Android 10 UBoot 开发说明书

V1.0

2020.03.05

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2020.03.05		建立初版

目录

1. 前言

1.1 编写目的

介绍 uboot 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 UBoot-2018 平台。

1.3 相关人员

uboot 开发/维护人员，内核开发人员。

2. 编译打包和烧写

2.1 编译方法介绍

2.1.1 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
```

2.1.2 编译 uboot

cd longan/brandy/brandy-2.0/u-boot-2018/进入 u-boot-2018 目录
以 sun50iw10p1 为例，依次执行如下操作即可。

```
1）make sun50iw10p1_defconfig
2）make -j
```

2.1.3 编译 boot0/fes/sboot

cd longan/brandy/brandy-2.0/spl 进入 spl 目录，需设置平台和要编译的模块参数。
以 sun50iw10p1 为例, 编译 nand/emmc 的方法如下：

1）编译 boot0

```
make distclean
make p=sun50iw10p1 m=nand
make boot0
```

```
make distclean
make p=sun50iw10p1 m=emmc
make boot0
```

2）编译 fes

```
make distclean
make p=sun50iw10p1 m=fes
make fes
```

3）编译 sboot

```
make distclean
make p=sun50iw10p1 m=sboot
make sboot
```

2.1.4 快速编译 boot0 及 uboot

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，sun50iw10p1/sun8iw18p1/等等。

```
./build.sh -p sun50iw10p1
```

3. uboot 功能及其配置方法介绍

3.1 uboot 功能介绍

在嵌入式操作系统中，BootLoader/UBOOT 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。

UBOOT 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

3.2 uboot 功能配置方法介绍

uboot 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:

3.2.1 通过 defconfig 方式配置

1）vim /longan/brandy/brandy-2.0/u-boot-2018/configs/sun50iw10p1_defconfig

2) 打开 sun50iw10p1_defconfig 或 sun50iw10p1_nor_defconfig 后, 在相应的宏定义前去掉或添加"#" 即可将相应功能关闭或开启, 示例如下图:

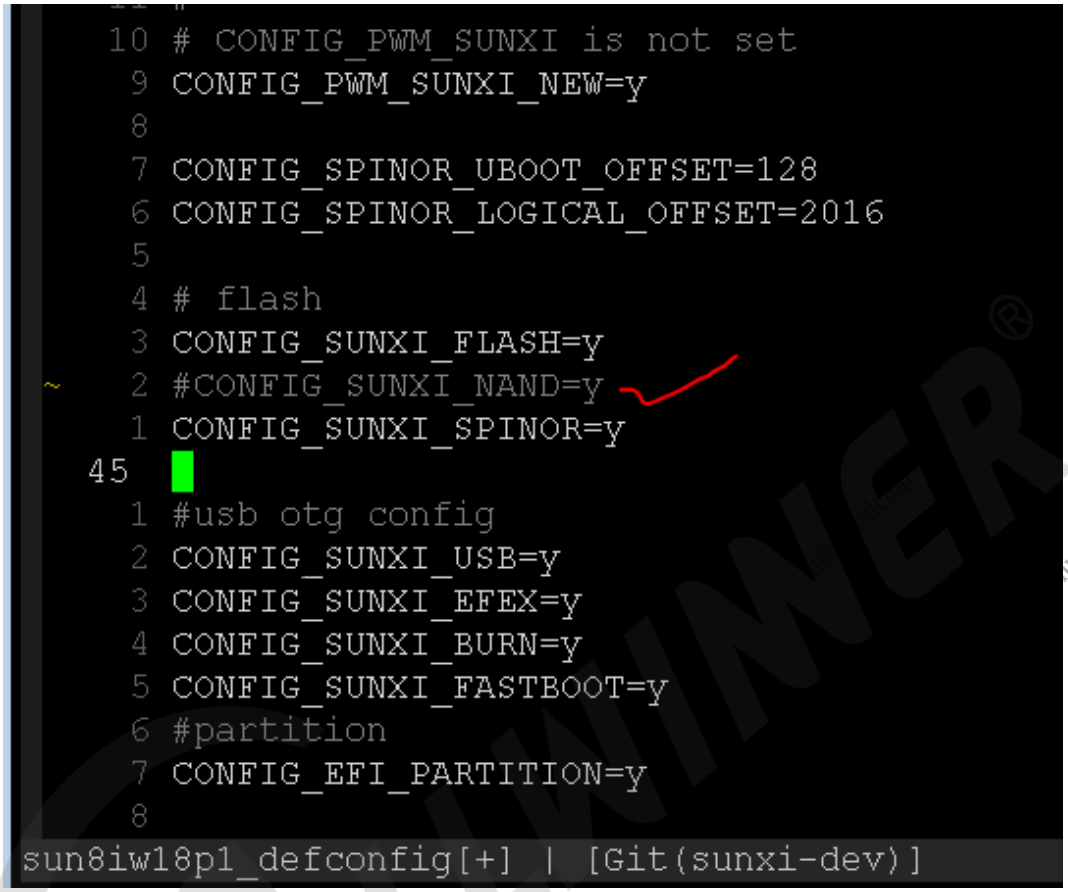

图 1: defconfig 配置图

如上图, 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

3.2.2 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）make menuconfig

执行上述命令后，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，通过这种方式修改的配置会在运行 make xxx_defconfig 后被 xxx_defconfig 中的配置覆盖。

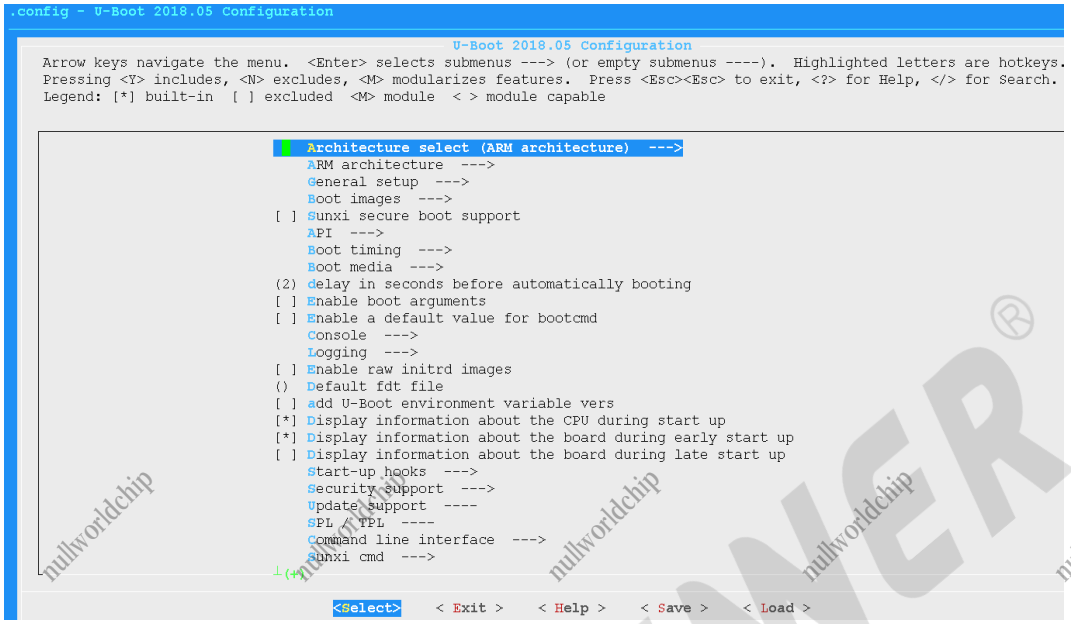

图 2: menuconfig 配置菜单图

4. U-Boot 常用命令介绍

4.1 env 命令说明

通过 env 命令可以对 longan/devices/configs/chips/h3/configs/default/env.cfg 中的环境变量进行查看及更改，让小机上电启动进入 uboot shell 命令状态，输入命令"env" 即可查看命令帮助信息。

env 命令使用示例：

1）输入命令"env print"，可查看当前所有的环境变量信息，如下：

```
=> env print
boot_fastboot=fastboot
boot_normal=sunxi_flash read 40007800 boot;bootm 40007800
boot_recovery=sunxi_flash read 40007800 recovery;bootm 40007800
bootcmd=run setargs_nand boot_normal
bootdelay=0
cma=320M
console=ttyS0,115200
earlyprintk=sunxi-uart,0x01c28000
fdtcontroladdr=7bee6f40
fileaddr=40000000
filesize=384038
init=/init
initcall_debug=0
keybox_list=hdcpkey
loglevel=8
mmc_root=/dev/mmcblk0p4
nand_root=/dev/nand0p4
partitions=bootloader@nand0p1:env@nand0p2:boot@nand0p3:super@nand0p4:misc@nand0p5:recovery@nand0p6:cache@nand0p7:vbmeta@nand0p8:
vbmeta_system@nand0p9:vbmeta_vendor@nand0p10:metadata@nand0p11:private@nand0p12:frp@nand0p13:empty@nand0p14:media_data@nand0p15:
Reserve0@nand0p16:UDISK@nand0p17
setargs_mmc=setenv bootargs earlyprintk=${earlyprintk} initcall_debug=${initcall_debug} console=${console} loglevel=${loglevel}
        root=${mmc_root}  init=${init} partitions=${partitions} cma=${cma} mac_addr=${mac} wifi_mac=${wifi_mac} bt_mac=${bt_mac} gpt=1
setargs_nand=setenv bootargs earlyprintk=${earlyprintk} initcall_debug=${initcall_debug} console=${console} loglevel=${loglevel}
        root=${nand_root} init=${init} partitions=${partitions} cma=${cma} mac_addr=${mac} wifi_mac=${wifi_mac} bt_mac=${bt_mac} gpt=1
snum=548783d025244000061

Environment size: 1291/131068 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。

4.2 sunxi_flash read 读命令说明

1）sunxi_flash reade命令的使用方法：
sunxi_flash read dram_addr flash_addr — 将flash指定地址中数据读到DRAM的指定地址处
2）sunxi_flash read命令的使用示例：
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。

4.3 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。

4.3.1 使用前提

使用 fastboot 前需安装 fastboot 相关驱动，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)。Fastboot 的 window 驱动安装是个问题 (Linux 下不需要安装驱动)，因为 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。

4.3.2 使用步骤

1）小机上电启动，按任意键进入 uboot 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（注：/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；

4.3.3 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 earse：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。

fastboot boot 将一个内核下载到内存中启动，例如 fastboot boot /path/to/uImage，将 uImage 下载到内存中启动。

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 uboot 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
```

4.4 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 3: fatls 命令执行示例图

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：

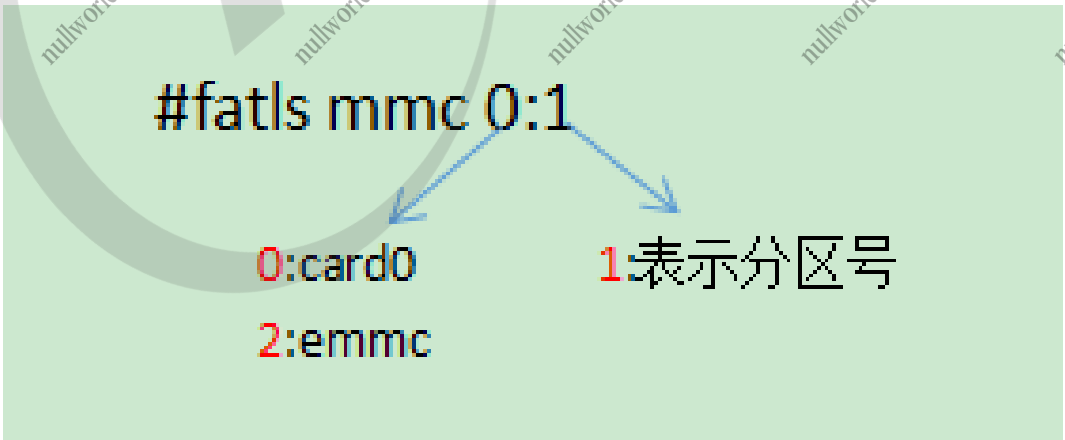

图 4: fatls 命令参数说明图

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: 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/
adask鲔
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。

4.5 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
```

4.6 FDT 命令说明

FDT：flattened device tree 的缩写在 UBOOT 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#
```

注：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

4.6.1 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 命令查询。

4.6.1.1 第一步：在根目录下查找

```
sunxi#fdt list  /
/ {
        model = "sun50iw1p1";
        compatible = "arm,sun50iw1p1", "arm,sun50iw1p1";
        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>;
};
```

4.6.1.2 第二步：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>;
};
```

4.6.1.3 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，别名可用 path 字段。

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

4.6.2 修改配置

4.6.2.1 修改整数配置

命令格式：fdt set path prop 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>

4.6.2.2 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"

4.6.3 GPIO 或者 PIN 配置特殊说明

4.6.3.1 port 接口对应的数字编号说明

```
#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
```

4.6.3.2 Sysconfig 中描述 gpio 的形式

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

4.6.3.3 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

4.6.3.4 查看 PIN 配置

PIN 配置属性字段说明:

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，allwinner,pull，allwinner,drive，allwinner,data 这些属性分别表示, 功能分配, 内部电阻状态, 驱动能力, 输出电平状态，其中值为 0xffffffff 表示使用默认值。

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#
```

查看 CPUS 的 PIN 配置查看 s_uart0 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#
```

4.6.3.5 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>;
};
```

注意：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 muxsel pull data 的值也会产生类似效果。

4.6.3.6 GPIO 配置说明

Device tree 和 sysconfig.fex 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>;
};
```

usb_id_gpio = port:PH09<0><1>

对应于 device tree 中 usb_id_gpio = <0x00000030 0x00000007 0x00000009 0x00000000 0x00000001 0xffffffff 0xffffffff> 由 5.1 节描述，端口 PH 组内序号功能分配内部电阻状态驱动能力输出电平其中首个 0x00000030 是 device tree 内部一个节点相关信息，这里可以略过。

修改 GPIO 配置如果需要修改 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#
```

4.7 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

注：其他更多 uboot 命令介绍，请进入 uboot shell 命令状态后输入"help" 进行了解。

5. 基本调试方法介绍

5.1 debug 调试信息介绍

1）debug_mode

debug_mode 可以控制 uboot 的打印等级，打开 longan/devices/configs/chips/h3/configs/p1/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 uboot 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 uboot 并打包烧录即可。

6. 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 uboot 控制台输入 efex
4. 进入 android 控制台输入 reboot efex

7. 常用接口函数

7.1 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: 失败。如果此时 lenp 非空，则是失败代码

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: 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

uboot 中 fdt 命令行的实现：cmd/fdt.c

7.2 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

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);

7.3 调用 uboot 命令行

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

- 作用：执行 uboot 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 uboot 的自动启动命令

s = env_get(”bootcmd”);

run_command_list(s, -1, 0);

7.4 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

7.5 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

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：
  - 偏移大小

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
  - -1: 失败

6. 应用参考

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

7.6 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

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：sprite/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

8. 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 uboot 运行开始即可访问
- malloc 在重定位后才能访问

9. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner. The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This document neither states nor implies warranty of any kind, including fitness for any particular application.tates nor implies warranty of any kind, including fitness for any particular application.