



# Linux MMC NOR 离线烧录 开发指南

版本号: 1.5

发布日期: 2023-05-26

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2021-10-13	AWA0332	初始版本
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, 修复错误, 增加说明
1.4	2022-07-2	AWA1832	集成 NOR 方案
1.5	2023-05-26	AWA1767	增加 A523 相关平台

# 目 录

<b>1 前言</b>	<b>1</b>
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
1.4 模块功能介绍	1
1.5 相关术语介绍	1
1.5.1 硬件术语	1
<b>2 模块使用范例</b>	<b>3</b>
2.1 烧录器固件包生成方法	3
<b>3 SPI NOR 存储分布</b>	<b>6</b>
3.1 存储分布	6
3.2 存储配置的修改	7
3.3 存储配置的优化	8
<b>4 烧录器固件生成方法</b>	<b>9</b>
4.1 GPT 方案	9
4.1.1 如何查看 eMMC 的 USR 分区的大小	9
4.1.2 修改 pack 脚本	10
4.1.3 打包生成 eMMC 烧录器固件	10
4.2 Mbr 的方案	10
4.2.1 修改工具源码	10
4.2.2 重新生成工具	10
4.2.3 打包生成 eMMC 烧录器固件	11
4.3 NOR 方案	11
<b>5 补充说明</b>	<b>12</b>
5.1 Update_mbr 工具说明	12
5.2 programmer_img 工具说明	12
5.2.1 参数为 3 个	12
5.2.2 参数为 6 个	13
5.3 merge_full_img 工具说明	13
5.4 dragonsecboot 工具说明	14
5.5 signature 工具说明	14
5.6 sigbootimg 工具说明	14
5.7 eMMC 的存储的分布图	15
5.8 NOR 的存储的分布图	15
5.9 Android 固件需要注意	16

<b>6 验证方法</b>	<b>17</b>
6.1 eMMC 方案自测验证方法	17
6.1.1 检查生成烧录器的固件	17
6.1.2 自测验证方法	17
6.1.3 制作特制的卡启动固件	18
6.1.4 烧录 eMMC	18
6.1.5 重新开机	18
6.2 NOR 方案自测验证方法	18
6.2.1 生成整个烧录器固件并验证	19
6.2.2 单独替换某个文件	19
6.3 烧写后无法启动的调试方法	19

## 插图

图 2-1	eMMC-datasheet 扇区大小	5
图 3-1	老版本 nor 的存储分布图	6
图 3-2	新版本 nor 的存储分布图	6
图 3-3	boot0 spinor 配置文件	7
图 3-4	uboot 配置文件	7
图 3-5	boot-package 配置文件	8
图 4-1	eMMC usrspace	9
图 4-2	eMMC SectorCount	9
图 5-1	eMMC 存储分布图	15
图 5-2	nor 存储分布图	16
图 6-1	programmer 到 flash 数据分布	17

# 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	产品名称
A133	T507&T507-H
h616	T517&T517-H
v853	R853
TV303	A523
A527	MR527
T527	AI985

## 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 可提供给烧录器厂家进行离线烧录。

## 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。



## 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 ${
+       ./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
+ #40960 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 40960 0
+ update_mbr sys_partition.bin 4 sunxi_mbr.fex dlinfo.fex 15269888 40960 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
            fi
        fi
    done
endfunction
```

```
+      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: eMMC-datasheet 扇区大小

```
if [ ! -f sys_partition_nor.bin ]; then
update_mbr sys_partition.bin 4
if [ $? -ne 0 ]; then
pack_error "update_mbr failed"
exit 1
fi

#15269888 is from emmc datasheet, sec_count in Extended CSD is 0xe90000 = 15269888.
#40960 is the mbr location for emmc
#0 is emmc/nand; 1 is spinor
update_mbr sys_partition.bin 4 sunxi_mbr.fex d1info.fex 15269888 40960 0
if [ $? -ne 0 ]; then
pack_error "update_mbr_gpt failed"
exit 1
fi
fi
```

## 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 TOC0，一旦更新出错，将带来了无法启动风险。
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：

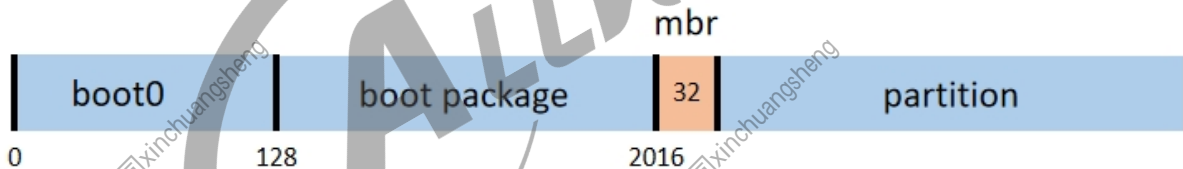


图 3-1: 老版本 nor 的存储分布图

- 新布局:SPINOR flash 新布局如下，默认从 uboot2018 开始的代码都会是这个布局，较旧布局，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 空间是从 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。

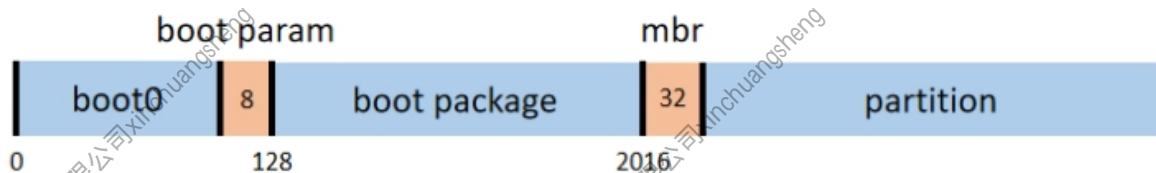


图 3-2: 新版本 nor 的存储分布图

## 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。

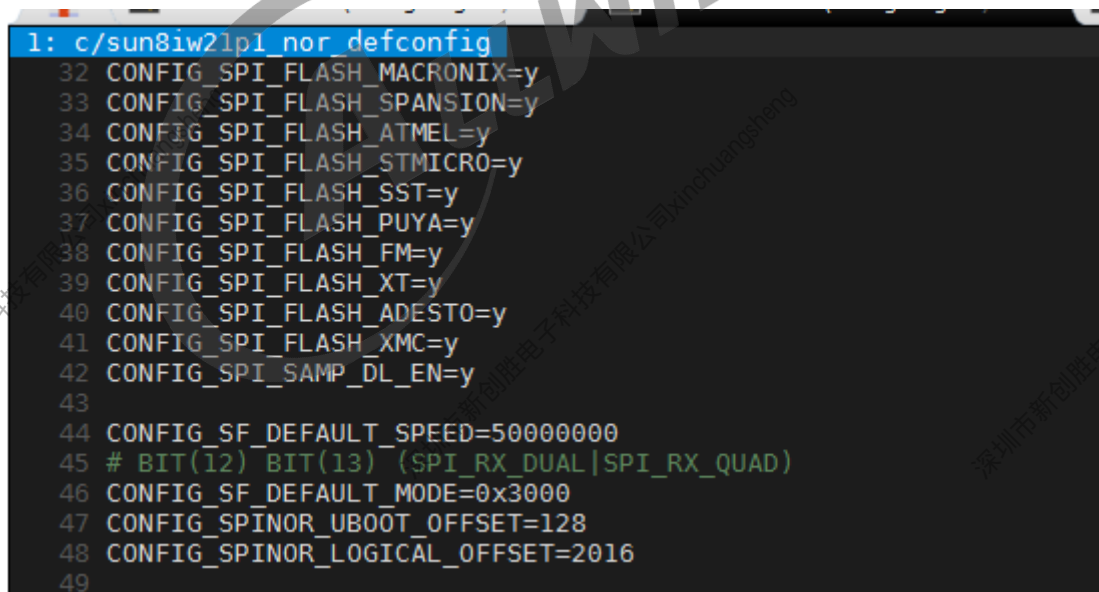
SPL 部分:spl/board/sun8iw19p1/spinor.mk, 修改 CFG\_SPINOR\_UBOOT\_OFFSET。



```
1: spinor.mk
1 #
2 #config file for sun8iw19
3 #
4
5 FILE_EXIST=$(shell if [ -f $(TOPDIR)/board/$(PLATFORM)/common.mk ]; then echo yes; else echo no; fi;)
6 ifeq (x$(FILE_EXIST),yes)
7 include $(TOPDIR)/board/$(PLATFORM)/common.mk
8 else
9 include $(TOPDIR)/board/$(CP_BOARD)/common.mk
10 endif
11
12 MODULE=spinor
13 CFG_SUNXI_SPINOR =y
14 CFG_SUNXI_SPI =y
15 CFG_SUNXI_DMA =y
16 CFG_SPI_USE_DMA =y
17 CFG_SPINOR_UBOOT_OFFSET=128
```

图 3-3: boot0 spinor 配置文件

uboot 部分: 在路径u-boot-2018/configs/sun8iw21p1\_nor\_defconfig下，修改CONFIG\_SPINOR\_UBOOT\_OFFSET。



```
1: c/sun8iw21p1_nor_defconfig
32 CONFIG_SFI_FLASH_MACRONIX=y
33 CONFIG_SFI_FLASH_SPANSION=y
34 CONFIG_SFI_FLASH_ATMEL=y
35 CONFIG_SFI_FLASH_STMICRO=y
36 CONFIG_SFI_FLASH_SST=y
37 CONFIG_SFI_FLASH_PUYA=y
38 CONFIG_SFI_FLASH_FM=y
39 CONFIG_SFI_FLASH_XT=y
40 CONFIG_SFI_FLASH_ADESTO=y
41 CONFIG_SFI_FLASH_XMC=y
42 CONFIG_SFI_SAMP_DL_EN=y
43
44 CONFIG_SF_DEFAULT_SPEED=50000000
45 # BIT(12) BIT(13) (SPI_RX_DUAL|SPI_RX_QUAD)
46 CONFIG_SF_DEFAULT_MODE=0x3000
47 CONFIG_SPINOR_UBOOT_OFFSET=128
48 CONFIG_SPINOR_LOGICAL_OFFSET=2016
49
```

图 3-4: uboot 配置文件

MBR 偏移的配置 uboot 部分: 在路径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。

```
./d/boot_package_nor.cfg
[package]
;item=Item_TOC_name,      Item_filename,
;item=scp,                scp.fex
;item=optee,              optee.fex
;item=u-boot,>--->--->--- u-boot-spinor.fex
;item=dtb,                sunxi.fex
;item=logo,               bootlogo.bmp.lzma
;item=shutdowncharge,     bempty.bmp.lzma
;item=androidcharge,      battery_charge.bmp.lzma
```

图 3-5: boot-package 配置文件

#### 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 =  $2016 - (128 + \text{使用空间}[1504])$ 。
- 修改值 =  $(2016 - \text{可优化大小} + 64(\text{预留空间}))$ 。
- 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  $((\text{修改值} + 32) / 2 - 16) * 1024$ 。
- 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  $((\text{修改值} + 32) / 2 - 16) * 1024$ 。

## 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 的数据手册，例如某款 emmc 物料的 usr 分区的大小在图中的红框位置。

##### 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 H/W reset 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 <sup>1</sup>		6	-	[511:506]	-			
Extended Security Commands Error	EXT_SECURITY_ERR	1	R	[505]	0x00			

图 4-1: eMMC usrspace

Sleep Notification Timeout	SLEEP_NOTIFICATION_TIME	1	R	[216]	0x07			
<b>Sector Count</b>	SEC_COUNT	4	R	[215:212]	<b>0xE90000</b>	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: eMMC SectorCount

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 dlinfex 15269888 40960 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_perfl_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_perfl_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

在打包工具目录下运行 make 编译生成 programmer\_img，会自动拷贝到longan/tools/pack/pctools/linux/mod\_update/目录。



## 4.2.3 打包生成 eMMC 烧录器固件

longan 的环境打包命令：#./build.sh pack\_rawo

```
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 40960 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  
--40960-----该flash的逻辑地址，单位为sector，注意旧版本单位为Mbytes  
--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，将整个文件填充 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。

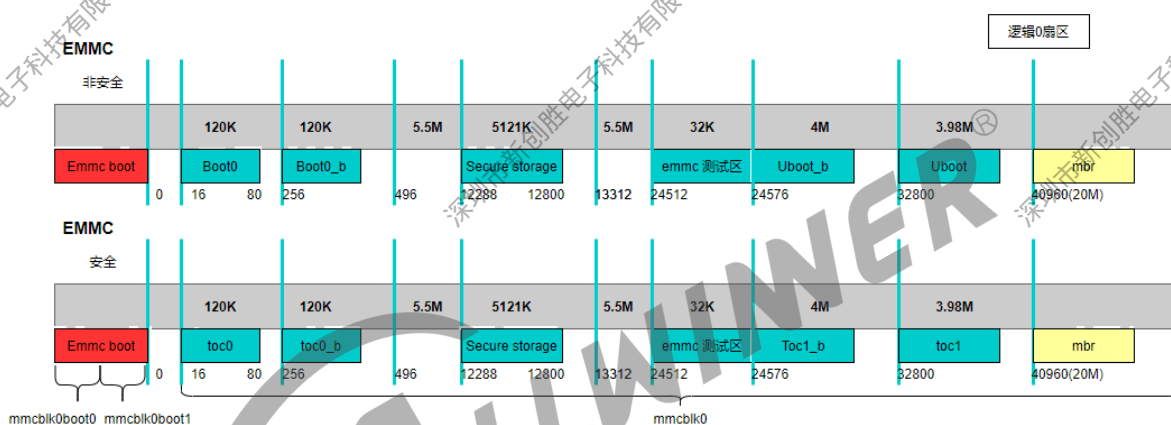


图 5-1: eMMC 存储分布图

## 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。

NOR

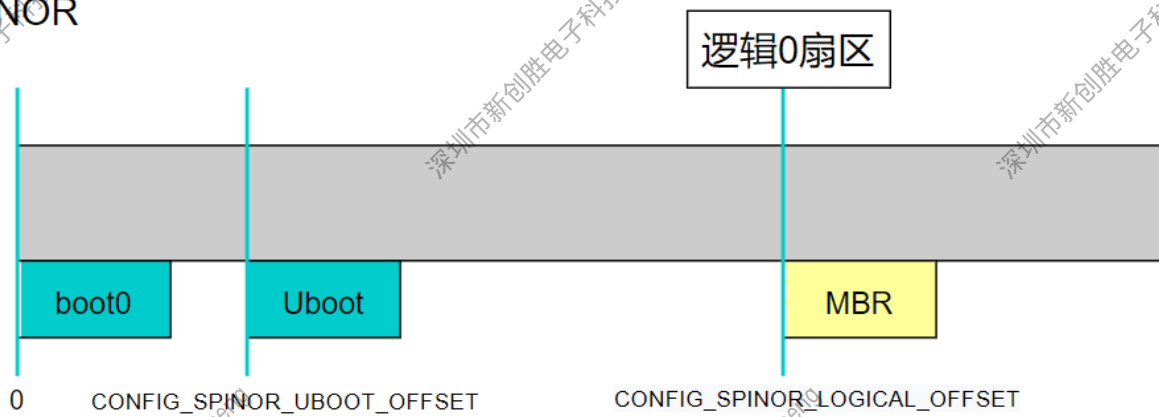


图 5-2: nor 存储分布图

## 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：

```
3 --- a/pack
4 +++ b/pack
5 @@ -1229,7 +1229,13 @@ function do_pack_android()
6     for fex_name in ${fex_list}; do
7         img_name=${fex_name%.fex}.img
8         if [ -f ${ANDROID_IMAGE_OUT}/${img_name} ]; then
9             ln -sf ${link_real}/${img_name} ${fex_name}
10 +         if [ "${PACK_PROGRAMMER}" = "xprogrammer" -a "${fex_name%.fex}" = "xsuper" ]; then
11 +             simg2img ${link_real}/${img_name} ${link_real}/${img_name}.tempbin
12 +             img_name=${img_name}.tempbin
13 +             ln -sf ${link_real}/${img_name} ${fex_name}
14 +         else
15 +             ln -sf ${link_real}/${img_name} ${fex_name}
16 +         fi
17 +         echo "link ${img_name} -> ${fex_name}"
18     done
19 fi
```

## 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, 备份是否在第 256 个扇区 (特殊的如 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, 备份是否在第 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 的方式拷贝到 eMMC 上即可完成离线烧录操作, 如下图。

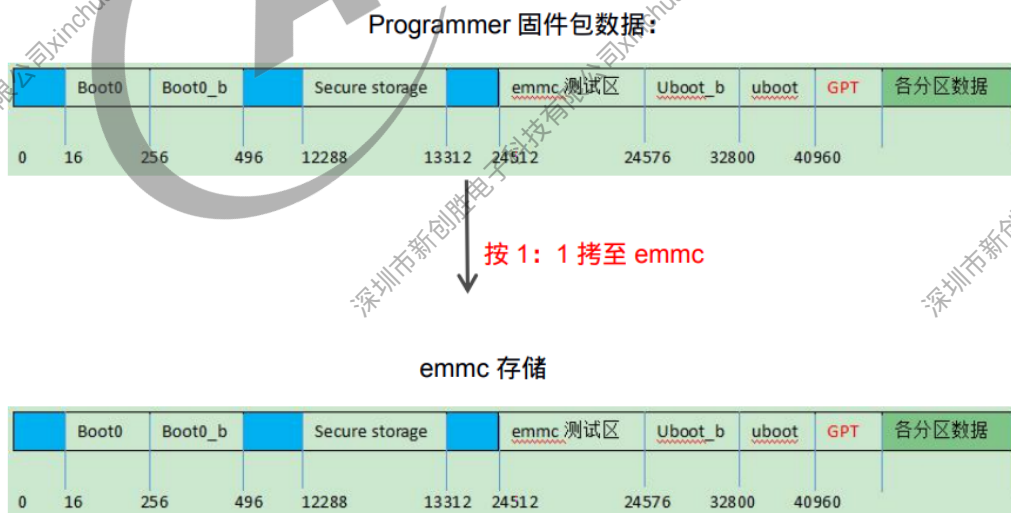


图 6-1: programmer 到 flash 数据分布

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. 使用 update\_mbr 工具生成 GPT。

- ./update\_mbr sys\_partition\_nor.bin 1 sunxi\_mbr\_nor.fex dlinfo.fex 65535 2016 1
- 65535 – nor flash 大小，单位为 sector
- 2016 – 逻辑偏移，单位为 sector

2. 使用 merge\_full\_img 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. 将生成的烧录器固件。

- adb push full\_img.fex ./

4. 使用烧录器将固件写入 flash，然后重启。

## 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。






## 著作权声明

版权所有 © 2023 珠海全志科技股份有限公司。保留一切权利。

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。