在 $(CONFIG_SYS_CONFIG_NAME)_defconfig 找到该宏的定义。
- 系统 dts 最终会 include 板级 dts，文件路径 {LICHEE_BOARD_CONFIG_DIR}，例如 sdk 下的 device/config/chips/a537/configs/evb1/uboot-board.dts，文件名:uboot-board.dts。

2. 我们可以通过编译时的打印判断启动的 dts

```
OBJCOPY examples/standalone/hello_world.srec
OBJCOPY examples/standalone/hello_world.bin
LD    u-boot
OBJCOPY u-boot.srec
OBJCOPY u-boot-nodtb.bin
'{LICHEE_BOARD_CONFIG_DIR}/uboot-board.dts' -> '~/longan/brandy/brandy-2.0/u-boot-2018/arch/{LICHEE_ARCH
    }/dts/.board-uboot.dts'
DTC    arch/{LICHEE_ARCH}/dts/{LICHEE_CHIP}-soc-system.dtb
SYM    u-boot.sym
SHIPPED dts/dt.dtb
FDTGREP dts/dt-spl.dtb
COPY    u-boot.dtb
CAT    u-boot-dtb.bin
COPY    u-boot.bin
'u-boot.bin' -> '{LICHEE_CHIP}.bin'
'u-boot-g{LICHEE_CHIP}.bin' -> '{LICHEE_BRANDY_OUT_DIR}/bin/u-boot-g{LICHEE_CHIP}.bin'
'u-boot-g{LICHEE_CHIP}.bin' -> '{LICHEE_PLAT_OUT}/u-boot-g{LICHEE_CHIP}.bin'
CFGCHK  u-boot.cfg
```

5.3.3.2 语法注意事项

当系统 dts 与板级 dts 存在同路径下同名节点时，板级 dts 将会覆盖系统 dts。

5.3.3.3 运行时注意事项

1. 为了在启动内核前更新参数到内核 dts 和可以在 U-Boot 控制台查看修改 dts。按阶段划分可以分为使用内部 dts 阶段和使用内核 dts 阶段，如下图所示。

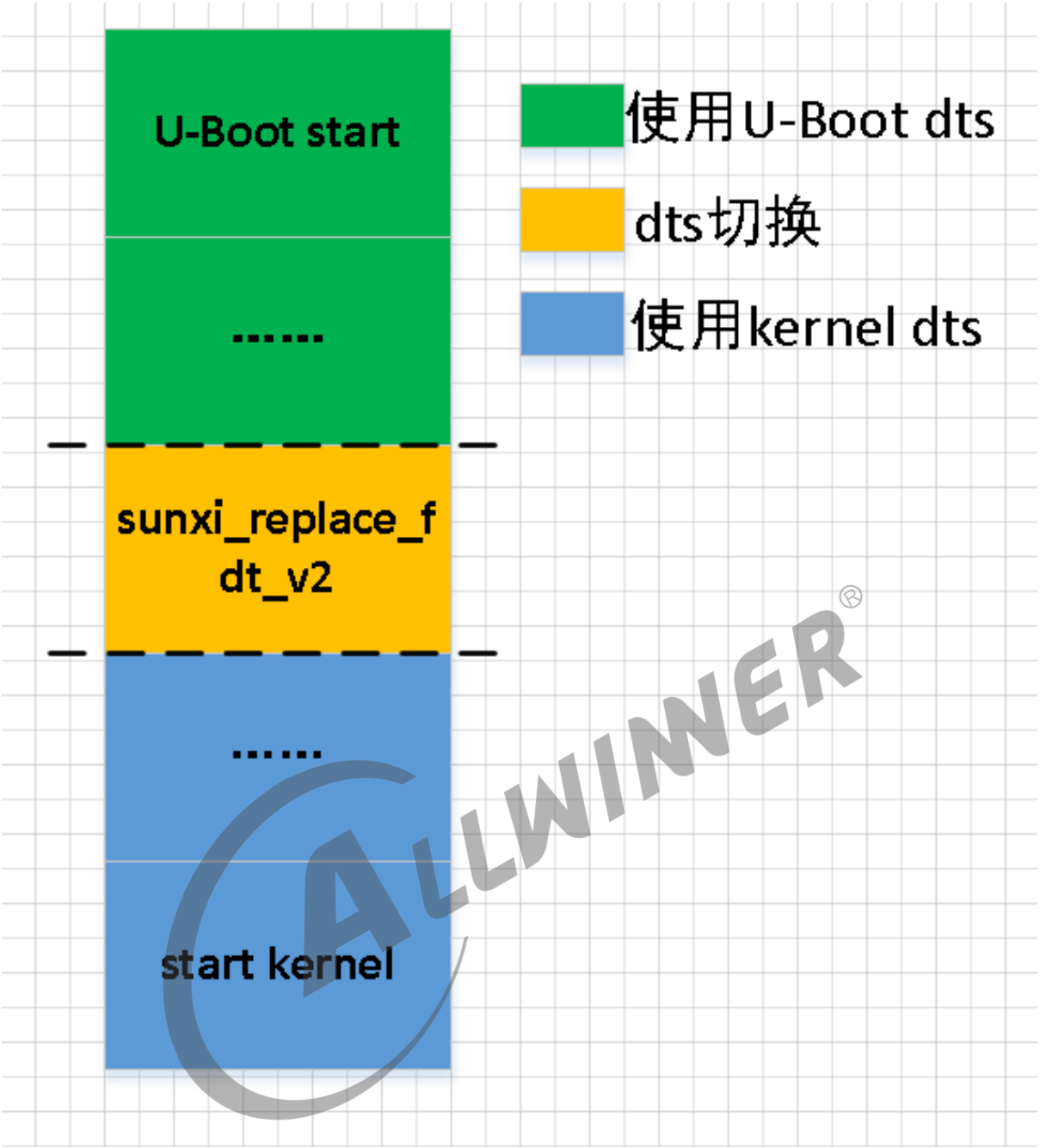

图 5-2: dts 变化图

2. 可以通过命令 set_working_fdt 来切换当前生效的 fdt。

```
[04.562]update bootcmd
[04.576]change working_fdt 0x7bebee58 to 0x7be8ee58
[04.587]update dts
Hit any key to stop autoboot:  0
=> set
  set_working_fdt setenv setexpr
=> set_working_fdt 0x7bebee58
change working_fdt 0x7be8ee58 to 0x7bebee58
=>
```

6 U-Boot 常用命令介绍

6.1 env 命令说明

通过 env 命令可以对 {LICHEE_CHIP_CONFIG_DIR}/configs/default/env.cfg 中的环境变量进行查看及更改。在小机启动过程中按任意键进入 U-Boot shell 命令状态，输入命令 “env” 即可查看命令帮助信息。具体示例如下：

1. 输入命令 “env print”，可查看当前所有的环境变量信息，如下：

```
=> pri
ab_partition_list=bootloader,env,boot,vendor_boot,dtbo,vbmeta,vbmeta_system,vbmeta_vendor
android_trust_chain=true
boot_fastboot=fastboot
boot_normal=sunxi_flash read 45000000 boot;bootm 45000000
boot_recovery=sunxi_flash read 45000000 recovery;bootm 45000000
bootcmd=run setargs_mmc boot_normal
bootdelay=0
bootreason=charger
bt_mac=20:A1:11:12:13:44
cma=8M
console=ttyAS0,115200
earlyprintk=sunxi-uart,0x05000000
fdtcontroladdr=7bed0e60
fileaddr=40000000
filesize=15cf6
force_normal_boot=1
init=/init
initcall_debug=0
keybox_list=widevine,ec_key,ec_cert1,ec_cert2,ec_cert3,rsa_key,rsa_cert1,rsa_cert2,rsa_cert3
loglevel=8
mac=10:14:15:15:9A:CA
mmc_root=/dev/mmcblk0p4
nand_root=/dev/nand0p4
partitions=bootloader_a@mmcblk0p1:bootloader_b@mmcblk0p2:env_a@mmcblk0p3:env_b@mmcblk0p4:
        boot_a@mmcblk0p5:boot_b@mmcblk0p6:vendor_boot_a@mmcblk0p7:vendor_boot_b@mmcblk0p8:
        super@mmcblk0p9:misc@mmcblk0p10:vbmeta_a@mmcblk0p11:vbmeta_b@mmcblk0p12:
        vbmeta_system_a@mmcblk0p13:vbmeta_system_b@mmcblk0p14:vbmeta_vendor_a@mmcblk0p15:
        vbmeta_vendor_b@mmcblk0p16:frp@mmcblk0p17:empty@mmcblk0p18:metadata@mmcblk0p19:
        private@mmcblk0p20:dtbo_a@mmcblk0p21:dtbo_b@mmcblk0p22:media_data@mmcblk0p23:
        UDISK@mmcblk0p24
rotpk_status=0
setargs_mmc=setenv  bootargs earlyprintk=${earlyprintk} clk_ignore_unused initcall_debug=${initcall_debug} console=
        ${console} loglevel=${loglevel} root=${mmc_root}  init=${init} cma=${cma} snum=${snum} mac_addr=${mac}
        wifi_mac=${wifi_mac} bt_mac=${bt_mac} specialstr=${specialstr} gpt=1 androidboot.force_normal_boot=${
        force_normal_boot} androidboot.slot_suffix=${slot_suffix}
setargs_nand=setenv bootargs earlyprintk=${earlyprintk} clk_ignore_unused initcall_debug=${initcall_debug} console=
        ${console} loglevel=${loglevel} root=${nand_root} init=${init} cma=${cma} snum=${snum} mac_addr=${mac}
```

```
        wifi_mac=${wifi_mac} bt_mac=${bt_mac} specialstr=${specialstr} gpt=1 androidboot.force_normal_boot=${
        force_normal_boot} androidboot.slot_suffix=${slot_suffix}
slot_suffix=_a
snum=A100B3N041
wifi_mac=10:A1:11:12:13:44

Environment size: 2078/131068 bytes
=>
```

2. 输入命令 “env set bootdelay 3”，可更改环境变量 bootdelay（即 boot 启动时 log 中的倒计时延迟时间）值的大小。
3. 输入命令 “env save”，即可将上述更改进行保存，保存后重新上电，或输入命令 “reset”，即可看到上述更改 bootdelay 的延时时间被更改生效。
4. 其他 env 命令请查看 env 帮助信息。

6.2 sunxi_flash read 命令说明

6.2.1 使用方法

用以下命令将 flash 指定地址中数据读到 DRAM 的指定地址处：

```
sunxi_flash read dram_addr flash_addr
```

6.2.2 使用示例

```
sunxi_flash read 0x45000000 env—将env分区数据读到DRAM的0x45000000地址处
sunxi_flash read 45000000 boot;bootm 45000000—将flash中boot分区数据读到DRAM的0x45000000地址，并从
        0x45000000处启动。
```

6.3 fastboot 命令说明

fastboot 是 Android 平台上一个通用的刷机工具，也是一个很好的开发调试工具，以下介绍 fastboot 的基本使用方法。

6.3.1 使用前提

fastboot PC 端工具可以从 Google Android SDK(Android-sdk-windows/tools) 中获得，也可以在 Android 源代码编译过后的生成文件获得 (out/host/linux-x86/bin)。

在 Linux 系统中，使用 fastboot 不需要安装驱动。但在 Windows 系统中，使用 fastboot 前需安装 fastboot 相关驱动。adb 的驱动在 fastboot 模式下也可以安装成功，但是无法使用，请使用我们提供的驱动，并手动安装。

6.3.2 使用步骤

1. 小机上电启动，按任意键进入 U-Boot 命令状态；
2. 串口端输入 **“fastboot”** 命令；
3. 打开 PC 端 fastboot 工具，并输入 **“fastboot devices”** 命令，看是否有 fastboot 设备显示；
4. 在正确获取 fastboot 设备的前提下，输入命令 **“fastboot flash env /path/to/env.fex”**，将 **env.fex** 写到 **env** 分区（**/path/to/**目录下的 **env.fex** 中 **“bootdelay”** 值应该与 flash 中原有 **env** 中 **“bootdelay”** 值不同，这样可根据 **“bootdelay”** 值不同来确定 fastboot 烧写是否成功），同下载 **env.fex** 分区一样，输入命令 “fastboot flash boot /path/to/boot.img” 将内核下载到内存中；
5. 输入 **“fastboot reboot”** 命令重启，查看启动倒计时即 **bootdelay** 的值是否改变；

6.3.3 fastboot 基本命令使用示例

1. fastboot 几个基本命令示例如下:

fastboot devices：显示 fastboot 的设备。**fastboot erase**：擦除分区，例如 **fastboot erase boot**，擦除 boot 分区。**fastboot flash**：旧分区（待写分区），例如 **fastboot flash boot/path/to/boot.img**，将 boot.img 写到 boot 分区。

2. 注意事项:

   fastboot 中使用的分区和 sys_partition.fex 中分区一致，具体的分区信息可以从小机上电启动进入 U-Boot shell 命令状态，输入命令 **“part list sunxi_flash 0”** 中获取，分区信息如下：

```
=> part list sunxi_flash 0

Partition Map for UNKNOWN device 0  --   Partition Type: EFI

Part    Start LBA     End LBA      Name
     Attributes
     Type GUID
     Partition GUID
 1    0x00008000     0x00017fff     "bootloader"
     attrs:  0x8000000000000000
     type:   ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
     guid:   a0085546-4166-744a-a353-fca9272b8e45
 2    0x00018000     0x0001ffff     "env"
     attrs:  0x8000000000000000
     type:   ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
     guid:   a0085546-4166-744a-a353-fca9272b8e46
```

```
3   0x00020000    0x0002ffff    "boot"
    attrs: 0x8000000000000000
    type:  ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
    guid:  a0085546-4166-744a-a353-fca9272b8e47
4   0x00030000    0x0032ffff    "super"
    attrs: 0x8000000000000000
    type:  ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
    guid:  a0085546-4166-744a-a353-fca9272b8e48
5   0x00330000    0x00337fff    "misc"
    attrs: 0x8000000000000000
    type:  ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
    guid:  a0085546-4166-744a-a353-fca9272b8e49
6   0x00338000    0x00347fff    "recovery"
    attrs: 0x8000000000000000
    type:  ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
    guid:  a0085546-4166-744a-a353-fca9272b8e4a
```

6.4 fat 命令说明

fat 命令可以对 FAT 文件系统的相关存储设备进行查询及文件读写操作，在打包固件的时候，我们会制作启动资源分区镜像，把指定的目录下的文件按照文件系统的格式排布，文件中包括了原来目录中的所有文件，并完全按照目录结构排列。当把这个镜像文件烧写到存储设备上的某一个分区的时候，可以看到这个分区和原有目录的内容一样。使用fat可以方便地以文件和目录的方式对小机 flash 进行数据访问，如显示 logo。这些指令基本上要和 U 盘或者 SD 卡同时使用，主要用于读取这些移动存储器上的 FAT 分区。其相关操作命令如下：

1. **fatls** : 列出相应设备目录上的所有文件，示例如下图：

```
sunxi#fatls mmc 2:2
            bat/
       344813   font24.sft
       357443   font32.sft
       307256   bootlogo.bmp
          512   magic.bin

4 file(s), 1 dir(s)

sunxi#
```

图 6-1: fatls 命令执行示例图

说明

补充说明，fatls mmc 2:2 中的第一个 2 表示的是 emmc 设备，2 表示其分区号，其说明如下图：

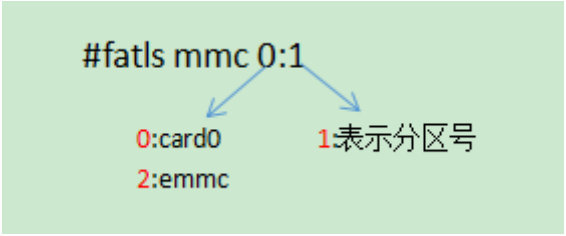

图 6-2: fatls 命令参数说明图

2. **fatinfo**: 打印出相应设备目录的文件系统信息，示例如下图：

```
sunxi#fatinfo mmc 2:2
Interface:  MMC
  Device 2: Vendor: MID 000011 PSN 7da44958 Rev: PRV 0.4 Prod: PNM 008G30
            Type: Removable Hard Disk
            Capacity: 7456.0 MB = 7.2 GB (15269888 x 512)
Filesystem: FAT16 "Volumn"
sunxi#
```

图 6-3: fatinfo 命令执行示例图

3. **fatload**: 从 FAT 文件系统中读取二进制文件到 RAM 存储中，示例如下：

```
sunxi#usb start
(Re)start USB...
USB0: start sunxi ehci1...
config usb pin success
config usb clk ok
sunxi ehci1 init ok...
USB EHCI 1.00
scanning bus 0 for devices... 3 USB Device(s) found
scanning usb for storage devices... 1 Storage Device(s) found
sunxi#fatls usb 0:1 /
16024600 sandisksecureaccessv3_win.exe
sandisk secureaccess/
lost.dir/
Android/
test/
video test/
amapauto/
0 vid_20161017_160818.ts
phoenixsuit/
system volume information/
0 vid_20161017_160919.ts
video/
156672 wifi pro_com su.exe
495 sys.ini
1035 pr_80211g_all.ini
config/
158208 wifi pro_new.exe
158208 wifi pro.exe
0 vid_20161017_164822.ts
0 vid_20161017_164906.ts
```

```
sunxi-tvd/
71149 sys_config.fex
vga/
397836884 system.img
14180352 boot.img
13 file(s), 13 dir(s)
sunxi#fatload usb 0:1 0x42000000 boot.img
reading boot.img
14180352 bytes read in 1149 ms (11.8 MiB/s)
sunxi#mmc dev 2
mmc2(part 0) is current device
sunxi#mmc write 0x42000000 0x15000 5000
MMC write: dev # 2, block # 86016, count 20480 ... 20480 blocks written: OK
```

说明：以上操作即将 U 盘的 boot.img 写到对应的 mmc 分区地址处。

4. **fatwrite**: 从内存中将对应的文件写到设备文件系统中。

6.5 md 命令说明

md 命令可以对指定内存的数据进行查看，方便了解内存的数据情况及调试工作。其使用方法如下：

```
md 0xF0000000： 即用md命令查看内存DRAM 0xF0000000处内容。
```

6.6 FDT 命令说明

FDT：flattened device tree 的缩写在 U-Boot 控制台停下后，输入 **fdt**，可以查看 **fdt** 命令帮助。

```
sunxi#fdt
fdt - flattened device tree utility commands
Usage:
fdt addr [-c]  <addr> [<length>]   - Set the [control] fdt location to <addr>
fdt move   <fdt> <newaddr> <length> - Copy the fdt to <addr> and make it active
fdt resize                  - Resize fdt to size + padding to 4k addr
fdt print  <path> [<prop>]        - Recursive print starting at <path>
fdt list   <path> [<prop>]        - Print one level starting at <path>
fdt get value <var> <path> <prop>   - Get <property> and store in <var>
fdt get name <var> <path> <index>   - Get name of node <index> and store in <var>
fdt get addr <var> <path> <prop>    - Get start address of <property> and store in <var>
fdt get size <var> <path> [<prop>]  - Get size of [<property>] or num nodes and store in <var>
fdt set    <path> <prop> [<val>]    - Set <property> [to <val>]
fdt mknode <path> <node>          - Create a new node after <path>
fdt rm     <path> [<prop>]        - Delete the node or <property>
fdt header                  - Display header info
fdt bootcpu <id>               - Set boot cpuid
fdt memory <addr> <size>          - Add/Update memory node
fdt rsvmem print               - Show current mem reserves
fdt rsvmem add <addr> <size>       - Add a mem reserve
fdt rsvmem delete <index>         - Delete a mem reserves
fdt chosen [<start> <end>]        - Add/update the /chosen branch in the tree
```

```
                    <start>/<end> - initrd start/end addr
NOTE: Dereference aliases by omiting the leading '/', e.g. fdt print ethernet0。
sunxi#
```

说明

其中常用的命令就是 fdt list 和 fdt set，fdt list 用来查询节点配置，fdt set 用来修改节点配置。

6.6.1 查询配置

首先确定要查询的字段在 device tree 的路径，如果不知道路径，则需要用 **fdt** 命令按以下步骤进行查询。

1. 在根目录下查找。

```
sunxi#fdt list
/ {
    model = "{LICHEE_CHIP}";
    compatible = "arm,{LICHEE_CHIP}", "arm,{LICHEE_CHIP}";
    interrupt-parent = <0x00000001>;
    #address-cells = <0x00000002>;
    #size-cells = <0x00000002>;
    ......................
    cpuscfg {
    };
    ion {
    };
    dram {
    };
    memory@40000000 {
    };
    interrupt-controller@1c81000 {
    };
    sunxi-chipid@1c14200 {
    };
    timer {
    };
    pmu {
    };
    dvfs_table {
    };
    dramfreq {
    };
    gpu@0x01c40000 {
    };
    wlan {
    };
    bt {
    };
    btlpm {
    };
};
```

如果找到需要的配置，比如 wlan 的配置，运行如下命令即可。

```
sunxi#fdt list /wlan            //注意路径中的 /
wlan {
    compatible = "allwinner,sunxi-wlan";
    clocks = <0x00000096>;
    wlan_power = "vcc-wifi";
    wlan_io_regulator = "vcc-wifi-io";
    wlan_busnum = <0x00000001>;
    status = "okay";
    device_type = "wlan";
    wlan_regon = <0x00000077 0x0000000b 0x00000002 0x00000001 0xffffffff 0xffffffff 0x00000000>;
    wlan_hostwake = <0x00000077 0x0000000b 0x00000003 0x00000006 0xffffffff 0xffffffff 0x00000000>;
};
```

2. 在 soc 目录下找。如果在第一步中没有发现要找的配置，比如 nand0 的配置，则该配置可能在 soc 目录下。

```
sunxi#fdt list /soc
soc@01c00000 {
    compatible = "simple-bus";
    #address-cells = <0x00000002>;
    #size-cells = <0x00000002>;
    ranges;
    device_type = "soc";
    ......................
    hdmi@01ee0000 {
    };
    tr@01000000 {
    };
    pwm@01c21400 {
    };
    nand0@01c03000 {
    };
    thermal_sensor {
    };
    cpu_budget_cool {
    };
    ......................
};
```

然后用如下命令显示即可:

```
sunxi#fdt list /soc/nand0
nand0@01c03000 {
    compatible = "allwinner,sun50i-nand";
    device_type = "nand0";
    reg = <0x00000000 0x01c03000 0x00000000 0x00001000>;
    interrupts = <0x00000000 0x00000046 0x00000004>;
    clocks = <0x00000004 0x0000007e>;
    pinctrl-names = "default", "sleep";
    pinctrl-1 = <0x00000081>;
    nand0_regulator1 = "vcc-nand";
    nand0_regulator2 = "none";
    nand0_cache_level = <0x55aaaa55>;
    nand0_flush_cache_num = <0x55aaaa55>;
    nand0_capacity_level = <0x55aaaa55>;
    nand0_id_number_ctl = <0x55aaaa55>;
    nand0_print_level = <0x55aaaa55>;
    nand0_p0 = <0x55aaaa55>;
    nand0_p1 = <0x55aaaa55>;
```

```
        nand0_p2 = <0x55aaaa55>;
        nand0_p3 = <0x55aaaa55>;
        status = "disabled";
        nand0_support_2ch = <0x00000000>;
        pinctrl-0 = <0x000000a9 0x000000aa>;
};
```

3. 使用路径别名查找。别名是 device tree 中完整路径的一个简写，有一个专门的节点 (/aliases) 来表示别名的相关信息，用如下命令可以查看系统中别名的配置情况：

```
sunxi#fdt list /aliases
aliases {
        serial0 = "/soc@01c00000/uart@01c28000";
        .............
        mmc0 = "/soc@01c00000/sdmmc@01c0f000";
        mmc2 = "/soc@01c00000/sdmmc@01C11000";
        nand0 = "/soc@01c00000/nand0@01c03000";
        disp = "/soc@01c00000/disp@01000000";
        lcd0 = "/soc@01c00000/lcd0@01c0c000";
        hdmi = "/soc@01c00000/hdmi@01ee0000";
        pwm = "/soc@01c00000/pwm@01c21400";
        boot_disp = "/soc@01c00000/boot_disp";
};
sunxi#
```

由于配置了 nand0 节点的路径别名，因此可以用如下命令来显示 nand0 的配置信息。

```
sunxi#fdt list nand0
nand0@01c03000 {
        compatible = "allwinner,sun50i-nand";
        device_type = "nand0";
        reg = <0x00000000 0x01c03000 0x00000000 0x00001000>;
        ..................
        pinctrl-names = "default", "sleep";
        pinctrl-1 = <0x00000081>;
};
```

注：在 fdt 的所有命令中，alias 可以用作 path 参数。

```
fdt list   <path> [<prop>]        - Print one level starting at <path>
fdt set   <path> <prop> [<val>]   - Set <property> [to <val>]
```

6.6.2 修改配置

6.6.2.1 修改整数配置

命令格式：fdt set path prop <xxx> 示例：fdt set /wlan wlan_busnum <0x2>

```
sunxi#fdt list /wlan
wlan {
        compatible = "allwinner,sunxi-wlan";
        clocks = <0x00000096>;
        wlan_power = "vcc-wifi";
        wlan_io_regulator = "vcc-wifi-io";
```

```
    wlan_busnum = <0x00000001>;
    status = "disable";
    device_type = "wlan";
};
sunxi#fdt set /wlan wlan_busnum <0x2>
sunxi#fdt list /wlan
wlan {
    compatible = "allwinner,sunxi-wlan";
    clocks = <0x00000096>;
    wlan_power = "vcc-wifi";
    wlan_io_regulator = "vcc-wifi-io";
    wlan_busnum = <0x00000002>;  //修改后
    status = "disable";
    device_type = "wlan";
};
```

注：修改整数时，根据需要也可配置为数组形式，需要用空格来分隔。命令格式：**“fdt set path prop <0x1 0x2 0x3>”**

6.6.2.2 修改字符串配置

命令格式：**“fdt set path prop”xxxxx“”** 示例：**“fdt set /wlan status”disable“”**

```
sunxi#fdt list /wlan
wlan {
    compatible = "allwinner,sunxi-wlan";
    clocks = <0x00000096>;
    wlan_power = "vcc-wifi";
    wlan_io_regulator = "vcc-wifi-io";
    wlan_busnum = <0x00000001>;
    status = "okay";
    device_type = "wlan";
};
sunxi#fdt set /wlan status "disable"
sunxi#fdt list /wlan
wlan {
    compatible = "allwinner,sunxi-wlan";
    clocks = <0x00000096>;
    wlan_power = "vcc-wifi";
    wlan_io_regulator = "vcc-wifi-io";
    wlan_busnum = <0x00000001>;
    status = "disable";         //修改后
    device_type = "wlan";
};
sunxi#
```

注：修改字符串时，根据需要也可配置为数组形式，需要用空格来分隔。命令格式：**“fdt set path prop”string1” “string2” “**

6.6.3 GPIO 或者 PIN 配置特殊说明

接口对应的数字编号说明如下：

```
#define PA 0
#define PB 1
#define PC 2
#define PD 3
#define PE 4
#define PF 5
#define PG 6
#define PH 7
#define PI 8
#define PJ 9
#define PK 10
#define PL 11
#define PM 12
#define PN 13
#define PO 14
#define PP 15
#define default 0xffffffff
```

Sysconfig 中描述 gpio 的形式如下：Port: 端口 + 组内序号

6.6.3.1 Pin 配置说明

Pinctrl 节点分为 cpux 和 cpus，对应的节点路径如下：Cpux：“/soc/pinctrl@01c20800” Cpus: “/soc/pinctrl@01f02c00”

6.6.3.2 查看 PIN 配置

PIN 配置属性字段说明:

属性字段	含义
allwinner,function	对应于 sysconfig 中的主键名
allwinner,pins	对应于 sysconfig 中每个 gpio 配置中的端口名
allwinner,pname	对应于 sysconfig 中主键下面子键名字
allwinner,muxsel	功能分配
allwinner,pull	内部电阻状态
allwinner,drive	驱动能力
allwinner,data	输出电平状态

说明

其中 0xffffffff 表示使用默认值。

按以下方法查看 cpux 的 PIN 配置。

```
sunxi#fdt list /soc/pinctrl@01c20800/lcd0
lcd0@0 {
    linux,phandle = <0x000000ab>;
    phandle = <0x000000ab>;
    allwinner,pins = "PD12", "PD13", "PD14", "PD15", "PD16", "PD17", "PD18", "PD19", "PD20", "PD21";
```

```
        allwinner,function = "lcd0";
        allwinner,pname = "lcdd0", "lcdd1", "lcdd2", "lcdd3", "lcdd4", "lcdd5", "lcdd6", "lcdd7", "lcdd8", "lcdd9";
        allwinner,muxsel = <0x00000003>;
        allwinner,pull = <0x00000000>;
        allwinner,drive = <0xffffffff>;
        allwinner,data = <0xffffffff>;
};
sunxi#
```

按以下方法查看 cpus 的 PIN 配置。

```
sunxi#fdt list /soc/pinctrl@01f02c00/s_uart0
s_uart0@0 {
        linux,phandle = <0x000000b4>;
        phandle = <0x000000b4>;
        allwinner,pins = "PL2", "PL3";
        allwinner,function = "s_uart0";
        allwinner,pname = "s_uart0_tx", "s_uart0_rx";
        allwinner,muxsel = <0x00000002>;
        allwinner,pull = <0xffffffff>;
        allwinner,drive = <0xffffffff>;
        allwinner,data = <0xffffffff>;
};
sunxi#
```

6.6.3.3 修改 PIN 配置

使用 **fdt set** 命令可以修改 PIN 中相关属性字段

```
sunxi#fdt set /soc/pinctrl@01c20800/lcd0 allwinner,drive <0x1>
sunxi#fdt list /soc/pinctrl@01c20800/lcd0
lcd0@0 {
        linux,phandle = <0x000000ab>;
        phandle = <0x000000ab>;
        allwinner,pins = "PD12", "PD13", "PD14", "PD15", "PD16", "PD17", "PD18", "PD19", "PD20", "PD21";
        allwinner,function = "lcd0";
        allwinner,pname = "lcdd0", "lcdd1", "lcdd2", "lcdd3", "lcdd4", "lcdd5", "lcdd6", "lcdd7", "lcdd8", "lcdd9";
        allwinner,muxsel = <0x00000003>;
        allwinner,pull = <0x00000000>;
        allwinner,drive = <0x00000001>;
        allwinner,data = <0xffffffff>;
};
```

说明

示例中该处修改会影响 “allwinner,pins” 表示的所有端口的驱动能力配置，修改 “allwinner,muxsel”，“allwinner,pull”，“allwinner,data” 的值也会产生类似效果。

6.6.3.4 GPIO 配置说明

Device tree 中 GPIO 对应关系，以 usb 中 usb_id_gpio 为例

```
sunxi#fdt list /soc/usbc0
usbc0@0 {
    test = <0x00000002 0x00000003 0x12345678>;
    device_type = "usbc0";
    compatible = "allwinner,sun50i-otg-manager";
    ........
    usb_serial_unique = <0x00000000>;
    usb_serial_number = "20080411";
    rndis_wceis = <0x00000001>;
    status = "okay";
    usb_id_gpio = <0x00000030 0x00000007 0x00000009 0x00000000 0x00000001 0xffffffff 0xffffffff>;
};
```

对应于 device tree 中 **“usb_id_gpio = <0x00000030 0x00000007 0x00000009 0x00000000 0x00000001 0xffffffff 0xffffffff>”**，解释如下：

属性数值	含义
0x00000030	device tree 内部一个节点相关信息，这里可以略过
0x00000007	端口 PH，即 #define PH 7
0x00000009	组内序号，即 PH09
0x00000000	功能分配，即将 PH09 配为输入
0x00000001	内部电阻状态，即配为上拉
0xffffffff	驱动能力，默认值
0xffffffff	输出电平，默认值

如果需要修改 usb_id_gpio 的配置，可按如下方式（示例修改了驱动能力，输出电平两项）：

```
sunxi#fdt set /soc/usbc0 usb_id_gpio <0x00000030 0x00000007 0x00000009 0x00000000 0x00000001 0x2 0x1>
sunxi#fdt list
usbc0@0 {
    test = <0x00000002 0x00000003 0x12345678>;
    device_type = "usbc0";
    compatible = "allwinner,sun50i-otg-manager";
     ........
    usb_serial_unique = <0x00000000>;
    usb_serial_number = "20080411";
    rndis_wceis = <0x00000001>;
    status = "okay";
    usb_id_gpio = <0x00000030 0x00000007 0x00000009 0x00000000 0x00000001 0x00000002 0x00000001>;  //修改ok
};
sunxi#
```

6.7 其他命令说明（boot，reset，efex）

1. boot : 启动内核
2. reset: 复位重启系统
3. efex: 进入烧录状态

说明

注：其他更多 U-Boot 命令介绍，请进入 U-Boot shell 命令状态后输入 “help” 进行了解。

7 基本调试方法介绍

debug 调试信息介绍如下：

1. debug_mode

   debug_mode 可以控制 Boot0 的打印等级，打开文件 {LICHEE_BOARD_CONFIG_DIR}/sys_config.fex，在主键 [platform] 下添加子键 **”debug_mode = 8” 即表示开启所有打印，debug_mode=0 表示关闭启动时 Boot0 的打印 log，未显式配置 debug_mode 时，按 debug_mode=8 处理。目前常用的打印等级有 0（关闭所有打印）、1（只显示关键节点打印）、4（打印错误信息）、8（打印所有 log 信息）。

   debug_mode 可以控制 U-Boot 的打印等级，打开文件 {LICHEE_BOARD_CONFIG_DIR}/b3/uboot-board.dts，在 platform 节点下添加子键”**debug_mode = 8**” 即表示开启所有打印，debug_mode=0 表示关闭启动时 U-Boot 的打印 log，未显式配置 debug_mode 时，按 debug_mode=8 处理。目前常用的打印等级有 0（关闭所有打印）、1（只显示关键节点打印）、4（打印错误信息）、8（打印所有 log 信息）。

2. usb_debug 在烧录或启动过程中，若遇到烧录失败或启动失败大致挂死在 usb 相关模块，但又不确定具体位置，这时可以打开 usb_debug 进行调试，开启 usb_debug 后有关 usb 相关的运行信息会被较详细打印出来。打开 usb_debug 的方式：打开 usb_base.h 文件，将其中的 #defineSUNXI_USB_DEBUG 宏定义打开，打开后重新编译 U-Boot 并打包烧录即可。

8 烧录方法

目前提供以下几种烧录方法，请按照情况选择：

1. 开机时按住 fel 键。
2. 开机时打开串口按住键盘数字 “2”。
3. 进入 U-Boot 控制台输入 “efex”。
4. 进入 Android 控制台输入 “reboot efex”。
5. 长按 phy key 键，当出现 volume key pressed 打印后，在 3 秒内按 3 次电源键。

第五种烧写方法需确定板子使用的是何种硬件 phy key，不同 phy key，在设备树下的修改不同，具体如下：

gpio key

```
uboot-board.dts:
&soc {
    key_detect_en: key_detect_en {
        key_type = "gpiokey"; /* 填写phy key类型 */
        /* 需指明两个音亮键的gpio */
        volume-down-gpios = <&pio 0x1 0x0 0xe 0xffffffff 0xffffffff 0x1>;/* 注，仅需关注前3个参数： &pio 代表pio节点，0x1 代表 PB, 0x0 代表 PB 中的port 0。其余参数是为了补齐格式，按当前设置即可 */
        volume-up-gpios = <&pio 0x1 0x1 0xe 0xffffffff 0xffffffff 0x1>;/* 注，仅需关注前3个参数： &pio 代表pio节点，0x1 代表 PB, 0x1 代表 PB 中的port 0。其余参数是为了补齐格式，按当前设置即可 */
        active-level = <0>; /* 写0无需修改 */
    };
};
```

lradc key

```
uboot-board.dts:
&soc {
    key_detect_en: key_detect_en {
        key_type = "lrkey";  /* 填写phy key类型 */
        adc_channel = <0>;  /* 根据实际情况配置adc通道 */
        keyen_flag = <1>;   /* 写1无需修改 */
    };
};
```

gpadc key

```
uboot-board.dts:
&soc {
    key_detect_en: key_detect_en {
        key_type = "gpkey";  /* 填写phy key类型 */
        adc_channel = <0>;  /* 根据实际情况配置adc通道 */
```

```
        keyen_flag = <1>;   /* 写1无需修改 */
    };
};
```

9 常用接口函数

9.1 fdt 相关接口

1. const void *fdt_getprop(const void *fdt, int nodeoffset, const char *name, int *lenp)

- 作用：检索指定属性的值
- 参数：
  - fdt: 工作 flattened device tree
  - nodeoffset: 待修改节点的偏移
  - name: 待检索的属性名
  - lenp: 检索属性值的长度（会被覆盖）或者为 NULL
- 返回：
  - 非空（属性值的指针）：成功
  - NULL（lenp 为空）：失败
  - 失败代码（lenp 非空）：失败

2. int fdt_set_node_status(void *fdt, int nodeoffset, enum fdt_status status, unsigned int error_code)

- 作用：设置节点状态
- 参数：
  - fdt: 工作 flattened device tree
  - nodeoffset: 待修改节点的偏移
  - status:FDT_STATUS_OKAY, FDT_STATUS_DISABLED, FDT_STATUS_FAIL, FDT_STATUS_FAIL_ERROR_CODE
  - error_code:optional, only used if status is FDT_STATUS_FAIL_ERROR_CODE
- 返回：
  - 0: 成功
  - 非 0: 失败

3. int fdt_path_offset(const void *fdt, const char *path)

- 作用：通过全路径查找节点的偏移量
- 参数：

  - fdt: 工作 fdt
  - path: 全路径名称

- 返回：
  - >=0(节点的偏移量): 成功
  - <0: 失败代码

4. static inline int fdt_setprop_u32(void *fdt, int nodeoffset, const char *name, uint32_t val)

- 作用：将属性值设置为一个 32 位整型数值，如果属性值不存在，则新建该属性
- 参数：
  - fdt: 工作 flattened device tree
  - nodeoffset: 待修改节点的偏移
  - name: 待修改的属性名
  - val:32 位目标值
- 返回：
  - 0: 成功
  - <0: 失败代码

5. static inline int fdt_setprop_u64(void *fdt, int nodeoffset, const char *name, uint64_t val)

- 作用：与fdt_setprop_u32类似，将属性值设置为一个 64 位整型数值，如果属性值不存在，则新建该属性
- 参数：
  - fdt: 工作 flattened device tree
  - nodeoffset: 待修改节点的偏移
  - name: 待修改的属性名
  - val:64 位目标值
- 返回：
  - 0: 成功
  - <0: 失败代码

6. #define fdt_setprop_string(fdt, nodeoffset, name, str) fdt_setprop((fdt), (nodeoffset), (name), (str), strlen(str)+1)

- 作用：将属性值设置为一个字符串，如果属性值不存在，则新建该属性
- 参数：
  - fdt: 工作 flattened device tree
  - nodeoffset: 待修改节点的偏移

  - name: 待修改的属性名
  - str: 目标值

- 返回：
  - 0: 成功
  - <0: 失败代码

注意：在sys_config.fex的配置中，节点的启用状态为 0 或 1。转换到 fdt 中对应的 status 属性为disable或okay。

7. int save_fdt_to_flash(void *fdt_buf, size_t fdt_size)

- 作用：保存修改到 flash
- 参数：
  - fdt_buf: 当前工作 flattened device tree
  - fdt_size: 当前工作 flattened device tree 的大小，可以通过fdt_totalsize(fdt_buf)获取
- 返回：
  - 0: 成功
  - <0: 失败

8. 应用参考

U-Boot 中 fdt 命令行的实现：cmd/fdt.c

9.2 env 相关接口函数

1. int env_set(const char *varname, const char *varvalue)

- 作用：将环境变量 varname 的值设置为 varvalue，重启失效
- 参数：
  - varname: 待设置环境变量的名称
  - varvalue: 将指定的环境变量修改为该值
- 返回：
  - 0: 成功
  - 非 0: 失败

2. char *env_get(const char *name)

- 作用：获取指定环境变量的值
- 参数：
  - name: 变量名称
- 返回：
  - NULL: 失败
  - 非空（环境变量的值）：成功

3. int env_save(void)

- 作用：保存环境变量，重启仍保存
- 参数: 无
- 返回：
  - 0: 成功
  - 非 0: 失败

4. 应用参考

board/sunxi/sunxi_bootargs.c update_bootargs通过 cmdline 向 kernel 提供信息，主要是通过更新bootargs变量实现env_set("bootargs", cmdline)。

9.3 调用 U-Boot 命令行

1. int run_command_list(const char *cmd, int len, int flag)

- 作用：执行 U-Boot 命令行
- 参数：
  - cmd: 命令字符指针
  - len: 命令行长度，设置为-1 则自动获取
  - flag: 任意，因为 sunxi 中没有用到
- 返回：
  - 0: 成功
  - 非 0: 失败

2. 应用参考：

common/autoboot.c autoboot_command实现了 U-Boot 的自动启动命令

s = env_get("bootcmd");

run_command_list(s, -1, 0)。

9.4 Flash 的读写

1. int sunxi_flash_read(uint start_block, uint nblock, void *buffer)

- 作用：将指定起始位置start_block的nblock读取到buffer
- 参数：
  - start_block: 起始地址
  - nblock:block 个数
  - buffer: 内存地址
- 返回：
  - 0: 成功
  - 非 0: 失败

2. int sunxi_flash_write(uint start_block, uint nblock, void *buffer)

- 作用：将buffer写入指定起始位置start_block的nblock中
- 参数：
  - start_block: 起始地址
  - nblock:block 个数
  - buffer: 内存地址
- 返回：
  - 0: 成功
  - 非 0: 失败

3. int sunxi_sprite_read(uint start_block, uint nblock, void *buffer)

- 作用与sunxi_flash_read相似

4. int sunxi_sprite_write(uint start_block, uint nblock, void *buffer)

- 作用与sunxi_flash_write相似

5. 应用参考

common/sunxi/board_helper.c sunxi_set_bootcmd_from_mis实现了对 misc 分区的读写操作

9.5 获取分区信息

1. int sunxi_partition_get_partno_byname(const char *part_name)

- 作用：根据分区名称获取分区号
- 参数：
  - part_name: 分区名称
- 返回：
  - <0: 失败
  - >0（分区号）： 成功

2. int sunxi_partition_get_info_byname(const char *part_name, uint *part_offset, uint *part_size)

- 作用：根据分区名称获取分区的偏移量和大小
- 参数：
  - part_name: 分区名称
  - part_offset: 分区的偏移量
  - part_size: 分区的大小
- 返回：
  - 0: 成功
  - -1: 失败

3. uint sunxi_partition_get_offset_byname(const char *part_name)

- 作用：根据分区名称获取偏移量
- 参数：
  - part_name: 分区名称
- 返回：
  - <=0 : 失败
  - >0 : 成功

4. int sunxi_partition_get_info(const char *part_name, disk_partition_t *info)

- 作用：根据part_name获取分区信息
- 参数：

  - part_name: 分区名称
  - info: 分区信息

- 返回：
  - 非 0: 失败
  - 0: 成功

5. lbaint_t sunxi_partition_get_offset(int part_index)

- 作用：card sprite 模式下获取分区的偏移量
- 参数：
  - part_index: 分区号
- 返回：
  - >=0（偏移量）：成功
  - -1: 失败

6. 应用参考

启动时加载图片：drivers/video/sunxi/logo_display/sunxi_load_bmp.c

9.6 GPIO 相关操作

1. int fdt_get_one_gpio(const char* node_path, const char* prop_name,user_gpio_set_t* gpio_list)

- 作用：根据路径node_path和 gpio 名称prop_name获取 gpio 配置
- 参数：
  - node_path：fdt 路径
  - prop_name：gpio 名称
  - gpio_list：待获取的 gpio 信息
- 返回：
  - 0：成功
  - -1：失败

2. ulong sunxi_gpio_request(user_gpio_set_t *gpio_list, __u32 group_count_max)

- 作用：根据 gpio 配置获取 gpio 操作句柄

- 参数：
  - gpio_list：gpio 配置列表，可以由fdt_get_one_gpio获得
  - group_count_max: gpio_list中最大的 gpio 配置个数
- 返回：
  - 0：失败
  - >0（gpio 操作句柄）：成功

3. __s32 gpio_write_one_pin_value(ulong p_handler, __u32 value_to_gpio, const char *gpio_name)

- 作用：根据 gpio 操作句柄写数据
- 参数：
  - p_handler：gpio 操作句柄，可由sunxi_gpio_request获取
  - value_to_gpio：待写入数据，0 或 1
  - gpio_name：gpio 名称
- 返回：
  - EGPIO_SUCCESS：成功
  - EGPIO_FAIL：失败

4. 应用参考

操作 led 状态：

```
ssprite/sprite_led.c

user_gpio_set_t gpio_init;

fdt_get_one_gpio("/soc/card_boot", "sprite_gpio0", &gpio_init);  //获取/soc/card_boot中sprite_gpio0的gpio配置

sprite_led_hd = sunxi_gpio_request(&gpio_init, 1);  //获取gpio操作句柄

gpio_write_one_pin_value(sprite_led_hd, sprite_led_status, "sprite_gpio0");  //操作led状态
```

10 常用资源的初始化阶段

- env： 环境变量初始化后可以访问
- fdt： 在 U-Boot 运行开始即可访问
- malloc： 在重定位后才能访问

11 FAQ

11.1 Erase flag 注意事项

11.1.1 具体表现

在 uboot_board.dts 中的 platform 下，有一个 eraseflag 字段。

```
&platform {
    eraseflag = <1>;
    debug_mode = <4>;
};
```

该字段会决定是否执行烧写是是否执行擦除操作。

```
0x0：不进行擦除。
0x01：进行擦除，private，secure storage、user data、mbr分区保留，其他擦除。
0x11：进行擦除，private，secure storage分区保留，其他擦除。
0x12：强制擦除(整个flash)。
```

11.1.2 erase flag 对量产工具的影响

1. PhoenixSuit 工具。

   - 单或多分区下载（只下载所选分区）：该按钮一旦勾选，并在下拉菜单中选择对应分区，烧录操作只会覆盖对应分区，不会影响其他分区。此时 erase flag = 0，不进行擦除操作。
   - 保留数据升级：勾选该按钮，则升级前不进行格式化. 此时 erase flag = 0x1,private，secure storage、user data、mbr 分区保留，其他擦除。
   - 分区擦除升级：勾选该按钮，则升级前只格式化普通分区。此时 erase flag = 0x11,private，secure storage 分区保留，其他擦除。
   - 全盘擦除升级：勾选该按钮，则升级前格式化全盘，包括 SN 等数据。此时 erase flag = 0x12，整块 flash 全部擦除。

2. PhoenixCard 工具。erase flag 的值取决于 sysconfig.fex 得 erase flag 字段。

3. PhoenixUsbPro 工具。存在两中情况:

   - 不勾选工具得全盘擦除。erase flag 的值取决于 sysconfig.fex 的 erase flag 字段。
   - 勾选工具得全盘擦除。此时 erase flag = 0x12，整块 flash 全部擦除。

11.2 sunxi uboot 平台在 uboot-board.dts 增加一个节点报错

11.2.1 问题背景

产品：

硬件：v853 + perf1 （型号）

软件：uboot

其他：用户需要在 uboot-board.dts 上增加一个节点，增加后编译报错

11.2.2 问题描述

1. 复现步骤在板级目录下的 uboot 增加 boot_init_gpio 节点。

```
diff --git a/configs/perf1/uboot-board.dts b/configs/perf1/uboot-board.dts
index f954a99..2a94a32 100755
--- a/configs/perf1/uboot-board.dts
+++ b/configs/perf1/uboot-board.dts
@@ -56,6 +56,13 @@
     };
 };
+&boot_init_gpio {
+           device_type = "boot_init_gpio";
+           logical_start = <40960>;
+           /* sprite_gpio0 = <&pio PH 6 1 0xffffffff 0xffffffff 1>; */
+           gpio0 = <&pio 0xb 0x1 0x1 0x1 0x0 0x1>;
+};
+
 &power_sply {
     dcdc1_vol = <1003300>;
     dcdc2_vol = <1000900>;
```

2. 具体表现重新编译 uboot 后报如下错误。

```
  DTC     arch/arm/dts/sun8iw21p1-soc-system.dtb
  SYM     u-boot.sym
Error: arch/arm/dts/.board-uboot.dts:59.1-16 Label or path boot_init_gpio not found
FATAL ERROR: Syntax error parsing input tree
make[2]: *** [arch/arm/dts/sun8iw21p1-soc-system.dtb] Error 1
make[1]: *** [arch/arm/dts/sun8iw21p1-soc-system.dtb] Error 2
make: *** [dts/dt.dtb] Error 2
make: *** Waiting for unfinished jobs....
make: *** wait: No child processes.  Stop.
ERROR: build brandy Failed
```

11.2.3 问题分析

根据打印可以看出为设备树找不到 boot_init_gpio 节点。

11.2.4 根本原因

uboot 的 dts 是通过源码中芯片的 soc-system.dts 和 device 仓库中的 uboot-board.dts 两个文件组成，如果需要增加一个不存在的节点，首先要在 uboot 源码中 soc-system.dts 文件里面添加。

11.2.5 解决办法

根据打印可以找到 uboot 源码中的 dts 是 arch/arm/dts/sun8iw21p1-soc-system.dts，在此文件添加 boot_init_gpio 即可通过编译。

```
diff --git a/arch/arm/dts/sun8iw21p1-soc-system.dts b/arch/arm/dts/sun8iw21p1-soc-system.dts
index 195c6fc7ef..1ce7f93fa6 100644
--- a/arch/arm/dts/sun8iw21p1-soc-system.dts
+++ b/arch/arm/dts/sun8iw21p1-soc-system.dts
@@ -31,6 +31,8 @@
                device_type = "platform";
        };
+       boot_init_gpio:boot_init_gpio@4 {
+       };

        target:target@45000008 {
                device_type = "target";
```

11.3 如何关闭检测烧写 key 提高启动时间

11.3.1 问题背景

产品：硬件：H618 + P1 软件：uboot2018 其他：如何关闭 uboot 中的烧 key 流程，从而提高启动时间

key 用于保存设备硬件相关的信息，在量产是不会被擦除根据，根据存储位置的不同，主要有以下几种：

- 安全 key(secure storage)：

> 保存在 flash 上，使用的扇区未映射到逻辑扇区。通常的 flash 操作无法访问。正常量产不会被擦除。- 私有 key

保存在 private 分区。量产时会把此分区内容读取到内存，量产完毕后从内存恢复到 flash 中，达到量产后分区内容保留的目的。key 保存到文件名为 key 名称的文件中，可以通过文件系统访问到 key 的内容。

11.3.2 问题描述

1. 打开和关闭检测 usb key 时间对比

打开方式

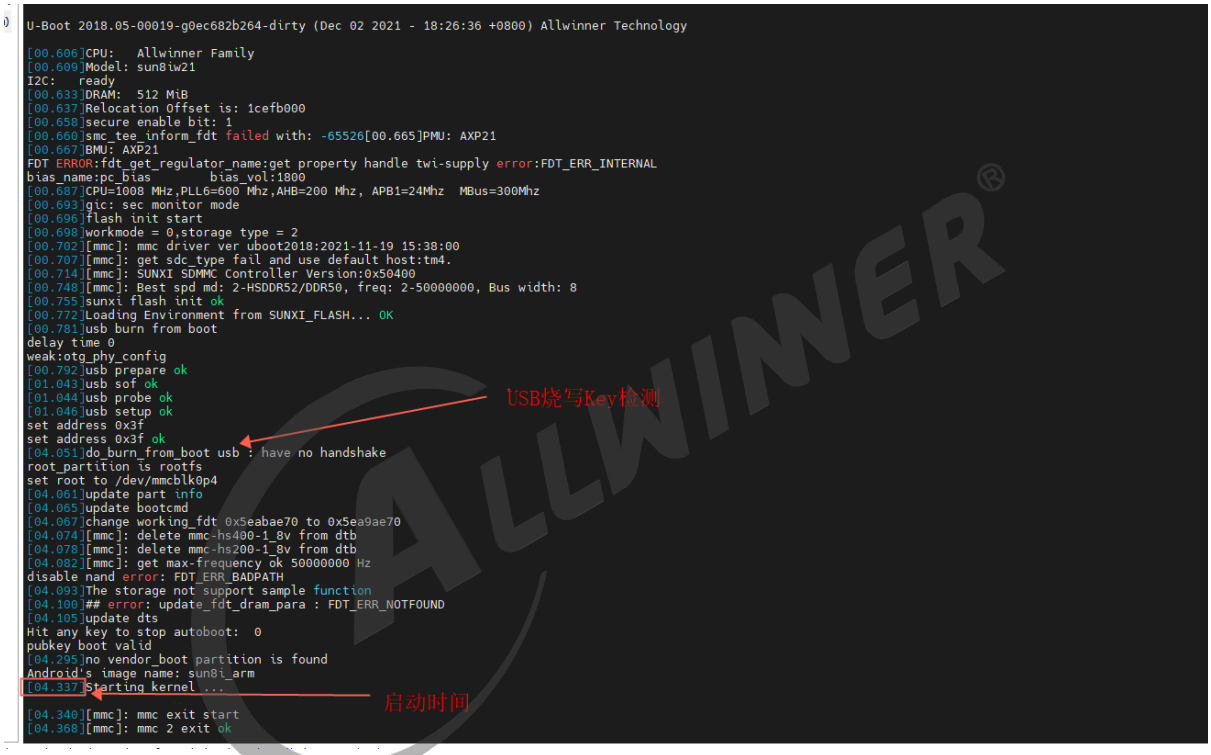

图 11-1: 打开方式启动时间

关闭方式

图 11-2: 关闭方式启动时间

2. 具体表现

以上可以看出当打开始启动时间为 4.337 秒，打开时 log 有打印 do_burn_from_boot usb 字段，关闭方式的启动时间为 1.090 秒。

11.3.3 解决办法

1. 方式一：打开 dragonSN 工具，点击配置 key，在跳出来的界面当中点击全局配置，在跳出来的界面当中设置烧写位为 1，点击确定。

根据需要写添加一些 key。

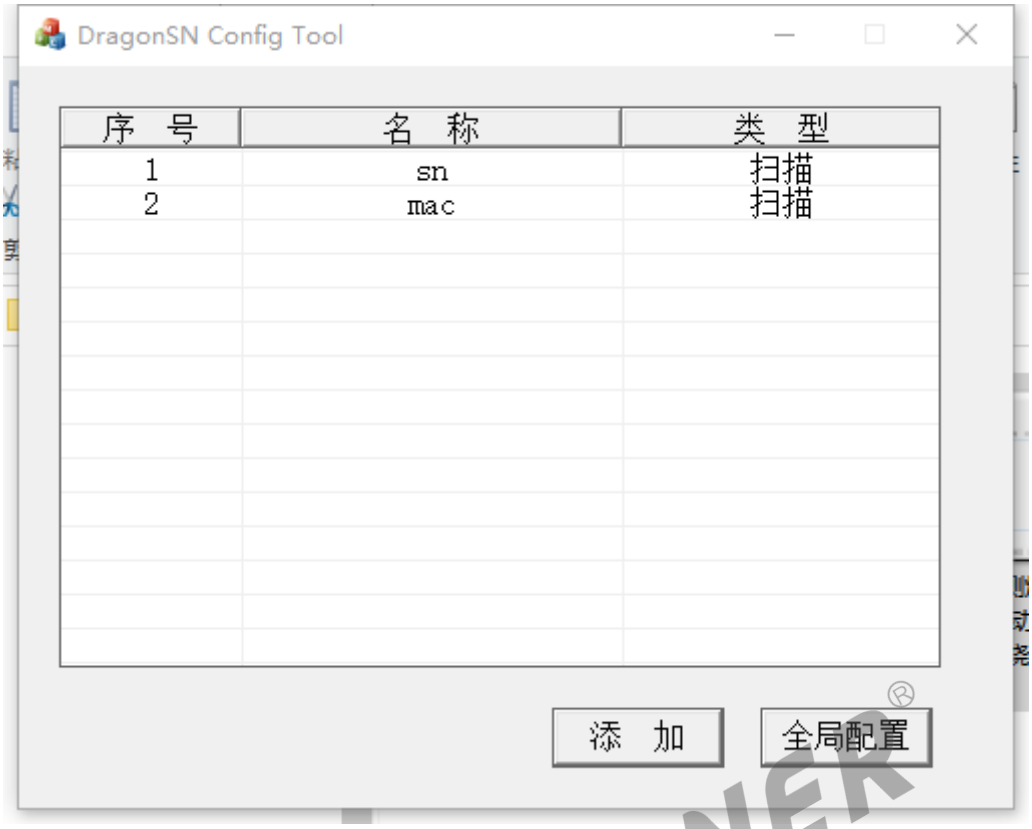

图 11-3: 添加 key 界面

在界面中写一些 key 的值，点击烧写 key，烧写成功后便生效了。图中的 123456789 是示例值，用户可以填入自己的值。

图 11-4: 烧写成功界面

2. 方式二：在 uboot shell 擦除 key_burned_flagl 关闭烧 key 流程。进入 uboot shell，在命令行界面输入：

```
=> pst erase key_burned_flag
```

此方式是将 secure storage 中的 key_burned_flag 擦除避免下次还进烧 key 流程。3. 方式三：使用环境变量关闭烧 key 流程。

进入 uboot shell，在命令行界面输入：

```
=> setenv burn_key 0
```

4. 方式四：更改 uboot dts 关闭烧 key 流程。

打开 SDK，找到在 device 目录下找到芯片对应的板级目录。将 target 节点下的 burn_key 的属性改成 0 即可。

```
&target {
    advert_enable = <1>;
    burn_key = <1>;
    auto_fel = <0>;
};
```

5. 总结

sunxi 平台的启动流程中有一个烧 key 流程，此过程占其启动时间 1-3 秒，关闭方式有四种，其中第一种和第二种针对安全 key 方案，第三种和第四种无论安全 key 方案还是私有 key 方案都支持。

11.4 缺失私有分区导致的烧 key 失败

11.4.1 问题背景

产品：

硬件：v853 + perf1 （型号）

软件：uboot2018

其他：烧写私有 key 失败

key 用于保存设备硬件相关的信息，在量产是不会被擦除根据，根据存储位置的不同，主要有以下几种：

- 安全 key(secure storage)：

> 保存在 flash 上，使用的扇区未映射到逻辑扇区。通常的 flash 操作无法访问。正常量产不会被擦除。

- 私有 key

> 保存在 private 分区。量产时会把此分区内容读取到内存，量产完毕后从内存恢复到 flash 中，达到量产后分区内容保留的目的。key 保存到文件名为 key 名称的文件中，可以通过文件系统访问到 key 的内容。

在当前 nor 方案上默认不支持 secure storage 分区，即不支持安全 key，用户如果需要 key 功能，可以使用私有 key，一般非安卓方案是没有私有分区的，容易引发错误。

11.4.2 问题描述

1. 复现步骤。

> - 样机烧录正确的固件。
> - 打开 dragonSN，全局配置中选择私有 key，添加 snum 和 mac 的 key，key type 为 flash。
> - usb 连接 PC 上电，识别到样机后烧写。

2. 具体表现。烧写失败，样机端出现报错 log private partition is not exist。

11.4.3 问题分析

私有 key 保存在 private 分区中，量产时会把此分区内容读取到内存，量产完毕后从内存恢复到 flash 中，达到量产后分区内容仍存在的目的。

11.4.4 根本原因

没有建立私有分区

11.4.5 解决办法

建立私有分区，找到此芯片对应的 device 仓库中的 sys_partition.fex 文件，添加如下内容即可。

```
diff --git a/configs/default/sys_partition.fex b/configs/default/sys_partition.fex
index 50e11c7..e0bc8ae 100644
--- a/configs/default/sys_partition.fex
+++ b/configs/default/sys_partition.fex
@@ -52,6 +52,12 @@ size = 16384
    downloadfile = "boot.fex"
    user_type    = 0x8000

+[partition]
+    name       = private
+    size       = 32768
+    ro         = 0
+    user_type    = 0x8000
+
 [partition]
    name       = rootfs
    size       = 40320
```

11.5 如何通过启动 log 判断卡启动和 emmc 启动

11.5.1 问题描述

当前有一台 emmc 介质的样机和 SD 卡，在测试卡启动时分辨系统是不是真的从 SD 卡引导系统。

11.5.2 解决办法

- 将固件直接烧录样机后观察 boot0 打印，出现 card no is 2。
- 制作启动卡，插入卡槽，观察 boot0 打印，出现 card no is 0。

11.6 emmc 方案如何将 uboot 的备份分区作为第一个引导分区

11.6.1 问题背景

产品：

硬件：T507 + deme2.0（型号）。

软件：uboot 2018（版本)。

其他：特有版本信息添加自由描述（如固件版本，复现概率，特定环境）。

11.6.2 问题分析

uboot emmc 方案默认存在两份 uboot，默认使用物理 32800 扇区开始的 uboot，如何更改使用顺序。

11.6.3 解决办法

uboot 是由 boot0 引导的，在 boot0 的代码打下如下补丁即可改变引导顺序。

- sunxi_flashmap_toc1_bak_start(FLASHMAP_SDMMC) 是获取 uboot 分区的起始物理扇区。
- sunxi_flashmap_toc1_bak_start(FLASHMAP_SDMMC) 是获取 uboot_b 分区的起始物理扇区。

```
diff --git a/nboot/load_image_mmc/load_image_sdmmc.c b/nboot/load_image_mmc/load_image_sdmmc.c
index 966b2356..f1ac6cea 100644
--- a/nboot/load_image_mmc/load_image_sdmmc.c
+++ b/nboot/load_image_mmc/load_image_sdmmc.c
@@ -101,8 +101,8 @@ int load_toc1_from_sdmmc(char *buf)
            goto __ERROR_EXIT;;
        }

-       start_sectors[0] = sunxi_flashmap_toc1_start(FLASHMAP_SDMMC);
-       start_sectors[1] = sunxi_flashmap_toc1_bak_start(FLASHMAP_SDMMC);
+       start_sectors[0] = sunxi_flashmap_toc1_bak_start(FLASHMAP_SDMMC);
+       start_sectors[1] = sunxi_flashmap_toc1_start(FLASHMAP_SDMMC);

        for(i=0; i < 4; i++)
        {
```

11.7 无法烧录安全固件

11.7.1 问题背景

产品：扫地机

硬件：MR122

软件：tina3.5.1

11.7.2 问题描述

MR112 无法烧录安全固件

11.7.3 根本原因

出现空板 MR112 无法烧录安全固件问题，是因为 sys_config.fex 中没有配置 burn_secure_mode=1 和 secure_without_OS=0 这两个配置，从而导致空板无法烧录安全固件，这两配置就是使能非安全空间烧录安全固件。

11.7.4 解决办法

配置 burn_secure_mode=1 和 secure_without_OS=0，通过 “pack -s” 命令将会往 sys_config.fex 增加该配置。可以在当前出现问题的环境进行确认，为什么打包脚本没有设置生效。确认打包脚本中有以下配置的内容：

```
\```
if [ "x${PACK_SECURE}" = "xsecure" -o "x${PACK_SIG}" = "xsecure" ] ; then
printf "add burn_secure_mode in target in sys config\n"
sed -i -e '/^\[target\]/a\burn_secure_mode=1' ${ROOT_DIR}/image/sys_config${SUFFIX}.fex
sed -i -e '/^\[platform\]/a\secure_without_OS=0' ${ROOT_DIR}/image/sys_config${SUFFIX}.fex
elif [ "x${PACK_SIG}" = "xprev_refurbish" ] ; then
printf "add burn_secure_mode in target in sys config\n"
sed -i -e '/^\[target\]/a\burn_secure_mode=1' ${ROOT_DIR}/image/sys_config${SUFFIX}.fex
sed -i -e '/^\[platform\]/a\secure_without_OS=1' ${ROOT_DIR}/image/sys_config${SUFFIX}.fex
else
sed -i '/^burn_secure_mod/d' ${ROOT_DIR}/image/sys_config${SUFFIX}.fex
sed -i '/^secure_without_OS/d' ${ROOT_DIR}/image/sys_config${SUFFIX}.fex
fi
\```
```

这个确认后续的 sys_config.fex 有这两配置生效即可。

11.8 插上电源是否开机配置

11.8.1 问题背景

产品：H700 投影

硬件:H616

软件系统：H616 AndroidQ

11.8.2 问题描述

插入电源，板子立即开机，需要不会立即开机，需要按电源键（也包括遥控器的 power 键）之后才开机

11.8.3 问题分析

需要 PMU 重新烧码

11.8.4 解决办法

如图所示，需要配置对应的参数 start_type 和 pmukey_uesd

2.1.5 [box_start_os]

配置项	配置项含义
used	是否启用该项功能: 1：启用 0：不启用
start_type	是否上电启动系统 1：直接启动系统；0：上电不允许直接启动系统
irkey_used	是否启用 ir 控制启动：1：启用 ir 按键启动 0：禁用 ir 按键启动
pmukey_used	1: 启用 PMU power key 启动 0：禁用 PMU power key

图 11-5: 开机电源配置项介绍

修改 start_type 配置成 0，pmukey_uesd 配置成 1

start_type 的修改位置如下

```
Binary files a/bin/bl31.bin and b/bin/bl31.bin differ
diff --git a/configs/tv/board.dts b/configs/tv/board.dts
index 2347c7d..87bdbd8 100755
--- a/configs/tv/board.dts
+++ b/configs/tv/board.dts
@@ -888,7 +888,7 @@
        };
        box_start_os0 {
                compatible = "allwinner,box_start_os";
-               start_type = <0x1>;
+               start_type = <0x0>;
                irkey_used = <0x0>;
                pmukey_used = <0x0>;
                pmukey_num = <0x0>;
```

图 11-6: start_type 修改示意图

11.9 spl 读取 adc 的值

11.9.1 问题背景

软件：V85X_ARTMOS

硬件：V85X

11.9.2 问题描述

spl 阶段获取 adc 的值

11.9.3 解决办法

在相应的配置宏文件的地方添加

```
+CFG_SUNXI_PHY_KEY=y
+CFG_GPADC_KEY=y
```

在需要 spl 相应的代码添加 sunxi_read_gpadc_vol 函数获取 adc 的值。

参考如下

```
diff --git a/board/sun8iw21p1/commonfastboot.mk b/board/sun8iw21p1/commonfastboot.mk
index 6e5d1e2c..ab955f6c 100644
--- a/board/sun8iw21p1/commonfastboot.mk
+++ b/board/sun8iw21p1/commonfastboot.mk
@@ -13,6 +13,10 @@ CFG_SBOOT_RUN_ADDR=0x20480
 CFG_SUNXI_GPIO_V2=y
 CFG_SUNXI_FDT=y

+CFG_SUNXI_PHY_KEY=y
+CFG_GPADC_KEY=y
+
+
 #LOGO
 CFG_BOOT0_LOGO_TO_KERNEL=y

diff --git a/board/sun8iw21p1/board.c b/board/sun8iw21p1/board.c
index b252a7bd..59edc09e 100644
--- a/board/sun8iw21p1/board.c
+++ b/board/sun8iw21p1/board.c
@@ -146,6 +146,12 @@ uint8_t audioldo_check(void)
 uint8_t sunxi_board_late_init(void)
 {
     audioldo_check();
+#ifdef CFG_SUNXI_PHY_KEY
+#ifdef CFG_GPADC_KEY
+    int sunxi_read_gpadc_vol(int channel);
+    printf("gpadc val:0x%x\n", sunxi_read_gpadc_vol(0));
+#endif
+#endif
     return 0;
 }
```

11.10 替换开机 logo

11.10.1 问题背景

产品：扫描笔

硬件：R818

软件：R818 Tina

11.10.2 问题描述

替换开机 logo

11.10.3 问题分析

替换开机 logo 的方法，并且可以通过 OTA 对已售机器进行升级

11.10.4 解决方法

1. 把新的 logo 放到 target/allwinner/generic/boot-resource/boot-resource 目录，并重命名为 bootlogo.bmp。注意图片格式要 bmp 的

2. 在 target/allwinner/generic/swupdate 打上补丁 0001-OTA-support-upgrade-logo.patch

补丁的内容如下

```
new file mode 100644
index 00000000..4481bfa2
--- /dev/null
+++ b/allwinner/generic/swupdate/sw-description-logo
@@ -0,0 +1,61 @@
+software =
+{
+    version = "0.1.0";
+    description = "Firmware update for Tina Project";
+
+    stable = {
+
+        /* upgrade boot-resource for new logo ==> reboot */
+        upgrade_logo = {
+            /* upgrade boot-resource */
+            images: (
+                {
+                    filename = "new_logo";
+                    device = "/dev/by-name/boot-resource";
+                    installed-directly = true;
+                }
+            );
+            /* clean & reboot*/
+            bootenv: (
+                {
+                    name = "swu_param";
+                    value = "";
+                },
+                {
+                    name = "swu_software";
+                    value = "";
+                },
+                {
+                    name = "swu_mode";
```

```
+                value = "";
+            },
+            {
+                name = "swu_next";
+                value = "reboot";
+            }
+        );
+    };
+
+};
+
+/* when not call with -e xxx,xxx   just clean */
+bootenv: (
+    {
+        name = "swu_param";
+        value = "";
+    },
+    {
+        name = "swu_software";
+        value = "";
+    },
+    {
+        name = "swu_mode";
+        value = "";
+    },
+    {
+        name = "swu_version";
+        value = "";
+    }
+);
+
+}
diff --git a/allwinner/generic/swupdate/sw-subimgs-logo.cfg b/allwinner/generic/swupdate/sw-subimgs-logo.cfg
new file mode 100644
index 00000000..90c2db63
--- /dev/null
+++ b/allwinner/generic/swupdate/sw-subimgs-logo.cfg
@@ -0,0 +1,10 @@
+swota_file_list=(
+target/allwinner/generic/swupdate/sw-description-logo:sw-description
+out/${TARGET_BOARD}/image/boot-resource.fex:new_logo
+#out/${TARGET_BOARD}/boot_initramfs_recovery.img:recovery
+#out/${TARGET_BOARD}/uboot.img:uboot
+#out/${TARGET_BOARD}/boot0.img:boot0
+#out/${TARGET_BOARD}/boot.img:kernel
+#out/${TARGET_BOARD}/rootfs.img:rootfs
+#out/${TARGET_BOARD}/usr.img:usr
+)
--
2.29.0
```

3. 重新编译打包

4. 打包 OTA 包：swupdate_pack_swu -logo

5. 小机端升级命令：swupdate_cmd.sh -i /mnt/UDISK/tina-r818-evb1-logo.swu -e stable,upgrade_logo

11.11 UDISK 分区如何重命名

11.11.1 问题背景

硬件：t113

软件：u-boot-2018

11.11.2 问题描述

UDISK 分区如何重命名

11.11.3 问题分析

- UDISK 分区是 SUNXI 平台的最后一个分区，当 FLASH 充足，给所用分区都分配好了大小，剩余的 FLASH 空间默认分配给 UDISK 分区。
- 满足烧录器烧录，卡烧录和 USB 烧录的要求

11.11.4 解决方法

USB 烧录和卡烧录方案：

在 SDK 的/brandy/brandy-2.0/u-boot-2018/configs/xxxx_defconfig 中新增 CONFIG_LAST_PARTITION_NAME=“last_partition_name” 配置项进行命名，last_partition_name 为最后一个分区的分区名，参考补丁如下：

```
diff --git a/configs/sun8iw20p1_auto_nand_defconfig b/configs/sun8iw20p1_auto_nand_defconfig
index 35c57f0..9ffd3f7 100644
--- a/configs/sun8iw20p1_auto_nand_defconfig
+++ b/configs/sun8iw20p1_auto_nand_defconfig
@@ -209,3 +209,5 @@
 # CONFIG_CONFIG_LCD_CHECK_SKIP_OPEN is not set
 # CONFIG_BOOT_GUI_DOUBLE_BUF is not set
 # CONFIG_BOOT_GUI_TEST is not set
+
+CONFIG_LAST_PARTITION_NAME="last_partition_name"
```

烧录器方案：

1. 修改 update_mbr 的工具，源码需通过 FAE 提供
2. 修改 #define LAST_PARTITION “aw_last_partition”
3. 重新编译生成 update_mbr

4. 重新打包后生成的 GPT 分区表和 MBR 分区表的最后一个表项即为 aw_last_partition

参考补丁如下

```
---
 pack_tools/create_mbr/update_mbr.c | 5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)

diff --git a/pack_tools/create_mbr/update_mbr.c b/pack_tools/create_mbr/update_mbr.c
index 5e58774..2851805 100644
--- a/pack_tools/create_mbr/update_mbr.c
+++ b/pack_tools/create_mbr/update_mbr.c
@@ -21,6 +21,7 @@
 #define ERR(fmt, arg...) printf("ERROR: "fmt, ##arg)

 #define  MAX_PATH  260
+#define  LAST_PARTITION  "aw_last_partition"

 int IsFullName(const char *FilePath);
 __s32  update_mbr_crc(sunxi_mbr_t *mbr_info, __s32 mbr_count, FILE *mbr_file);
@@ -416,7 +417,7 @@ __s32 update_for_part_info(sunxi_mbr_t *mbr_info, sunxi_download_info *dl_map, _
     if(!script_parser_fetch_mainkey_sub("name", part_handle, value))
     {
        DBG("disk name=%s\n", (char *)value);
-       if(!strcmp((char *)value, "UDISK"))
+       if(!strcmp((char *)value, LAST_PARTITION))
        {
          udisk_exist = 1;
          is_udisk    = 1;
@@ -605,7 +606,7 @@ __s32 update_for_part_info(sunxi_mbr_t *mbr_info, sunxi_download_info *dl_map, _
     mbr_info->array[part_index].addrhi = (__u32)((start_sector >>32) & 0xffffffff);

     strcpy((char *)mbr_info->array[part_index].classname, "DISK");
-    strcpy((char *)mbr_info->array[part_index].name, "UDISK");
+    strcpy((char *)mbr_info->array[part_index].name, LAST_PARTITION);

     mbr_info->array[part_index].sig_erase = 0x8000;

--
```

11.12 OPTEE 是否是必须的?

11.12.1 问题背景

硬件：t113

软件：u-boot-2018

11.12.2 问题描述

分析 optee 系统模块是否有保留的必要性?

11.12.3 问题解答

optee 是必须的，原因如下：

- 安全系统的初始化和安全密钥的烧写
- 实现 ARM 标准的 PSCI 接口，例如 CPU on/off，suspend/resume

著作权声明

商标声明

免责声明