Linux U-Boot
开发指南

版本号: 3.0
发布日期: 2021.05.24

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.7.19	xxx	添加基础模板
1.1	2020.7.21	xxx	添加快速编译 boot0 及 U-Boot
2.0	2020.11.10	AWA1538	1. 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	1. 增加 LICHEE 配置宏信息

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 brandy2.0, 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 LICHEE 类宏关键字解释

请到 longan 目录下的.buildconfig 查看目前使用了以下 LICHEE 类宏。

```
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
```

3 编译方法介绍

3.1 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
```

3.2 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
```

3.3 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
```

3.4 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
```

4 U-Boot 功能及其配置方法/文件介绍

4.1 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

4.2 U-Boot 功能配置方法介绍

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

4.2.1 通过 defconfig 方式配置

1. `vim /longan/brandy/brandy-2.0/u-boot-2018/configs/{LICHEE_CHIP}_defconfig`
2. 打开`{LICHEE_CHIP}_defconfig`或`{LICHEE_CHIP}_nor_defconfig`后，在相应的宏定义前去掉或添加"#"即可将相应功能开启或关闭。如下图，只要将`CONFIG_SUNXI_NAND`前的`#`去掉即可支持 NAND 相关功能，其他宏定义的开启关闭也类似。修改后需要运行`make xxx_defconfig`使修改后的配置生效。

```
# flash
CONFIG_SUNXI_SDMMC=y
CONFIG_MMC=y
CONFIG_SUNXI_FLASH=y
# CONFIG_SUNXI_NAND=y

#nsi
CONFIG_SUNXI_NSI=y

#usb otg config
CONFIG_SUNXI_USB=y
CONFIG_SUNXI_EFEX=y
CONFIG_SUNXI_BURN=y

#partition
CONFIG_EFI_PARTITION=y

#image
CONFIG_ANDROID_BOOT_IMAGE=y

#sprite
CONFIG_SUNXI_SPRITE=y
CONFIG_SUNXI_SECURE_STORAGE=y
CONFIG_SUNXI_SPRITE_CARTOON=y
```

图 4-1: defconfig 配置图

4.2.2 通过 menuconfig 方式配置

通过 menuconfig 方式配置的方法步骤如下：

1. `cd /longan/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。

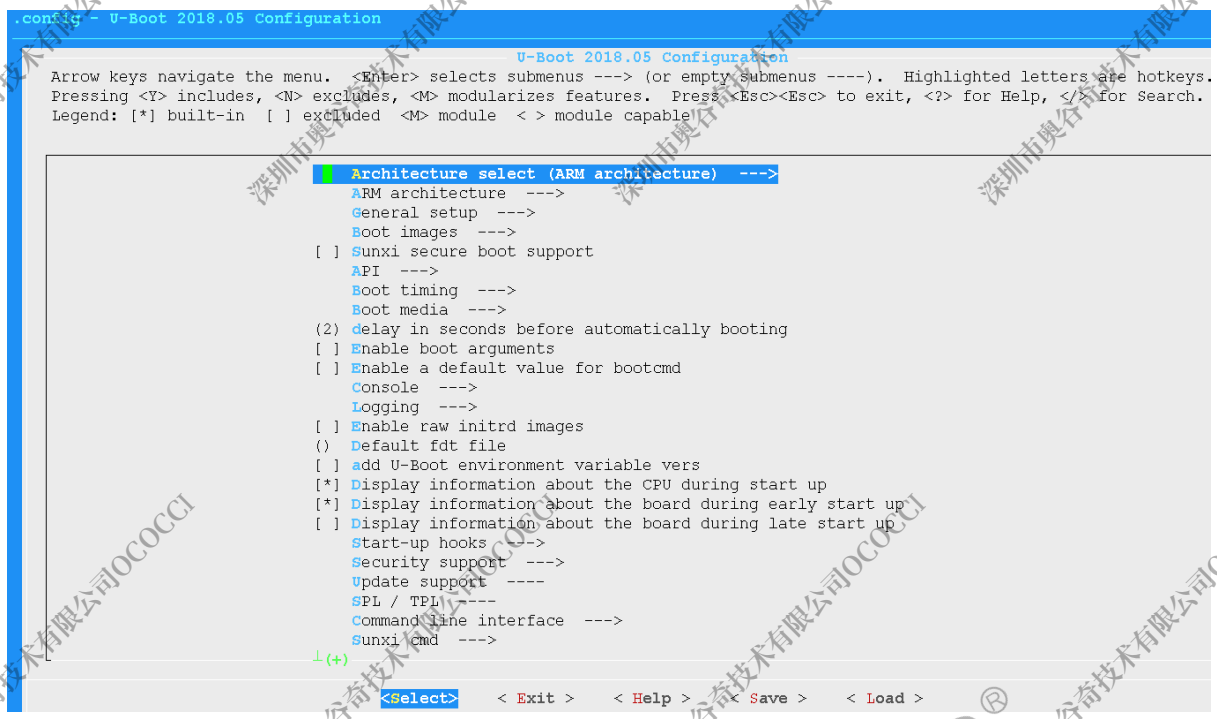

图 4-2: menuconfig 配置菜单图

4.3 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。

4.3.1 U-Boot-dts 路径

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

4.3.2 U-Boot-dts，defconfig 配置

配置项	配置项含义
CONFIG_OF_SEPARATE	构建 U-Boot 设备树成为 U-Boot 的一部分
CONFIG_OF_BOARD	关闭使用外部 dts
CONFIG_DEFAULT_DEVICE_TREE	选择构建的 dts 文件文件名
CONFIG_SUNXI_NECESSARY_REPLACE_FDT	开启选项，实现内部 dts 换成外部 dts

配置项	选项
CONFIG_OF_SEPARATE	y
CONFIG_OF_BOARD	n
CONFIG_DEFAULT_DEVICE_TREE	“{LICHEE_CHIP}-soc-system”
CONFIG_SUNXI_NECESSARY_REPLACE_FDT	y

4.3.3 U-Boot-dts 注意事项

4.3.3.1 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)_d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}，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
```

4.3.3.2 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。

4.3.3.3 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。

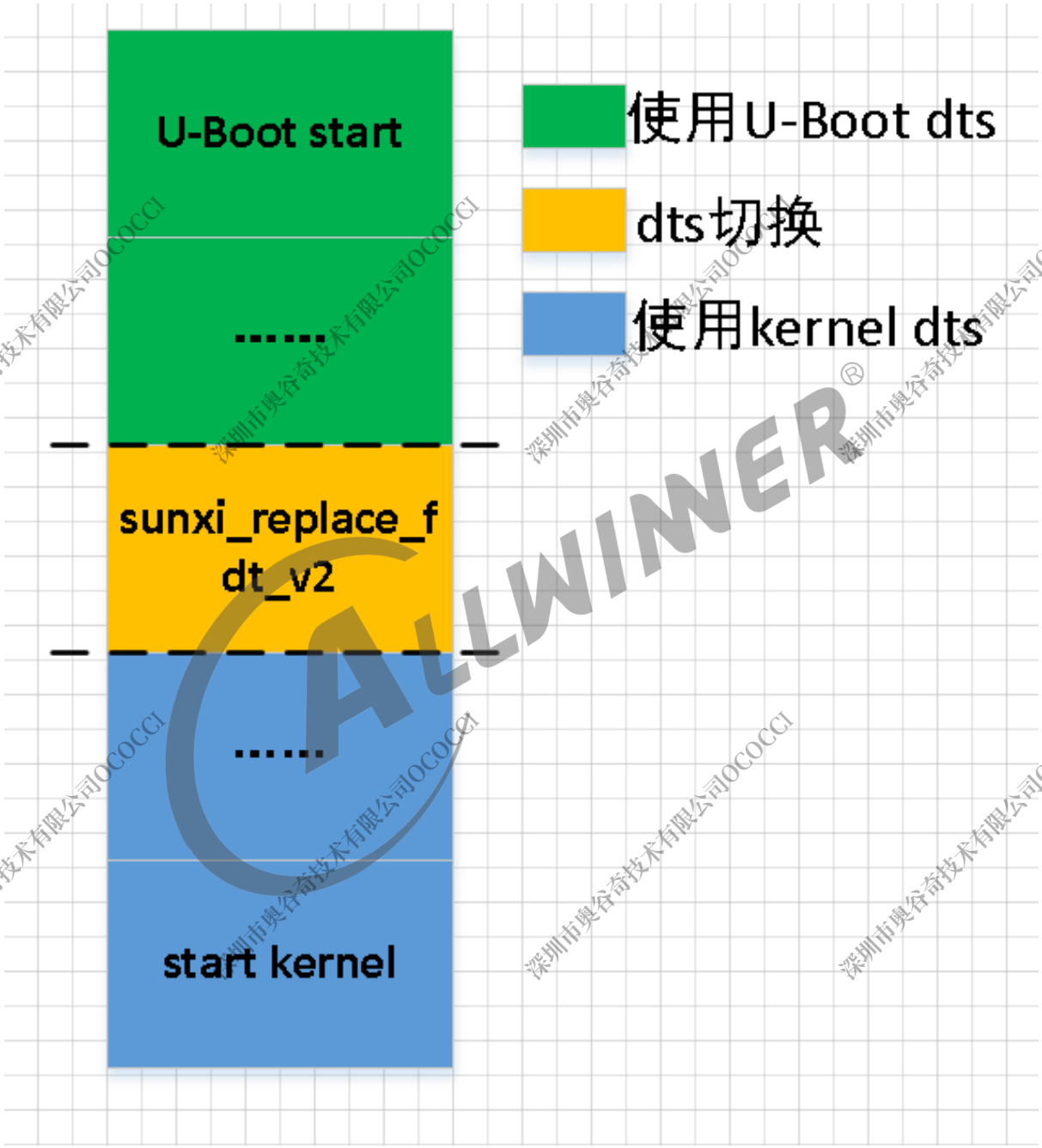

图 4-3: dts 变化图

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
=>
```

5 U-Boot常用命令介绍

5.1 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。

5.2 sunxi_flash read 命令说明

5.2.1 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
```

5.2.2 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。
```

5.3 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。

5.3.1 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。

5.3.2 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；

5.3.3 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
```

5.4 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 5-1: 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：

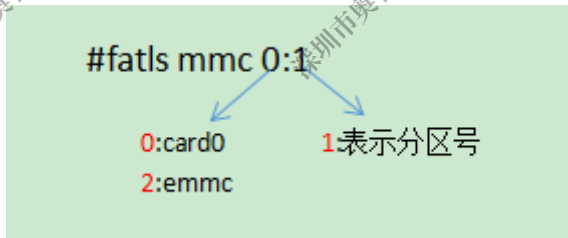

图 5-2: fatls 命令参数说明图

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 5-3: 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。

5.5 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
```

5.6 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。

5.6.1 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 0
    x00000000>;
        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>]
```

5.6.2 修改配置

5.6.2.1 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>`

5.6.2.2 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"`

5.6.3 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<功能分配><内部电阻状态><驱动能力><输出电平状态>`

5.6.3.1 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`

5.6.3.2 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

属性字段	含义
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#
```

5.6.3.3 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。

5.6.3.4 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 0
    xffffffff>;
};
```

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 0
    x00000001 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 0
    x00000001>;   //修改ok
};
sunxi#
```

5.7 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。

6 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 {LICHEE_BOARD_CONFIG_DIR}/sys_config.fex 文件，在主键 [platform] 下添加子键"debug_mode = 8"

   "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 {LICHEE_BOARD_CONFIG_DIR}/b3/uboot-board.dts 文件，在 platform 节点下添加子键"debug_mode = 8"

   "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。

7 进入烧写的方法

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`

8 常用接口函数

8.1 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`

8.2 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)`。

8.3 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)。

8.4 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

8.5 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`

8.6 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
```

9 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

著作权声明

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明


、 
、全志科技、  （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。