SigmaStar SmartCamera
BOOT LOGO 以及显示参数使用参考

REVISION HISTORY

Revision No.	Description	Date
{0.1}	• {Initial release}	{02/08/2019}
{0.2}	• {支持多屏参多 logo 动态切换，升级 UI 显示}	{03/31/2020}
{0.3}	• {支持 MISC 分区}	{06/09/2020}

TABLE OF CONTENTS

1. 概述

1.1. 概述

提供了在 SDK 编译环境中添加、删除、修改 Boot logo 的方法。

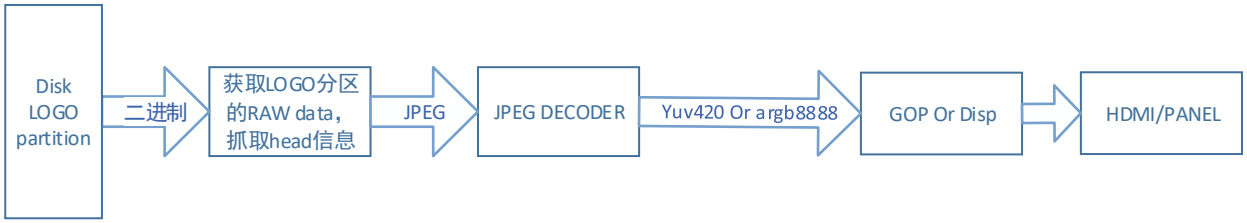
更改 logo 前需要注意以下几个地方：

1. 目前 bootlogo 只支持 jpg 图片格式，因为是软件解码，所以图片太大解码速度会很慢，导致出图慢。公版默认分配的 bootlogo buffer 为 1M，图片的分辨率需要满足 width*height*1.5 < 1M；图片大小需要小于 logo 分区的大小 128KB
2. 图片的宽需要 16 对齐

2. BOOT LOGO 显示原理

在 Uboot 中已经实现了一套解码 JPG、显示的驱动，使用者通过 bootcm 去显示 BOOT LOGO。
BOOT LOGO 显示分为以下几个步骤：

1、 从固定的 LOGO 分区中抓取屏参或者 HDMI/VGA 的显示相关信息、原始的 JPEG data。
2、 显示相关信息保存了多个屏参数据，uboot 会根据当前的 env 中的命名找到所对应的屏参数据。
3、 原始的 JPGE data 也支持多个，根据用户参数可以动态切换所要显示的 JPGE。
4、 显示信息中还提供了 uboot 用于显示的 buffer 地址以及屏幕分辨率和显示刷新率。
5、 把 JPEG RAW data 送进解码器解码。
6、 有两种方法可以把解码器解出的数据用于显示，其一是解出 YUV420 数据用于 DISP 的显示，显示内容在 video 层，其二是解出 ARGB8888 格式送给 GOP，显示的内容在图形层。
7、 初始化 HDMI、Panel，显示 BOOT LOGO。



3. MISC 与 LOGO 分区

Logo 支持两种分区格式，其一种是 LOGO 分区，围绕 LOGO 分区有相关的配置和分区制作的方法，在第 5 章节会有介绍。

另外一种分区格式是 MISC 分区，MISC 分区使用的是开源的 littlefs 文件系统，此文件系统支持读写，比较容易跨平台移植，rtk、uboot 和 linux 中都有支持，并且读取速度非常快，支持掉电保护以及有不错的写均衡机制，因此利用此文件系统可以替代原来的 LOGO 分区。屏参文件和图片 jpeg 文件可以做到在 uboot、linux、rtk 共享一套配置，相对于 LOGO 分区，使用 MISC 分区对于使用者来说更加方便对屏参进行修改。

在 misc 分区中，替代屏参数据的是 INI 脚本，在 uboot 中也集成了一套 INI parser 的代码，用于屏参数据解析，将来也可以扩展成其他功能配置的 INI 解析。

屏参的基本配置都在 INI 中，这份 INI 与 LINUX 共用，因此对于客户端，维护一套屏参就可以了，下表列出 LOGO 分区与 MISC 分区几个特性。

	LOGO 分区	MISC 分区
动态支持屏参变更	支持	支持
屏参参数动态修改	不支持	支持
开机 logo 图片动态修改	不支持	支持
屏参数据解析时间（估算）	55ms	59ms
是否与 LINUX 共用屏参	否	是
支持在 Linux 上 mount	不支持	支持
分区大小(spinand)	384KB	384KB

MISC 分区的 littlefs 使用的 Block size 为 32KB，因此对于 nor flash，在 MISC 分区中不宜添加多个文件，特别是多个很小的 INI 配置脚本，若遇到这种情况，请考虑把所有的配置都集中写到一个 ini 配置脚本中。

Uboot 的在开机 logo 阶段首先会先读取当前的 MTD part 中是否存在带”misc”、”logo”字段的分区，若存在 misc 分区，则会读取 misc 分区中的屏参信息进行点屏和显示 logo 的动作，否则则会读取 LOGO 分区中的相关信息。

4. BOOT LOGO 相关配置

4.1. Enable bootlogo hdmi

Uboot:

打开如下 config:

CONFIG_CMD_BOOTLOGO
CONFIG_SSTAR_DISP
CONFIG_SSTAR_DISP_HDMITX_VGA
CONFIG_SSTAR_HDMITX
CONFIG_SSTAR_JPD

4.2. Enable bootlogo panel

Uboot:

打开如下 config:

CONFIG_CMD_BOOTLOGO
CONFIG_SSTAR_DISP
CONFIG_SSTAR_PNL
CONFIG_SSTAR_JPD

4.3. Enable MISC partition & INI Parser

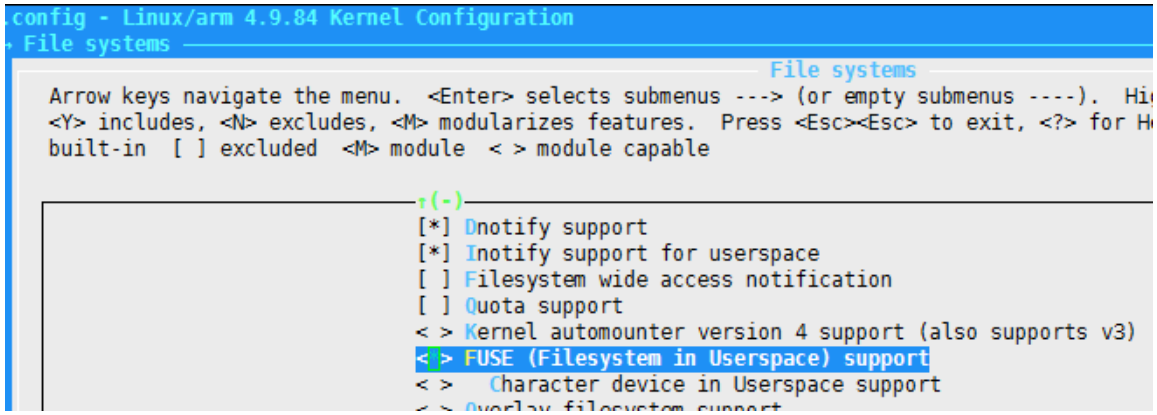
Uboot:

打开如下 config:

CONFIG_FS_LITTLEFS=y
CONFIG_SSTAR_INI_PARSER=y

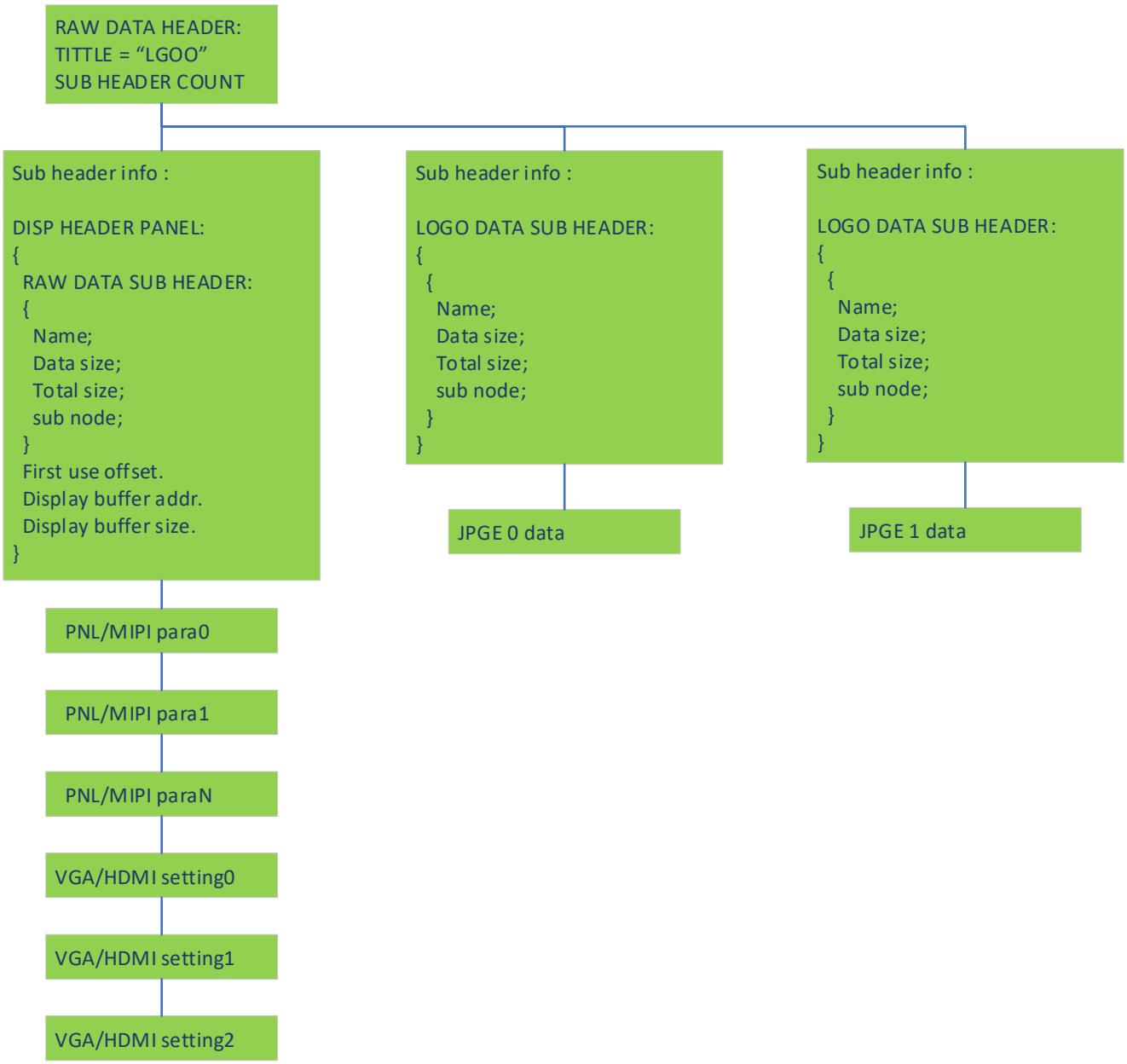
Kernel:

Build in fuse.



5. LOGO 分区制作

5. 1. LOGO 分区基本结构：



LOGO 分区分为 Partition 的 header 和若干个 sub header。

Partition header 中保存了 sub header 的计数以及 Tittle。

目前 Sub header 有两种数据，其一种是保存了屏参以及 HDMI/VGA 显示参数的数据，这些数据是以 C 语言结构体的形式保存在 Flash 中，另外一种是若干个 Sub header 保存的是 Jpeg 的图片。

在显示数据中，字段 Frist_use_offset 是用于索引当前默认使用的屏参或者 hdmi/vpa 的设定。

5.2. Project 的开机 LOGO 配置(LOGO 分区)

BOOT LOGO 的基本修改是基于 LOGO 分区已经创建，同时使用显示设备(display out: panel/HDMI/VGA)已经配置完成，在此基础上可以修改 LOGO 分区中指定的优先使用的显示设备，用于显示的内存地址，以及开机 LOGO 的文件名。

1、 打开 build config 文件修改三个字段：

BOOTLOGO_FILE = sigmastar1024_600.jpg

BOOTLOGO_ADDR = E_LX_FB

DISP_OUT_NAME = SAT070CP50

2、 BOOTLOGO_FILE 表示在 project\board\ini\misc\下使用对应的文件名用于 logo 显示。

3、 BOOTLOGO_ADDR 用于显示用的内存地址。

4、 DISP_OUT_NAME 使用的显示输出 name，如何制定可以参考 LOGO 分区制作。

5.3. LOGO 分区制作源码及工具使用

LOGO 的 RAW data 分区生成的源码路径在：project\image\makefiletools\src\rawgenerator。

在此路径下执行 make，会自动把可执行的 bin copy 到 project\image\makefiletools\bin\用于在编译阶段生成对应的 LOGO image。

LOGO 分区生成的脚本：project\image\image.mk

```
logo_nofsimage:
    @echo [[$@]]
    $(PROJ_ROOT)/image/makefiletools/bin/dispcfggen -c -o $(logo$(RESOUCE)) -p $(LOGO_ADDR) -s $(BOOTLOGO_BUFSIZE) -d $(DISP_OUT_NAME)
    $(PROJ_ROOT)/image/makefiletools/bin/logogen -a -i $(PROJ_ROOT)/board/ini/misc/$(BOOTLOGO_FILE) -o $(logo$(RESOUCE))
    $(PROJ_ROOT)/image/makefiletools/bin/logogen -a -i $(PROJ_ROOT)/board/ini/misc/upgrade.jpg -o $(logo$(RESOUCE))
```

LOGO 的 image 分别有两个工具生成，分别是 dispcfggen 和 logogen，前者用于产生显示相关的参数数据，后者生成由图片组成的 LOGO 数据。

显示数据部分生成的代码路径：

project\image\makefiletools\src\rawgenerator\disp_data_main.c

Jpeg 图片数据生成的代码路径：

project\image\makefiletools\src\rawgenerator\logo_data_main.c

使用参数说明：

dispcfggen：

-c 创建一个 Raw data 的 header，并添加显示的 sub header 和数据，若-o 指定的文件存在，则清空文件。

-a 在已经有 Raw data header 和相关数据的文件末尾追加一个显示相关的 sub header 以及数据。

-o 指定输出的文件。

-p 板子上用于显示的物理地址

-s 显示地址的内存大小

-d 显示输出的索引名

logogen：

-c 创建一个 Raw data 的 header，并添加 jpeg 文件的 sub header 和数据，若-o 指定的文件存在，则清空文件。

-a 在已经有 Raw data header 和相关数据的文件末尾追加一个 jpeg 文件的 sub header 以及数据。

-o 指定输出的文件。

-i jpeg 文件的路径。

5.4. 修改显示数据

显示数据分成两部分，一部分是屏参，用于 panel 显示，另外一部分是 HDMI/VGA 显示输出，这两种属性在使用上是互斥的，用户一旦设定了 DISP_OUT_NAME 为屏参的名字则输出的显示数据中都是屏参数据，否则若设定的是以 HDMI/VGA 开头的名字，则只会输出 HDMI/VGA 的参数数据，例如指定 HDMI_1080P60。

5.4.1 屏参修改

1、 屏参文件可以向 FAE 获取，或者自己根据相关文档进行配置。

2、 屏参文件是一个已经初始化好的结构体数据，存放在一个 h 文件中在路径：project\image\makefiletools\src\rawgenerator\pnl 中存放了公版所有用到的屏参文件，请注意，每个屏参的头文件中声明的数据结构名字和一些 define 不得重复，否则会编译失败。

3、 在文件 disp_data_main.c 中修改或者添加 stTable，分别指定屏参名字，屏参数据结构以及 mipi 命令数据结构。特别注意，对于 ttl 的屏没有 mipi 参数，因此在 mipi 那一栏填 NULL。

```
SS_SHEADER_TableHandler_t stTable[] = {{"RM68200", &stPanel_720x1280_60_RM68200, &tPanel_RM68200_720x1280_4Lane_Sync_Pulse_RGB888},
                                       {"SAT070CP50", &stPanel_SAT070CP50_1024x600, NULL},
                                       {"ADT07016BR50", &stPanel_ADT07016BR50_1024x600, NULL},
                                       {"LX50_RM68172_V2", &stPanel_LX50_RM68172_V2, &stMipiDsiConfig_RM68172_V2},
                                       {"LX50_RM68172_V3", &stPanel_LX50_RM68172_V3, &stMipiDsiConfig_RM68172_V3},
                                       {"LX50_ICN9700", &stPanel_LX50_ICN9700, &stMipiDsiConfig_ICN9700},
                                       {"HST_800x480", &stPanel_HST, &stMipiDsiConfig_HST},
                                       {"T30P133", &stPanel_T30P133, &stMipiDsiConfig_T30P133},
                                       {"T30P117", &stPanel_T30P117, &stMipiDsiConfig_T30P117},
                                       {"SAT070BO30I21Y0", &stPanel_SAT070BO30I21Y0_1024x60, &
                                       stMipiDsiConfig_SAT070BO30I21Y0_1024x600}};
```

4、 屏参 include：

```
#include <stdio.h>
#include <fcntl.h>  //open
#include <unistd.h> //getopt
#include <string.h> //memset
#include <stdlib.h> //strtol
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#include "ss_raw_header.h"

#include "mhal_pnl_datatype.h"
#include "pnl_table_RM68200.h"
#include "SAT070CP50_1024x600.h"
#include "ADT07016BR50_1024x600.h"
#include "SAT070BO30I21Y0_1024x600_MIPI.h"
#include "LX50FWB4001_RM68172_480x854_v2.h"
#include "LX50FWB4001_RM68172_480x854_v3.h"
#include "T30P133_480x854.h"
#include "T30P117_360x640.h"
#include "HST_800x480.h"
#include "LX50FWB4001_ICN9700_480x854.h"
```

5、 以上修改完成后重新编译生成执行文件，用执行文件生成 LOGO 的 image 中就有添加的屏参，在屏参生成的参数中用-d 指定优先使用的屏参的名字，若优先使用的屏参不是正确的屏参，在 uboot 还可以通过修改 env 来改变屏参，改动方法参考”“多屏参多 logo 切换”。

5.4.2 HDMI/VGA 设定修改

1、 HDMI、VGA 的参数设定没有屏参那么复杂，只需要填写输出 timing 的宽高和 clock。

2、 修改方式如下：

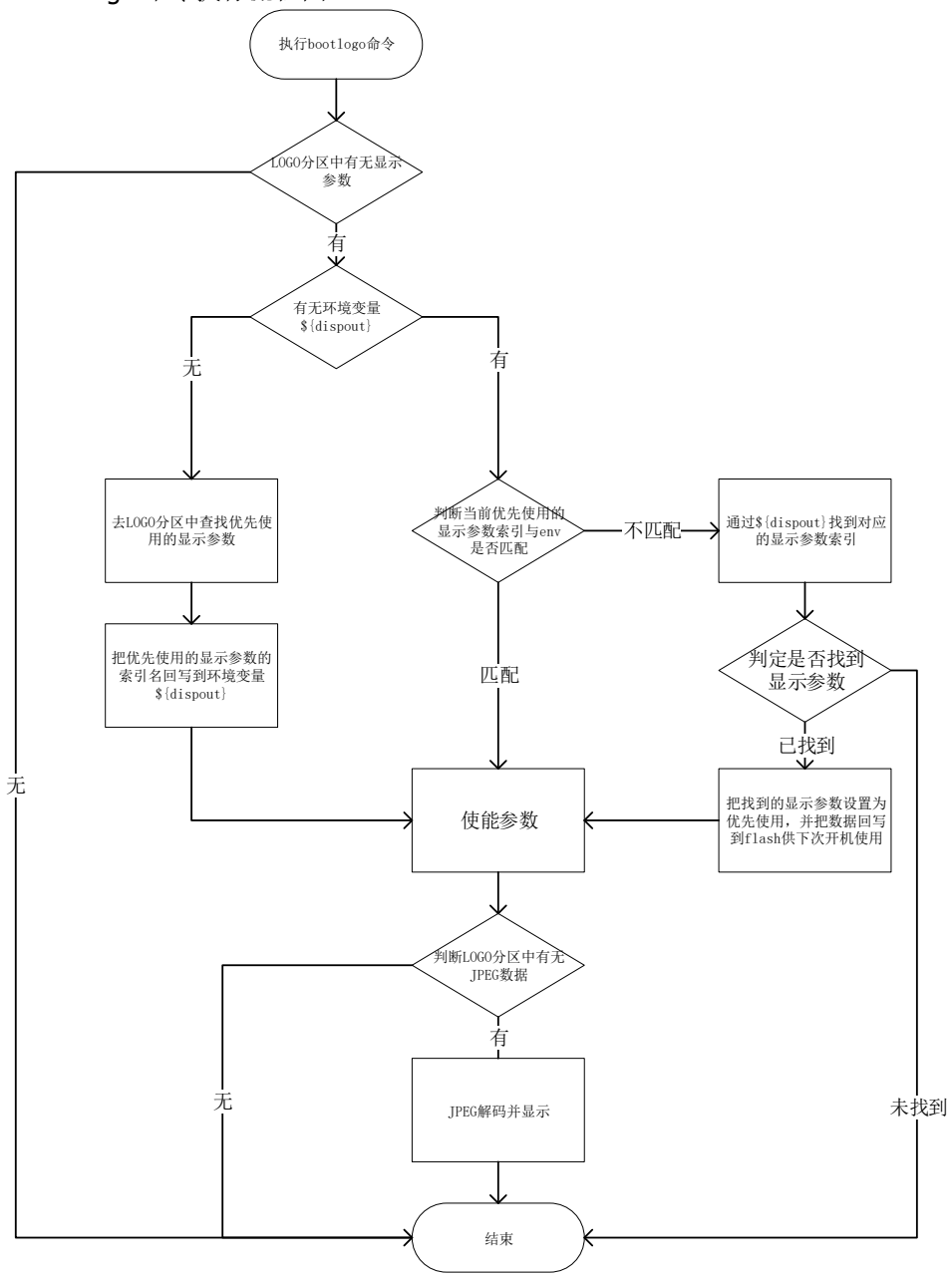
```
SS_SHEADER_DispConfig_t stTimmingTable[] = {{EN_DISPLAY_DEVICE_HDMI, "HDMI_1080P60", 1920, 1080, 60},
                                            {EN_DISPLAY_DEVICE_HDMI, "HDMI_720P60", 1280, 720, 60},
                                            {EN_DISPLAY_DEVICE_VGA, "VGA_1080P60", 1920, 1080, 60},
                                            {EN_DISPLAY_DEVICE_VGA, "VGA_720P60", 1280, 720, 60}};
```

5.5. 多显示参数切换

LOGO 的分区中保存了所有的显示参数数据，目前可以通过 env 来切换不同的显示参数。

使用 uboot 的 env ${dispout}当中显示设置的索引，默认情况下此 env 为空，在第一次上电起来后若在 uboot 阶段执行了 bootlogo 命令，则会把当前使用的设定对应的名字保存到${dispout}。

‘bootlogo’命令执行流程图：



通过以上流程图可知，在未指定环境变量${dispout}的情况下，系统会默认设定优先使用的显示参数，在${dispout}与优先使用的显示参数发生不一致的情况，系统则会自动按照${dispout}设定的值去索引对应的显示参数，若找到则回写到 flash 中，这样下次开机的流程就按照与环境变量匹配的流程执行。

6. MISC 分区制作与配置

6.1. MISC 分区制作

MISC 分区的文件系统是 littlefs，littlefs 的打包和 linux 端自动 mount 分区都已经在 ALKAID 的打包脚本中集成好了，如何添加一个 MISC 分区，请参考《分区介绍文档》，这里简单地把一些配置列出来：

根据分区文档中的介绍，若要添加一个 MISC 分区，需要在 project\image\configs\$(chip)中修改相应的 xxx.config 文件：

举例：

1、 添加分区的基本配置

```
misc$(RESOUCE) = $(OUTPUTDIR)/misc
misc$(FSTYPE)  = lfs
misc$(PATSIZE) = 0x60000
misc$(MOUNTPT) = /dev/mtdblock8
misc$(MOUNTTG) = /misc
misc$(MTDPART) = $(misc$(PATSIZE))(misc)
misc$(OTABLK) = /dev/mtdblock8
```

由于 littlefs 已经得到了 ALKAID 打包脚本的支持，因此在这里 misc$(FSTYPE)填成 lfs 后就可以自动打包成 lfs 的 image 包了。

2、 添加 mount 节点列表

```
USR_MOUNT_BLOCKS:=miservice customer appconfigs misc
```

3、 添加 image 列表

```
IMAGE_LIST = cis ipl ipl_cust uboot misc kernel rootfs miservice customer appconfigs
```

4、 添加 OTA 列表（如必要）

```
OTA_IMAGE_LIST = ipl ipl_cust uboot misc kernel miservice customer appconfigs
```

5、 在 misc.mk 或者 rootfs.mk 中添加文件拷贝命令

```
mkdir -p $(OUTPUTDIR)/misc;
mkdir -p $(OUTPUTDIR)/rootfs/misc;
cp -rf $(PROJ_ROOT)/board/ini/misc/sigmastar1024_600.jpg $(OUTPUTDIR)/misc/;
cp -rf $(PROJ_ROOT)/board/ini/misc/upgrade.jpg $(OUTPUTDIR)/misc/;
cp -rf $(PROJ_ROOT)/board/ini/misc/config.ini $(OUTPUTDIR)/misc/;
```

6.2. INI 文件中 LOGO 相关配置

关于屏参和 logo 显示相关的配置都集成在了/misc/config.ini 中，ini 文件可以在 linux 起来后在路径/misc/下手动修改，也可以重新打包一个 misc 分区，对分区重新烧写。对 ini 中相关的 logo 参数的修改就可以起到切换不同屏参，显示不同 logo 的效果，具体做法如下。

所有对 LOGO 显示配置的修改都集中在 INI 的"LOGO"的 section 中：

```
[LOGO]
m_eDeviceType = 1; #0 None, 1 LCD, 2 HDMI, 3 VGA
m_sParaTarget = ST7710S
m_wDispWidth = 0
m_wDispHeight = 0
m_wDispFps = 0
m_sLogoFile0 = sigmastar1024_600.jpg
m_sLogoFile1 = upgrade.jpg
```

m_eDeviceType：

表示需要显示 LOGO 的设备类型。

m_sParaTarget：

显示的屏参 section 索引，只有 LOGO 设备类型为 LCD 时才会生效。

m_wDispWidth，m_wDispHeight，m_wDispFps：

当显示设备为非 LCD 类型时，这些参数才有意义，表示的是 HDMI/VGA 输出的 timming。

m_sLogoFile0，m_sLogoFile1

LOGO 的 id 所对应的 jpeg 文件。

INI 文件修改完成后重启生效。

7. UBOOT 命令

7.1. Uboot 里的 bootlogo 命令介绍

在 Uboot 中执行 bootlogo [logo index] [aspect ratio] [x] [y]，此命令会带四个参数分别表示：

1: LOGO 中打包的 JPEG 的数字索引。

2：显示 logo 的 aspect ratio，0 表示 zoom，全屏显示，1 表示 center，居中显示，2 表示 user，用户设定起始点，点对点显示。

3：若 aspect ratio 为 user，x、y 表示图片显示的起始点。

7.2. LOGO 旋转显示

在 uboot 中修改 env ‘logo_rot’，修改完之后重启，或者再次执行 bootlogo 命令。

logo_rot = 0 表示不旋转

logo_rot = 1 顺时针旋转 90°

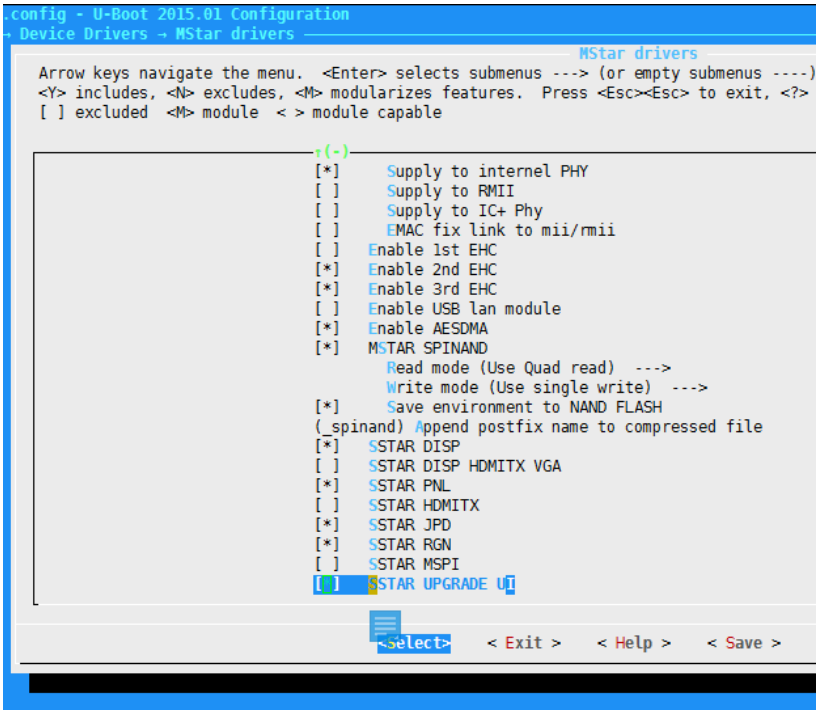
logo_rot = 2 顺时针旋转 180°

logo_rot = 3 顺时针旋转 270°

LOGO 旋转后画面左上角仍然从显示的(0, 0)坐标开始显示，其图片的宽和高有可能发生调换，请注意旋转后的图片都必须在显示的区域内，否则显示驱动会报错。

8. SD/USB 升级 UI 显示

在 uboot 中需要打开 SSTAR UPGRADE UI，此命令依赖于 SSTAR RGN 、SSTAR DISP。



SD 卡和 USB 升级的 UI 由一张背景图片和一个进度条组成。

UI 背景图片依赖于 bootlogo 命令，boologo 显示第二张图片作为升级的 UI。

目前使用的图片在：

project\board\ini\misc\upgrade.jpg

图片+进度条显示效果：

9. LINUX 使用屏参

Linux 代码中通过读取 LOGO 分区的数据可以获取当前的屏参，在应用端无需再维护多个屏参的头文件，这样可以增加代码的通用性，目前从 flash 中获取屏参的接口还未开放出来。

若使用的是 MISC 分区，屏参文件和 jpg 图片会自动 mount 到/misc 目录下，应用可以参考 Panel 文档中的介绍如何使用屏参 ini 文件。