Linux_MMC_NOR 离线烧录 _ 开发指南

开发指南

版本号: 1.4

发布日期: 2022-07-2

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021-10-13	AWA0332	1.Init from H616_Android10 EMMC 烧录器固件制作及验证
1.1	2021-11-23	AWA0332	更新描述
1.2	2021-11-29	AWA0332	更新标题格式和错别字
1.3	2021-12-1	AWA0332	转成 markdown 格式
1.3	2021-12-1	AWA0332	修复错误，增加说明
1.4	2022-07-2	AWA1832	集成 NOR 方案

目　录

插　图

1 前言

1.1 文档简介

介绍 sunxi 平台 SD/eMMC，NOR 离线烧录的使用方法

1.2 目标读者

sunxi 平台 SD/MMC/NOR 驱动的开发/维护人员

1.3 适用范围

产品名称
A133
h616
v853
R853
T507

1.4 模块功能介绍

该文档提供如何生成离线烧录的 img，该 img 提供给烧录器厂家进行进行离线烧录

1.5 相关术语介绍

1.5.1 硬件术语

离线烧录：

通常指“裸片烧录”，芯片在未贴板之前，搭配相应的适配座放在编程器上进行烧录，与之对应的

是在线烧录

1.5.2 软件术语

无

2 模块使用范例

2.1 烧录器固件包生成方法

说明：烧录器固件是专门生成的一个固件，生成这个固件后，可以直接交给烧录器厂家，用于离线烧录。生成方法如下：pack -w，生成固件 xxx_programmer.img，pack -w 的实现相关补丁如下，最新版 logan 已经集成此部分功能：

```
diff --git a/vendorsetup.sh b/vendorsetup.sh
index 9eb0126..3203880 100644
--- a/vendorsetup.sh
+++ b/vendorsetup.sh
@@ -244,12 +244,13 @@ function package()
     local debug=uart0
     local sigmode=none
     local securemode=none
+    local programmermode=none
     local packpath=$lichee_top_path/tools/pack

     [ "x$build_system" == "xlongan" ] && \
     packpath=$lichee_top_path/build

-    while getopts "i:c:p:b:dsvh" arg
+    while getopts "i:c:p:b:dsvhw" arg
     do
         case $arg in
             i)
@@ -277,6 +278,9 @@ function package()
                 package_usage
                 return 0
                 ;;
+            w)
+                programmermode=programmer
+                ;;
             ?)
                 return 1
                 ;;
@@ -285,7 +289,7 @@ function package()

     (
     cd $packpath
-    ./pack -i $ic -c $chip -p $platform -b $board -d $debug -s $sigmode -v $securemode --
    platform_version ${platform_version}
+    ./pack -i $ic -c $chip -p $platform -b $board -d $debug -s $sigmode -v $securemode -w
    $programmermode --platform_version ${platform_version}
     )
```

vendorsetup.sh: 文件路径 android/device/softwinner/common/vendorsetup.sh

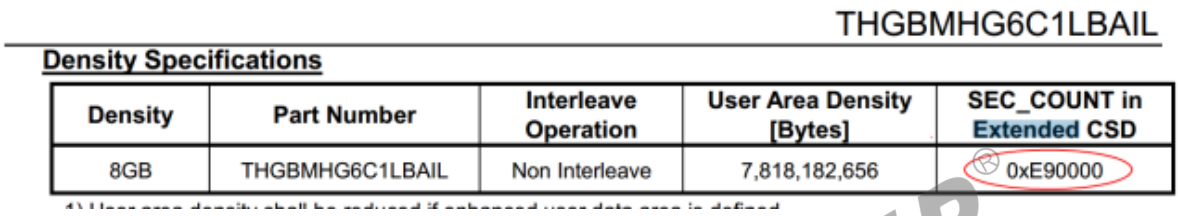
需将 Android 的 super 分区（sparse 格式）转换为裸数据格式，pack 脚本修改，最新版 logan 已经集成此部分功能：

```
diff --git a/pack b/pack
index a518ff8..9508bb7 100755
--- a/pack
+++ b/pack
@@ -589,13 +589,21 @@ function img_to_programmer()
        local out_img=$1
        local in_img=$2

+       cd ${LICHEE_PACK_OUT_DIR}/
+
        if [ "x${PACK_SIG}" = "xprev_refurbish" -o "x${PACK_SIG}" = "xsecure" ]; then
                programmer_img toc0.fex toc1.fex ${out_img} > /dev/null
        else
                programmer_img boot0_sdcard.fex boot_package.fex ${out_img} > /dev/null
        fi
-       #create_img toc0.fex toc1.fex
-       programmer_img sys_partition.bin sunxi_mbr.fex ${out_img} ${in_img} > /dev/null
+
+       if [ -f sunxi_gpt.fex ] ; then
+               #create_img for sunxi_gpt.fex
+               programmer_img sys_partition.bin sunxi_mbr.fex ${out_img} ${in_img}
    sunxi_gpt.fex > /dev/null
+       else
+               #create_img for sunxi_mbr.fex
+               programmer_img sys_partition.bin sunxi_mbr.fex ${out_img} ${in_img} > /dev/
    null
+       fi
 }

 function do_ini_to_dts()
@@ -879,6 +887,17 @@ function do_finish()
                        pack_error "update_mbr failed"
                        exit 1
                fi
+
+        #15269888 is from emmc datasheet, sec_count in Extended CSD is 0xe90000 =
    15269888.
+                # 0x1CE8000 for ISOCOM MEMA016G, 0x1CE8000 = 30310400
+               #20 is the mbr location for emmc
+               #0  is emmc/nand ; 1 is spinor
+               #update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 30310400 20 0
+               update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 15269888 20 0
+               if [ $? -ne 0 ]; then
+                       pack_error "update_mbr_gpt failed"
+                       exit 1
+               fi
        fi

        dragon image.cfg    sys_partition.fex
@@ -1229,7 +1248,13 @@ function do_pack_android()
        for fex_name in ${fex_list}; do
                img_name=${fex_name%\.fex}.img
                if [ -f ${ANDROID_IMAGE_OUT}/${img_name} ]; then
-                       ln -sf ${link_real}/${img_name} ${fex_name}
+                       if [ "x${PACK_PROGRAMMER}" = "xprogrammer" -a "x${fex_name%\.fex}"
    = "xsuper" ]; then
+                               simg2img ${link_real}/${img_name} ${link_real}/${img_name}.
```

```
    tempbin
+                               img_name=${img_name}.tempbin
+                               ln -sf ${link_real}/${img_name} ${fex_name}
+                       else
+                               ln -sf ${link_real}/${img_name} ${fex_name}
+                       fi
                        echo "link ${img_name} -> ${fex_name}"
                fi
        done
```

pack 脚本路径 android/longan/build/pack

注意，pack 脚本中的 sec_count in Extended CSD 需要根据实际 emmc 的 datasheet 填写，如下图所示。

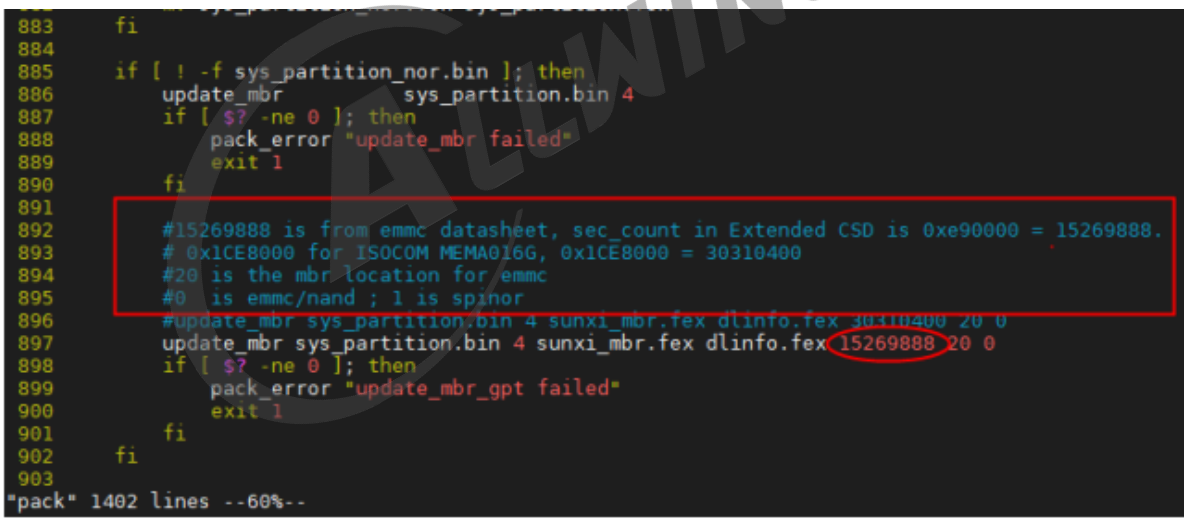


THGBMHG6C1LBAIL

Density Specifications

Density	Part Number	Interleave Operation	User Area Density [Bytes]	SEC_COUNT in Extended CSD
8GB	THGBMHG6C1LBAIL	Non Interleave	7,818,182,656	0xE90000

图 2-1: image-20211201145716795



```
883     fi
884
885     if [ ! -f sys_partition_nor.bin ]; then
886         update_mbr          sys_partition.bin 4
887         if [ $? -ne 0 ]; then
888             pack_error "update_mbr failed"
889             exit 1
890         fi
891
892         #15269888 is from emmc datasheet, sec_count in Extended CSD is 0xe90000 = 15269888.
893         # 0x1CE8000 for ISOCOM MEMA016G, 0x1CE8000 = 30310400
894         #20 is the mbr location for emmc
895         #0  is emmc/nand ; 1 is spinor
896         #update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 30310400 20 0
897         update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 15269888 20 0
898         if [ $? -ne 0 ]; then
899             pack_error "update_mbr_gpt failed"
900             exit 1
901         fi
902     fi
903
"pack" 1402 lines --60%--
```

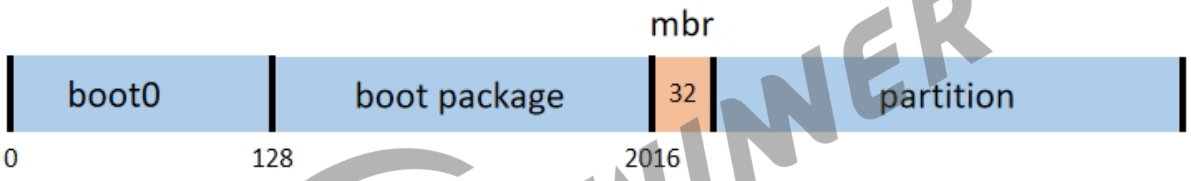
图 2-2: image-20211201145744611

3 SPI NOR 存储分布

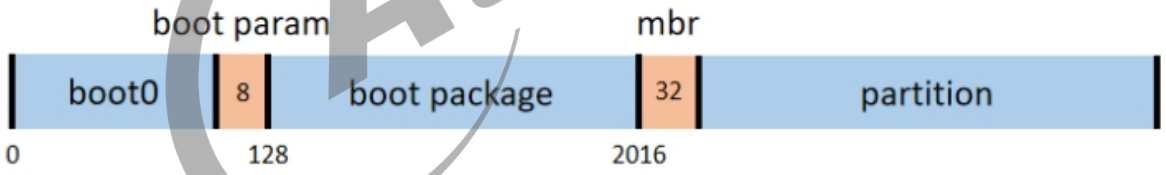
这里以这里以 sun8iw21p1 为例。

3.1 存储分布

每个 sectors 大小为 512byte 旧的方案，BOOT 启动或烧写阶段的时候都会往 boot0/TOC0 的头部更新参数（flash/axp/dram/drm），这种方案虽然方便存储启动参数，但是也带来一些问题：1. 重复更新 BOOT0 或 TOC0，万一更新出错，带来了无法启动风险，比如 T7 方案。2. 影响安全方案的 TOC0 签名规范，即对配置参数没有进行签名认证。旧布局：



新布局：SPINOR flash 新布局如下，默认 uboot2018 最新代码都是这个布局，较就布局，新布局会多了一个 boot param 分区，主要用于存放 flash/axp/dram 等参数



新布局的 4k 空间是拿 boot0 的空间划分出来的，这样 boot0 的大小要小于 60k（120 扇区），客户案如果有烧录器方案的，需要注意这个点，目前针对这个分区在离线烧录时，第一次运行系统时 uboot 会检测 boot param 分区是否被烧写，如果没被烧写则根据 fdt 的内容填充，在针对没有 uboot 的快启方案，此部分工作由 kernel 完成

3.2 存储配置的修改

修改的原则：不能把其他分区的数据占了。比如 boot0、bootpackage、mbr 的大小不能超过分配的大小 Uboot 偏移的配置

SPL 部分:spl/board/sun8iw19p1/spinor.mk, 修改 CFG_SPINOR_UBOOT_OFFSET

图 3-1: image-boot0spinor.png

uboot 部分: 在路经u-boot-2018/configs/sun8iw21p1_nor_defconfig下，修改CONFIG_SPINOR_UBOOT_OFFSET

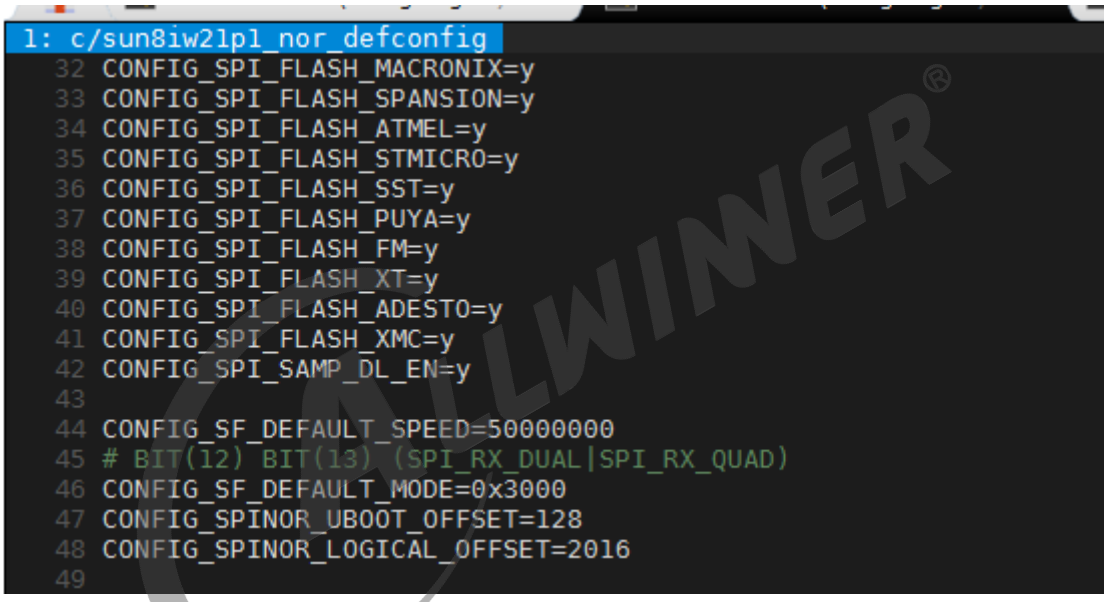


图 3-2: image-ubootspinor.png

MBR 偏移的配置 uboot 部分: 在路经u-boot-2018/configs/sun8iw21p1_nor_defconfig下，修改CONFIG_SPINOR_LOGICAL_OFFSET 其他分区的配置，详细请看 device/config/chips/v853/configs/default/sys_partition_nor.fex

3.3 存储配置的优化

在 boot_package.cfg，可查看 boot_package.fex 包含的文件。

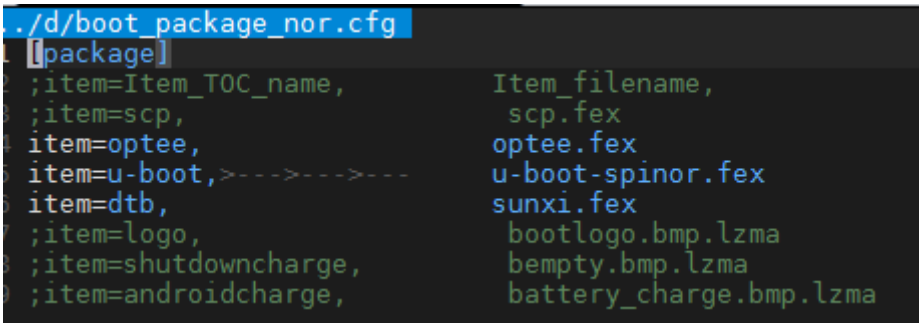


图 3-3: image-norbootpackage.png

1. 修改 uboot 偏移：

   - 查看`boot_package.fex`文件大小，便可知道从 128 扇区后，我们使用多少空间，
   - 可优化大小 =`2016 - (128 + 使用空间[1504])`
   - 修改值 = `(2016 - 可优化大小 + 64(预留空间))`
   - 将对应的 `spinor.mk`、`sun8iw21p1_nor_defconfig`中`CONFIG_SPINOR_LOGICAL_OFFSET`修改为修改值

2. 修改内核偏移：

   - 修改`linux4.9/drivers/mtd/sunxipart.c`中的`MBR_OFFSET`宏修改为`((修改值+32)/2 - 16)*1024`
   - 若通过 cmdlink 传参，需要修改 `linux4.9/arch/arm/boot/dts/sun8iw21p1.dtsi`中 bootargs 中的 mbr_offset 值为`((修改值+32)/2 - 16)*1024`

4 烧录器固件生成方法

4.1 GPT 方案

由于分区表中最后一个 udisk 分区的大小，就是其他分区用完所剩余的 flash 空间的大小，因此需要根据实际 EMMC flash 的大小，生成合适的分区表。

4.1.1 如何查看 EMMC 的 USR 分区的大小

查看该 emmc 的 data_sheet

7.4 Extended CSD Register

The Extended CSD register defines the eMMC properties and selected modes. It is 512 bytes long.
The most significant 320 bytes are the Properties segment, which defines the eMMC capabilities and cannot be modified by the host. The lower 192 bytes are the Modes segment, which defines the configuration the eMMC is working in. These modes can be changed by the host by means of the SWITCH command.

R: Read only
W: One time programmable and not readable.
R/W: One time programmable and readable.
W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and not readable.
R/W/E: Multiple writable with value kept after power failure, H/W reset assertion and any CMD0 reset and readable.
R/W/C_P: Writable after value cleared by power failure and HW/ rest assertion (the value not cleared by CMD0 reset) and readable.
R/W/E_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and readable.
W/E/_P: Multiple writable with value reset after power failure, H/W reset assertion and any CMD0 reset and not readable

[Table 26] Extended CSD Register

Name	Field	Size (Bytes)	Cell Type	CSD slice	CSD Value			
					8GB	16GB	32GB	64GB
Properties Segment								
Reserved[1]		6	-	[511:506]	-			
Extended Security Commands Error	EXT_SECURITY_ERR	1	R	[505]	0x00			

图 4-1: image-usrspace.png

Sleep Notification Timeout	SLEEP_NOTIFICATION_TIME	1	R	[216]	0x07			
Sector Count	SEC_COUNT	4	R	[215:212]	0xE90000	0x1D1F000	0x3A3E000	0x747C000
Secure Write Protect Information	SECURE_WP_INFO	1	R	[211]	0x01			
Minimum Write Performance for 8bit at 52MHz	MIN_PERF_W_8_52	1	R	[210]	0x00			

图 4-2: image-emmc-SectorCount.png

根据上图，可知 8G 的 EMMC 的大小为 0xE90000.

4.1.2 修改 pack 脚本

修改longan/build/pack的打包脚本中，生成sunxi_gpt.fex的工具，把” 15269888” 修改为 EMMC 实际大小。

```
#15269888 is from emmc datasheet, sec_count in Extended CSD is 0xe90000 = 15269888.
update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 15269888 20 0
if [ $? -ne 0 ]; then
        pack_error
        "update_mbr_gpt failed"
        exit 1
fi
```

4.1.3 打包生成 EMMC 烧录器固件

longan 的环境打包命令

```
#./build.sh pack_raw
```

```
Dragon execute image.cfg SUCCESS !
----------image is at----------
/home/yuxianyang/work/longan/out/a100_linux_perf1_uart0.img
waiting to ceate programmer img...
mbr_info->array[i].name=UDISK
this is not a partition key
line=152, fill_size=0
use_gpt = 1
----------programmer image is at----------
/home/yuxianyang/work/longan/out/a100_linux_perf1_uart0_programmer.img
```

4.2 Mbr 的方案

4.2.1 修改工具源码

打包工具的路径：longan/brandy/brandy-2.0/tools/pack_tools/create_paragremer_img 修改 part.h 中的宏定义 EMMC_CAPACITY，为实际的 emmc 大小 #define EMMC_CAPACITY (8*1024*1024)

4.2.2 重新生成工具

#make 编译生成 programmer_img，会自动拷贝到longan/tools/pack/pctools/linux/mod_update/programmer_img

4.2.3 打包生成 EMMC 烧录器固件

longan 的环境打包命令 `#./build.sh pack_raw`

```
Dragon execute image.cfg SUCCESS !
----------image is at----------
/home/yuxianyang/work/longan/out/a100_linux_perf1_uart0.img
waiting to ceate programmer img...
mbr_info->array[i].name=UDISK
this is not a partition key
line=152, fill_size=0
use_gpt = 1
----------programmer image is at----------
/home/yuxianyang/work/longan/out/a100_linux_perf1_uart0_programmer.img
```

4.3 NOR 方案

NOR 方案生成的文件 mbr，gpt 的方式与 mmc 一致，最新版 SDK 已集成，pack 时生成的 full_img.fex 文件即为最终的烧录器固件，用户只需根据需求修改对应的逻辑偏移即可，merge_full_img 工具见补充说明。

5 补充说明

5.1 Update_mbr 工具说明

根据分区配置文件，更新主引导目录文件 sunxi_mbr.fex ，sunxi_gpt.fex 及分区的下载文件列表 dlinfo.fex, 使用方法如下：`update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 15269888 20 0`

参数:

```
--sys_partition.bin----分区表的2进制文件
--分区表的备份个数，一般spinor为1，其他为4
--sunxi_mbr.fex------生成mbr格式分区表的文件
--dlinfo.fex-----------下载文件信息表
--15269888----------该flash的大小，比如EMMC的USR分区的大小，单位为sector
--20-----------------该flash的逻辑地址，单位MByte
--0--------------------代表flash的类型，1是spinor flash，0是非 spi nor类型
```

输出文件:

```
sunxi_mbr.fex
dlinfo.fex
sunxi_gpt.fex
sunxi_gpt_head.fex
Sunxi_gpt_pte.fex
```

5.2 programmer_img 工具说明

生成 mmc 介质的烧录固件

5.2.1 参数为 3 个

使用方法如下：`programmer_img boot0_sdcard.fex boot_package.fex ${out_img}`

参数说明:

```
--boot0_sdcard.fex----boot0文件
--boot_package.fex------即uboot文件
```

```
--${out_img}------为生成的镜像文件
```

备注：

1. 首先生成 20M 的文件，然后填充 0
2. 把 boot0 和 boot0 备份写到相应的位置
3. 把 uboot 和 uboot 备份写到相应的位置

5.2.2 参数为 6 个

使用方法如下：`programmer_img sys_partition.bin sunxi_mbr.fex ${out_img} ${in_img} sunxi_gpt.fex`
参数说明：

```
--sys_partition.bin----分区表的脚本文件
--sunxi_mbr.fex------为mbr格式的分区表文件
--${out_img}------为输入的镜像文件
--${in_img}-----为输出的镜像文件
--sunxi_gpt.fex----为gpt 格式的分区表文件
```

备注：

1. 读取分区表
2. 读取输入的镜像文件 ${out_img}
3. 写 MBR 文件到 ${out_img} 后面

5.3 merge_full_img 工具说明

指定起始逻辑地址，把 boot0,boot1,mbr 及分区文件下载列表里的文件合并成 img 固件包, 应用于小容量的 nor flash. 此时没有分区的概念使用有方法如下：

```
merge_full_img --out outfile
                --boot0 boot0.fex
                --boot1 boot1.fex
                --mbr mbr.fex
                --partition partition.fex
                --logic_start [512|256]
                --uboot_start [24|32|64]
                --help
```

参数说明：

```
--out <outfile>----指定输出目标文件
--boot0 <boot0.fex>----指定输入的 boot0 文件
```

```
--boot1 <boot1.fex>----指定输入的 boot1 文件
--mbr <mbr.fex>----指定输入的 mbr 文件
--partition <partition.fex>----指定输入的分区配置文件
--logic_start <512|256>----指定起始逻辑地址
--uboot_start [24|32|64]----bootpackage起始逻辑地址
--help----显示使用方法
```

5.4 dragonsecboot 工具说明

- 根据指定的 keys 生成 toc0 文件
- 根据指定的 keys 和 cnfbase 生成 toc1 文件
- 根据配置文件生成 keys
- 按配置文件的配置进行打包生成目的文件

使用方法如下：

```
dragonsecboot -toc0 <cfg_file> <keypath> <version_file>
dragonsecboot -toc1 <cfg_file> <keypath> <cnfbase> <version_file>
dragonsecboot -key <cfg_file> <keypath>
dragonsecboot -pack <cfg_file>
```

参数说明：

```
-toc0----表示要生成 toc0 文件
-toc1----表示要生成 toc1 文件大小
-key----表示要生成 key
-pack----表示进行打包
cfg_file----配置文件
keypath----key 的路径
cnfbase----输入的 cnf_base.cnf 文件
version_file----固件防回滚配置文件
```

5.5 signature 工具说明

对 MBR 指定要进行签名的分区文件进行签名，使用方法如下：

```
signature <sunxi_mbr_file> <dlinfo_file>
```

参数说明：

```
sunxi_mbr_file: 输入的 MBR 文件
dlinfo_file: 输入的分区下载列表文件
```

5.6 sigbootimg 工具说明

在输入文件的后面添加一个证书，并输出一个带证书的文件，使用方法如下：`sigbootimg --image <input_img> --cert <cert_file> --output <output_img>`

参数说明:

```
input_img: 输入的文件或镜像
cert_file: 文件或镜像对应的证书
output_img: 带证书的文件或镜像
```

5.7 EMMC 的存储的分布图

- Boot0_b 为 boot0 的备份，Uboot_b 为 Uboot 的备份
- toc0 存放安全 boot0
- Uboot 分区存放的实际为`bootpackage`的数据，里面一般包含 uboot 的 bin 文件，内核设备树，以及 optee 的 bin 文件，toc1 存放安全的`bootpackage`

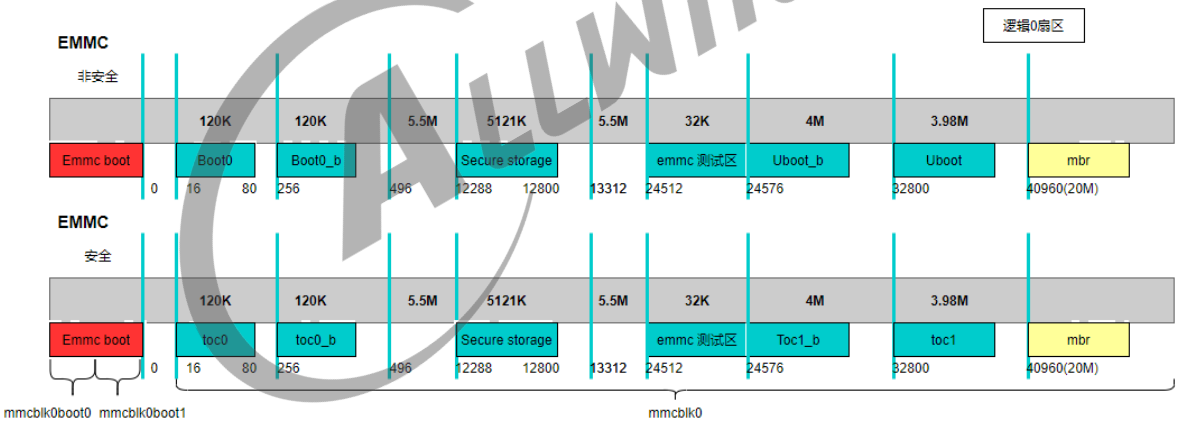


图 5-1: image-emmcpartition.png

5.8 NOR 的存储的分布图

- nor 的存储分布通过宏来控制，此配置一般在 defconfig 中，以 v853 为例，其路径为`longan/brandy/brandy-2.0/u-boot-2018/configs/sun8iw21p1_nor_defconfig`
- 在 v853 项目中安全和非安全方案的`CONFIG_SPINOR_LOGICAL_OFFSET`是不同的，需要根据`bootpackage`的大小进行动态适配

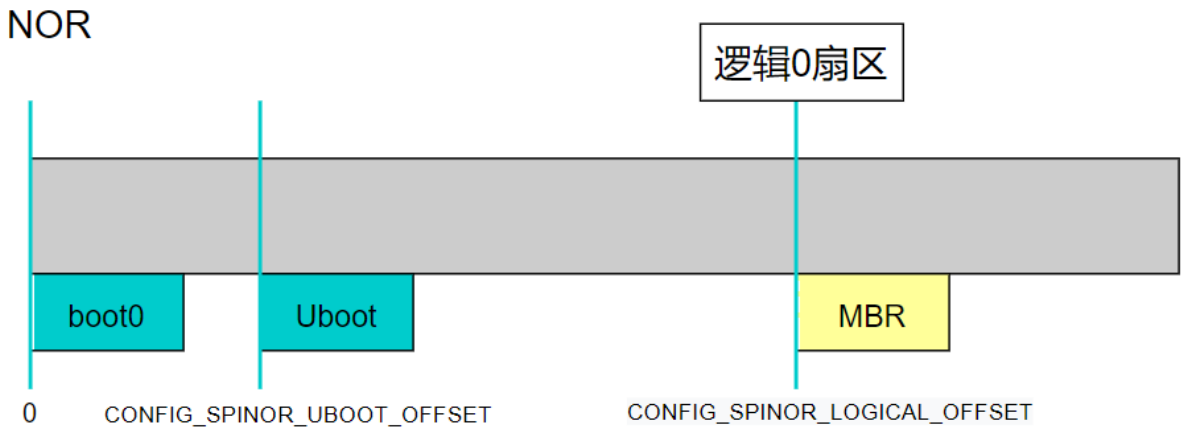


图 5-2: image-norpartition.png

5.9 Android 固件需要注意

由于 Android 的镜像是 spare 格式，需要转化为裸数据格式。因此在打包的时候需要修改如下：

```
--- a/pack
+++ b/pack
@@ -1229,7 +1229,13 @@ function do_pack_android()
        for fex_name in ${fex_list}; do
                img_name=${fex_name%\.fex}.img
                if [ -f ${ANDROID_IMAGE_OUT}/${img_name} ]; then
-                       ln -sf ${link_real}/${img_name} ${fex_name}
+                       if [ "x${PACK_PROGRAMMER}" = "xprogrammer" -a "x${fex_name%\.fex}" = "xsuper" ]; then
+                               simg2img ${link_real}/${img_name} ${link_real}/${img_name}.tempbin
+                               img_name=${img_name}.tempbin
+                               ln -sf ${link_real}/${img_name} ${fex_name}
+                       else
+                               ln -sf ${link_real}/${img_name} ${fex_name}
+                       fi
                        echo "link ${img_name} -> ${fex_name}"
                fi
        done
```

6 验证方法

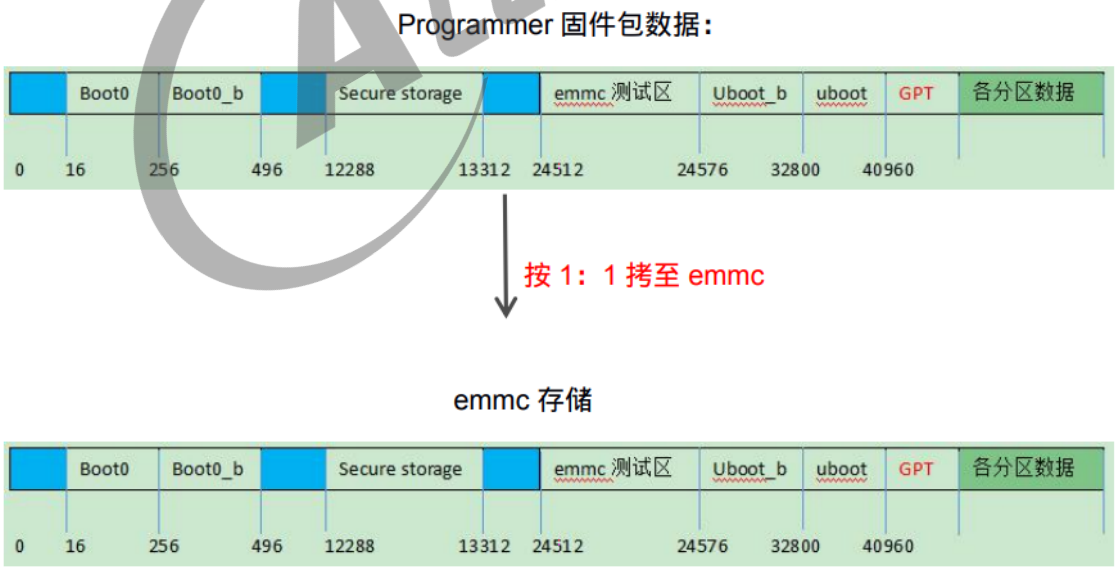
6.1 EMMC 方案自测验证方法

6.1.1 检查生成烧录器的固件

- 使用 hexdump 工具查看 boot0_sdcard.fex 是否在第 16 个扇区, 备份在第 256 个扇区？，A50 平台在 384 个扇区?
- 使用 hexdump 工具查看 boot_package.fex 是否在第 32800 个扇区，备份在第 24576 个扇区?

6.1.2 自测验证方法

只要将 programmer 固件包中的数据按 1：1 的方式拷到 emmc 中就行了，如下图。



Programmer 固件包中蓝色区域为填充区域，以 0 填充，且 boot0 等非蓝色区域其实际数据大小一般都不会超过图中分配的大小，所以剩下的空间也以 0 填充，所以在将 programmer 固件包数据拷到 emmc 存储的时候，请做一些预处理，即拷贝时如遇到为 0 的连续大片区域时，则跳过，不拷贝，这样会提高拷贝效率。

6.1.3 制作特制的卡启动固件

1. 修改 sun50iw3p1.dtsi 文件

- 把 mmc2=&sdc2 注释掉。
- 把sdc0: sdmmc@04020000{2..... }整体（包括 {} 里面的内容）移动到sdc2: sdmmc@04022000前面

2. 重新编译 linux
3. 执行命令:pack —重新打包生成卡启动固件:sun50iw3p1_android_y2_uart0.img

注意：这一步只有在验证的时候使用，正式量产时不需要这一步，sun50iw3p1.dtsi 保持未修改前的状态。

6.1.4 烧录 EMMC

1. PC 启动 adb
2. 然后输入以下命令

- mkdir /mnt/sd
- mount /dev/block/mmcblk0p1 /mnt/sd
- cd /mnt/sd
- time dd if=sun50iw3p1_android_y2_card0_programmer.img of=/dev/block/mmcblk1

3. 等待命令执行完毕，这个时间会比较长。完成后关机

6.1.5 重新开机

拔掉卡，重新开机，如果能够正常启动，说明烧录器固件是正常的。

6.2 NOR 方案自测验证方法

生成烧录器固件操作最新版 SDK 已集成

6.2.1 生成整个烧录器固件并验证

1. 生成 GPT

- ./update_mbr sys_partition_nor.bin 1 sunxi_mbr_nor.fex dlinfo.fex 65535 2016 1
- 65535 – nor flash 大小
- 2016 —逻辑偏移

2. 生成烧录器固件

- ./merge_full_img --out full_img.fex --boot0 boot0_spinor.fex --boot1 boot_package_nor.fex --mbr sunxi_gpt.fex --partition sys_partition_nor.bin --logic_start 1008 --uboot_start 64

3. 传入小机

- adb push full_img.fex ./

4. dd，然后重启

```
# dd if=full_img.fex of=/dev/mtdblock0 bs=512
# sync
# reboot
```

6.2.2 单独替换某个文件

若需要单独替换某些文件，只需修改上节第四小点的操作即可，例如单独替换 GPT 可参照以下方式

```
# dd if=sunxi_gpt.fex of=/dev/mtdblock0 bs=1024 count=16 seek=1008
# sync
# reboot
```

6.3 烧写后无法启动的调试方法

- 对比烧写进去镜像文件是否在相应的位置，详细查看对应存储介质的存储分布图
- 使用 TigerDump 工具 dump 相应的文件
- 使用 SDC0 启动到内核的命令行，使用 DD 命令进行读取 flash 中的镜像文件。

7 FAQ

无

7.1 常见问题

无

著作权声明

商标声明

免责声明