USB&OAD 升级详解

V0.2

REVISION HISTORY

Revision No.	Description	Date
V0.1	• Add description for USB upgrade	02/17/2014
V0.2	• Add how to make MstarUpgrade.bin	04/05/2014
	• Add description for OAD upgrade	04/20/2014

LIST OF FIGURES

1. 简介

1.1. 用途

- USB 升级主要用于客户的小批量生产；
- OAD 升级主要用于 pure linux 系统的网络升级；

1.2. USB 升级简介

- USB 升级使用于 android 以及 pure linux 系统的完整（部分分区）升级；
- 通过特定的 shell 脚本制作出来相应的 USB 升级包；
- **将制作出来的 USB 升级包 copy 至 U 盘的根目录**，通过特定的方式进入 USB 升级；
- Mboot 解析 USB 升级包中的脚本进行整个系统（部分分区）的升级；
- 升级完成之后会自动重启，此时的系统即是通过 USB 升级之后的新系统；

1.3. OAD 升级简介

- OAD 升级使用于 pure linux 系统的完整（部分分区）的网络升级；
- 通过特定的 shell 脚本制作出来相应的 OAD 升级包；
- **将制作出来的 OAD 升级包上传至网络，通过码流的方式下载至 OAD 分区**，通过特定的方式进入 OAD 升级；
- Mboot 解析 OAD 升级包中的脚本进行整个系统（部分分区）的升级；
- 升级完成之后会自动重启，此时的系统即是通过 OAD 升级之后的新系统；

2. USB&OAD 升级包

2.1. USB&OAD 升级包架构

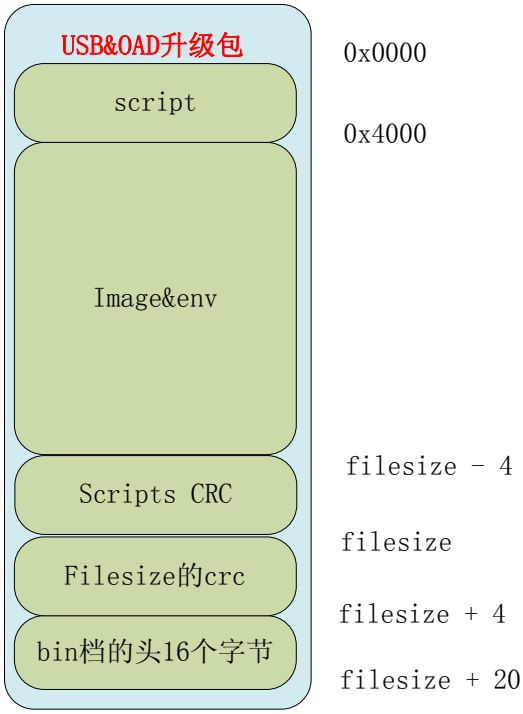


Figure 2-1: USB 升级包架构

- 0x00-0x4000 中主要包含的是升级时的执行命令；
- Image 的区域是用于存放各个分区的烧写内容，其中每个分区的内容的偏移量以及大小会存储在烧写脚本中；
- Script crc 是对 0x00 – 0x4000 这整个区域的中的内容计算出的 CRC 值，主要用于确保在 USB 升级的时候 load 的烧写脚本的正确性；
- Filesize 的 crc 是对 script、image、以及 script 计算出来的 CRC 值，目前没有使用到；
- Filesize + 4 到 filesize + 20 这部分存储的是整个 bin 档的前 16 个字节，功能不明，目前没有使用到。

2.2. 升级包分类

目前 Mstar 公版支持如下几种 USB 升级包的升级：

- MstarUpgrade.bin

- MbootUpgrade.bin
- SbootUpgrade.bin
- UrsaUpgrade.bin

2.3. MstarUpgrade.bin

MstarUpgrade.bin 是 android 以及 supernova 的 USB 升级包 bin 档，不同的只是制作的脚本不同，升级方式均相同，这里的 Supernova 指的是纯 linux 系统。

2.3.1. 获取升级包的制作脚本

2.3.1.1. Android

以 git 上的 jb4.3 android source code 为例，编译 USB 升级包的脚本路径如下：

jb4.3-mstar-master/development/scripts/make_usb_upgrade.sh

2.3.1.2. Supernova

以 nugget 为例，编译 USB 升级包的脚本路径如下：

//DAILEO/SN_Branch/SN__-04.01/Supernova/target/build_sw_upgrade_image.sh

2.3.2. 如何制作 MstarUpgrade.bin

2.3.2.1. Android

以 napoli 平台为例

- source build/envsetup.sh;lunch（选择 mstarnapoli）
- make
- ./development/scripts/releaseimage.sh
- Copy supernova、mboot 以及 RTPM 等 image 至 android source code 的同级目录下的指定位置 images/jb4.3/mstarnapoli
- ./development/scripts/make_usb_upgrade.sh
- 按照如下选项进行制作 USB 升级包

  Android ver? 1)ics; 2)jb_4.2; 3)jb_4.2.2; 4)jb_4.3 :4

  Chip? 1)amber3; 2)eagle; 3)edison; 4)kaiser; 5)nike; 6)einstein; 7)napoli :7

  Device? 1)mstarnapoli; 2)mstarnapoli_gtv; 3)mstarnapoli_stb; 4)mstarnapoli_mi :1

  enable secureboot(y/n)?n

  STORAGE? 1)EMMC ; 2)NAND (1/2):1

Upgrade all? (y/n)y

Enable upgrade systembackup? (y/n)n

Update RTPM? (y/n)y

enable upgrade mboot(y/n)?y

mboot partition type 1)SPI ;2)EMMC (1/2):2

Enable upgrade pm(PM51.bin)? (y/n)n

此时在 images/jb4.3/mstarnapoli 会生成 MstarUpgrade.bin，即为 android 系统的 USB 升级包。

2.3.2.2. Supernova

以 nugget 为例（可制作 USB/OAD 升级包）：

- cd project
- source buildsettings/build_Nugget_050B_DVB_IPC_256x1_BootFromROM.sh
- make rebuild_all
- make image_all
- copy mboot.bin to target/europe_dtv.nugget/images/ubifs
- cd ../target
- ./build_sw_upgrade_image.sh
- 按照如下选项进行制作 USB 升级包:

Please input your storage type(nand, emmc, spi, k2sqfs): nand

Is Secure Booting? (y/N)n

Is UBOOT 2011.06? (Y/n)y

Is NAND MLC MODE? (y/N)y

Upgrade for usb? (OAD choose N) (Y/n)y

Upgrade all? (Y/n)y

Add Mboot.bin in MstarUpgrade.bin? (y/N)y

如果用来制作 USB 升级包，则 Upgrade for usb? (OAD choose N) (Y/n)y 选择 y，如果用来制作 OAD 升级包，Upgrade for usb? (OAD choose N) (Y/n)n 选择 n。此时在 target/europe_dtv.nugget/images/ubifs 下生成的 MstarUpgrade.bin 即为 nugget 的 USB/OAD 升级包 bin 档。

2.3.3. 制作脚本详解

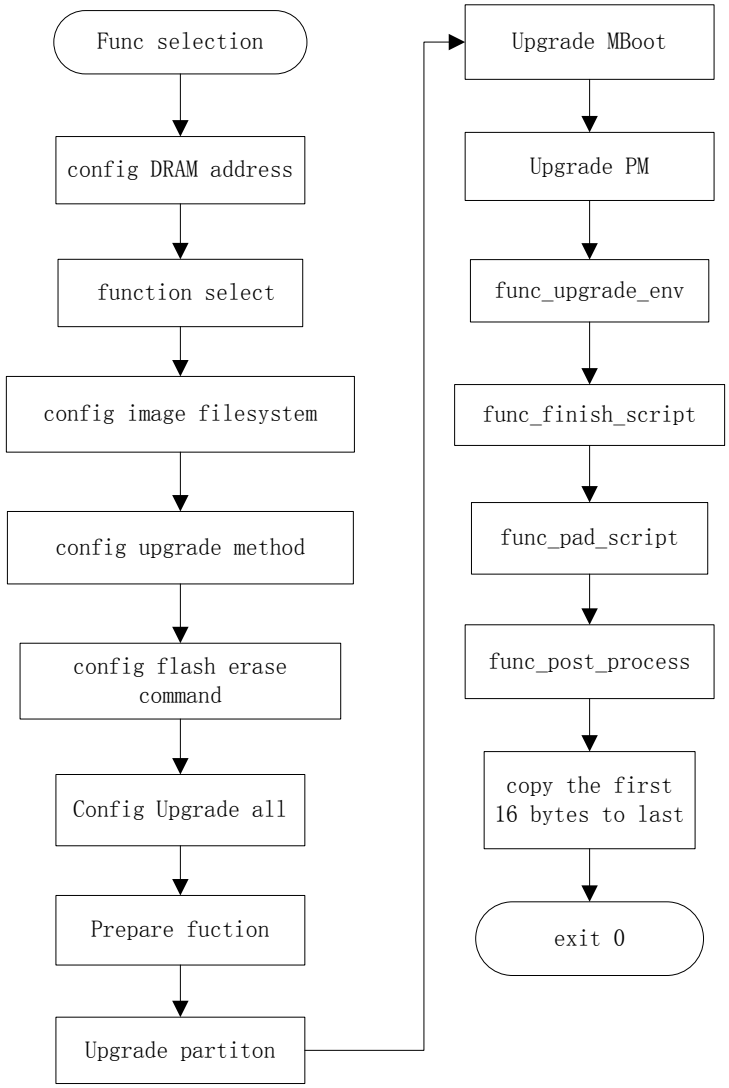


Figure 2-2:USB 流程图-AN/SN

- 配置 USB 升级包基本信息，用词信息来配置升级时 load 的内存地址，device 信息等；
- 配置 flash，emmc 还是 nand 主要用于初始化升级时所需要的命令；
- 获取分区的大小以及 image 的大小；
- 将 set_partition 中的命令写入 usb_upgrade.txt；
- 升级分区的时候会进行如下几步：

  将 image 4k 对齐；

  然后将分区的偏移量以及 image 大小写入 usb_upgrade.txt；

  将 image 内容 cat 入 usb_temp.bin；
- 如果是升级所有分区的时候会进行 func_upgrade_env 更新环境变量；
- 将 usb_upgrade.txt 16K 对齐；
- 将 usb_upgrade.txt 以及 usb_temp.bin cat 入 MstarUpgrade.bin；

- 分别计算 usb_upgrade.txt 以及 MstarUpgrade.bin 的 CRC 值然后存入 MstarUpgrade.bin；

Android 与 Supernova 中的原理基本一致不同的是 supernova 中会设置如下两个变量：

DONT_OVERWRITE：Mboot 在检测至此变量之后，会跳过该分区的所有操作；

FORCE_OVERWRITE：Mboot 在检测至此变量之后，会强制执行该分区的所有操作；

FORCE_OVERWRITE 的优先级高于 DONT_OVERWRITE。

2.4. mboot 升级包

Mboot 升级包包含 MbootUpgrade.bin 以及 SbootUpgrade.bin

2.4.1. 获取升级包的制作脚本

以 p4 mainline 上的 code 为例，编译 USB 升级包的脚本路径如下：

//DAILEO/MBoot/sboot/util/build_MbootUpgrade_image.sh

2.4.2. 如何制作 mboot 升级包

以 napoli box 为例

- cd MBoot\sboot
- cp .config.napoli.android.190a_s.emmc.rom .config
- make menuconfig（选择 exit & save）
- make clean ;make
- cd util/
- ./build_MbootUpgrade_image.sh
- 请按照如下的选择制作 mboot 升级包

  Is ARM platform (Y/n)y

  MBoot Burn in Emmc ? (y/N)y

  Fully Erase ? (y/N)y

此时在 MBoot\sboot\out 下即会生成 MbootUpgrade.bin 和 SbootUpgrade.bin

2.4.3. 制作脚本详解

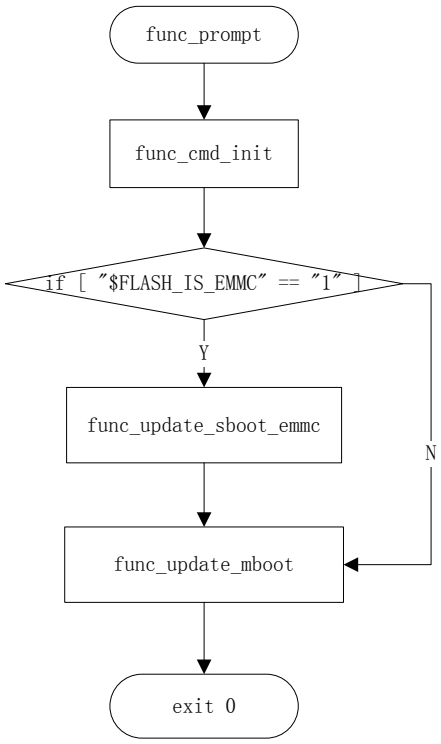


Figure 2-3: USB 流程图-MBoot

- func_prompt 中判断此时是否为 arm 平台以及是否是 emmc flash 等基本信息；
- func_cmd_init 中根据 mboot 是写入 emmc、nand 还是 spi 来初始化赋值相应的 cmd；
- func_update_sboot_emmc 主要是针对 emmc 中 boot from emmc 的平台制作的 SbootUpgrade.bin，将升级指令以及 sboot.bin 打包按照升级包的架构打包成 SbootUpgrade.bin；
- func_update_mboot 的原理同 func_update_sboot_emmc 相同，即将升级脚本以及 mboot.bin 打包生成 MbootUpgrade.bin。

2.5. UrsaUpgrade.bin

主要用于 ursa 平台升级单独升级 uinside 分区。

2.5.1. 获取升级包的制作脚本

制作 UrsaUpgrade.bin 和 MstarUpgrade.bin 的脚本是一样的，不同的只是，UrsaUpgrade.bin 是单独升级 uinside 分区，MstarUpgrade.bin 是升级所有分区。

以 git 上的 jb4.2 android source code 为例，编译 USB 升级包的脚本路径如下：

jb4.2-mstar-master/development/scripts/make_usb_upgrade.sh

2.5.2. 如何制作 UrsaUpgrade.bin

以 nikeu 平台为例

- source build/envsetup.sh;lunch（选择 mstarnikeu）
- make
- ./development/scripts/releaseimage.sh
- Copy uinside.img 至 android source code 的同级目录下的指定位置 images/jb4.2.2/mstarnikeu
- ./development/scripts/make_usb_upgrade.sh
- 按照如下选项进行制作 USB 升级包

  Android ver? 1)ics; 2)jb_4.2; 3)jb_4.2.2; 4)jb_4.3 :3

  Chip? 1)amber3; 2)eagle; 3)edison; 4)kaiser; 5)nike; 6)einstein; 7)napoli :7

  Device? 1)mstarnike; 2)mstarnikeu :2

  enable secureboot(y/n)?n

  STORAGE? 1)EMMC ; 2)NAND (1/2):1

  Upgrade all? (y/n)n

  Update recovery? (y/n)n

  Update boot? (y/n)n

  Update system? (y/n)n

  Update userdata? (y/n)n

  Upgrade cache? (y/n)n

  Upgrade tvservice? (y/n)n

  Update tvconfig? (y/n)n

  Update tvdatabase? (y/n)n

  Update tvcustomer? (y/n)n

  Enable upgrade mboot? (y/n)n

  Enable upgrade pm(PM51.bin)? (y/n)n

  Update uinside? (y/n)y

此时在 images/jb4.2.2/mstarnikeu 会生成 MstarUpgrade.bin，即为 android 系统升级 ursa 的 USB 升级包将其重命名为 UrsaUpgrade.bin 即可。

3. USB&OAD 升级

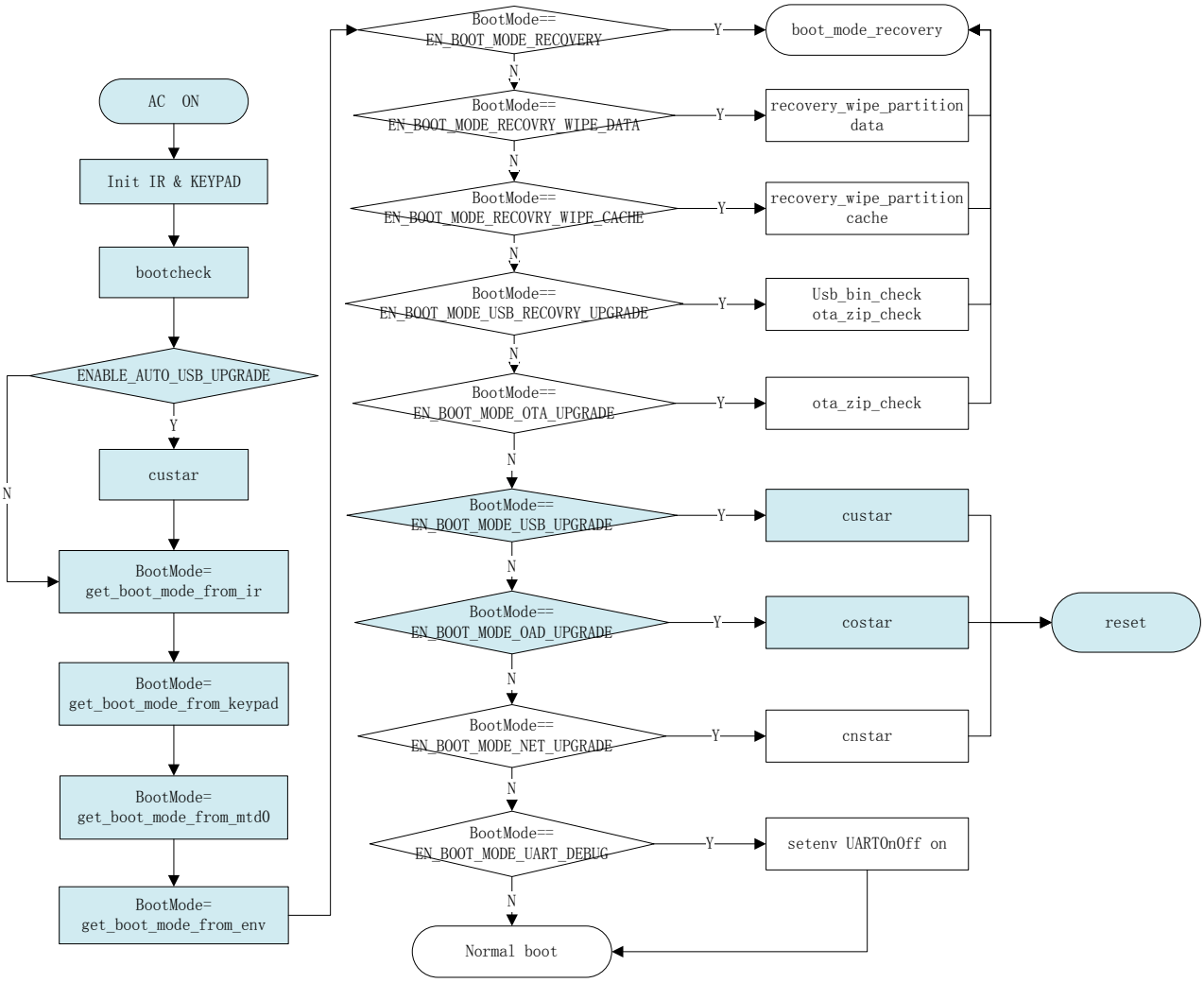


Figure 3-1:USB/OAD 流程

- 将带有升级包的板子（USB 升级的需要插入带有 USB 升级包的 U 盘；OAD 升级的需要已经在 OAD 分区下载好 OAD 升级包）AC 上电；
- 初始化 IR 以及 KEYPAD；
- 进入 bootcheck 函数检测此时是否进入 USB/OAD；
- 如果检测到开机自动 USB 升级的开关开启，则在此时就会进入 USB 升级；
- 检测此时是否有按键 VOL+，如果有，则将 bootmode 赋值为 EN_BOOT_MODE_USB_UPGRADE；
- 检测此时是否有按键 KEYKPAD（且所按的 KEYPAD 对应设置为 VOL+），如果有，则将 bootmode 赋值为 EN_BOOT_MODE_USB_UPGRADE；

- 检测此时环境变量中 force_upgrade 是否有值，或者 upgrade_mode 是否等于"usb"如果有，则将 bootmode 赋值为 EN_BOOT_MODE_USB_UPGRADE；如果 upgrade_mode 是否等于"oad"如果有，则将 bootmode 赋值为 EN_BOOT_MODE_OAD_UPGRADE；
- 根据 bootmode 的值进行 custar/costar；
- 升级完成后 reset 重启；

4. 进入 USB&OAD 升级的方式

进入 USB 升级的方式为：

- AC/DC ON；
- 开机按键 VOL+；
- 使能 KEYPAD 按键对应 VOL+；
- ENV 设置 upgrade_mode = usb；
- ENV 设置 force_upgrade = 0x08；
- 串口输入 custar；

进入 OAD 升级的方式为：

- ENV 设置 upgrade_mode = oad；
- 串口输入 costar；

4.1. IR 模式

4.1.1. 使能 BOOT IR 功能

在 make menuconfig 中将 BOOT IR 功能打开之后 mboot 才会通过获取 IR 的值做相应的升级动作

具体步骤如下：

- make menuconfig
- 进入 Module Options

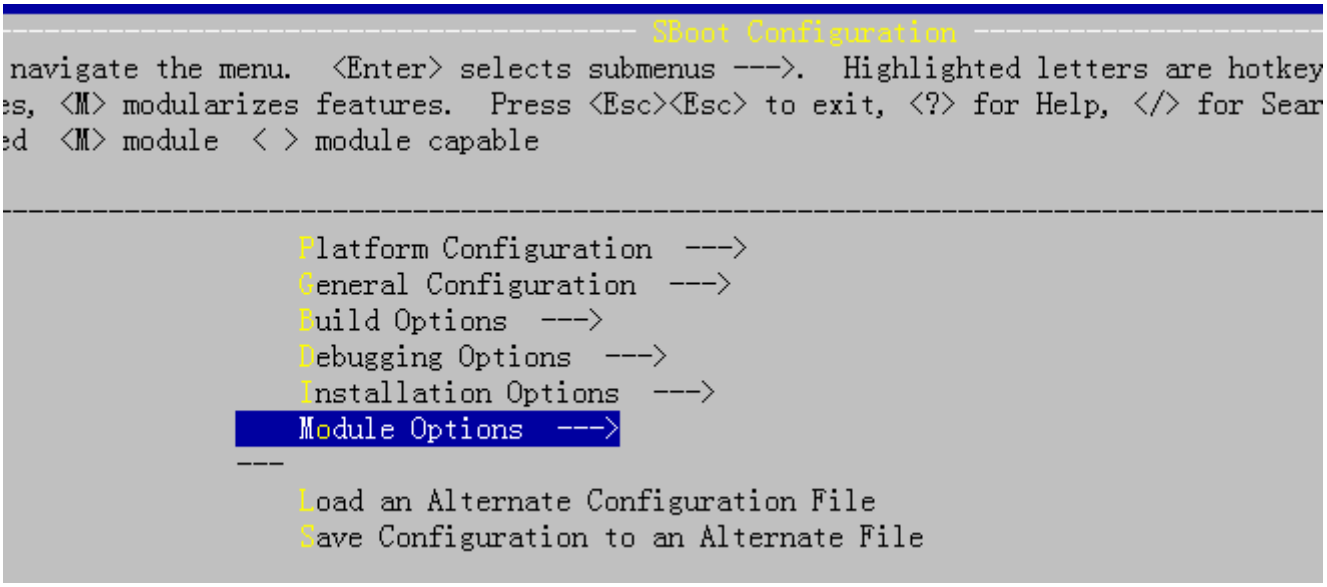


Figure 4-1: MBoot 使能 IR-1

➢ 勾选 IR 并且勾选 IR 中的 BOOT_IR

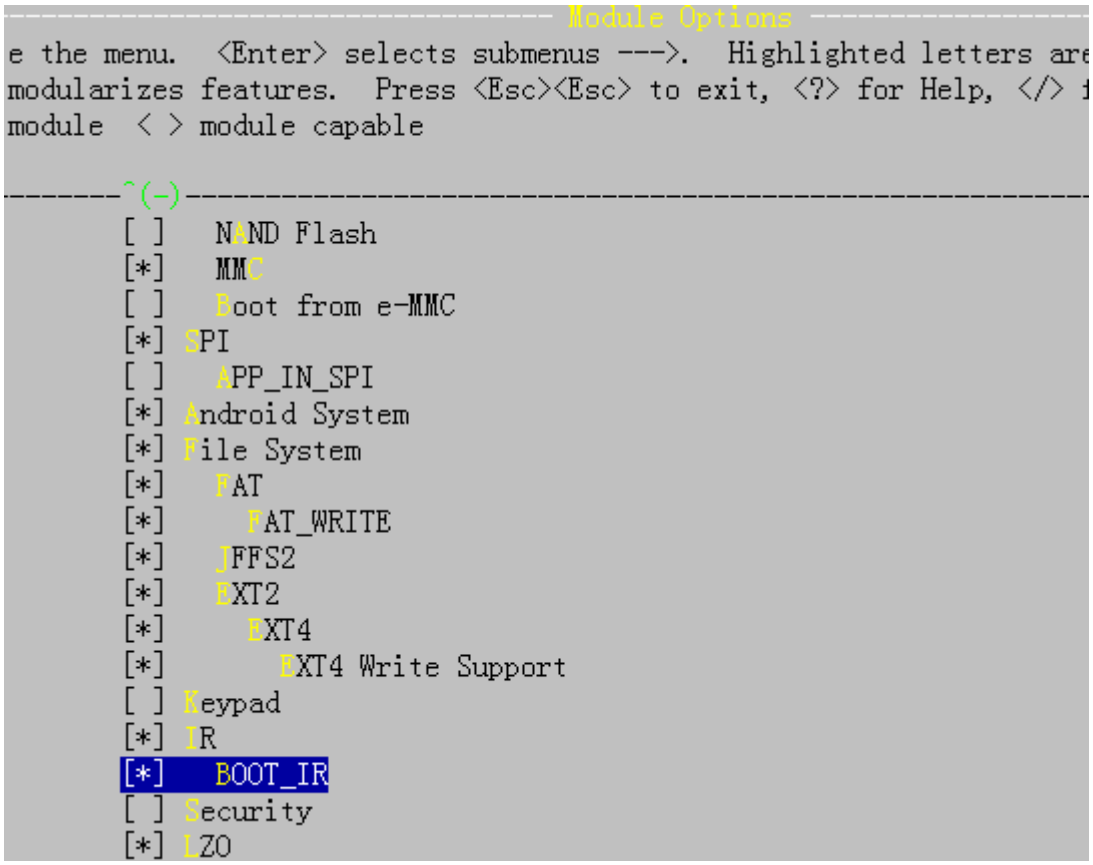


Figure 4-2: MBoot 使能 IR-2

4.1.2. 工作原理

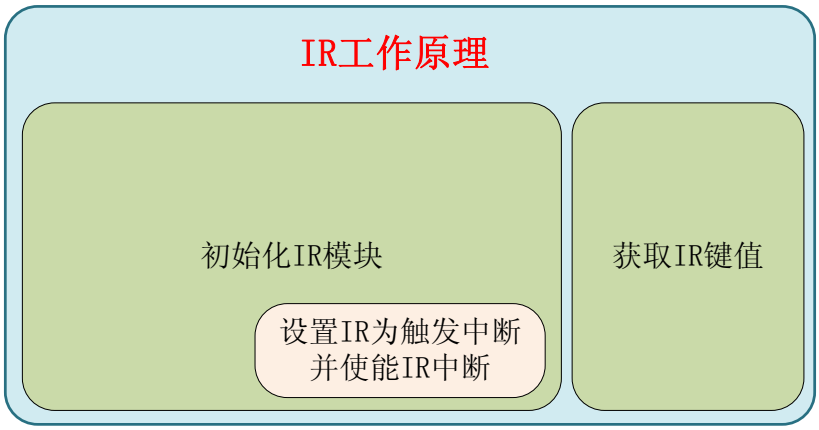


Figure 4-3: IR 工作原理

- 在初始化 IR，使能 timerout、ir_control 等功能；
- 设置 IR 为处罚中断，且使能 IR 中断
- MsOS_AttachInterrupt(E_INT_FIQ_IR, (InterruptCb)MDrv_IR_SW_Isr);
- 设置 IR 中断为触发中断；E_INT_FIQ_IR 是 IR 中断，(InterruptCb)MDrv_IR_SW_Isr 是在触发 IR 中断时执行的回调函数，在每次按键 IR 的时候都会触发 IR 中断，并且执行 MDrv_IR_SW_Isr 函数。
- MsOS_EnableInterrupt(E_INT_FIQ_IR);
- 使能 IR 中断
- MDrv_IR_SW_Isr 是解析当前 IR 的按键值，通过读取 IR 对应的 sub bank 值来获取。

4.1.3. 流程图

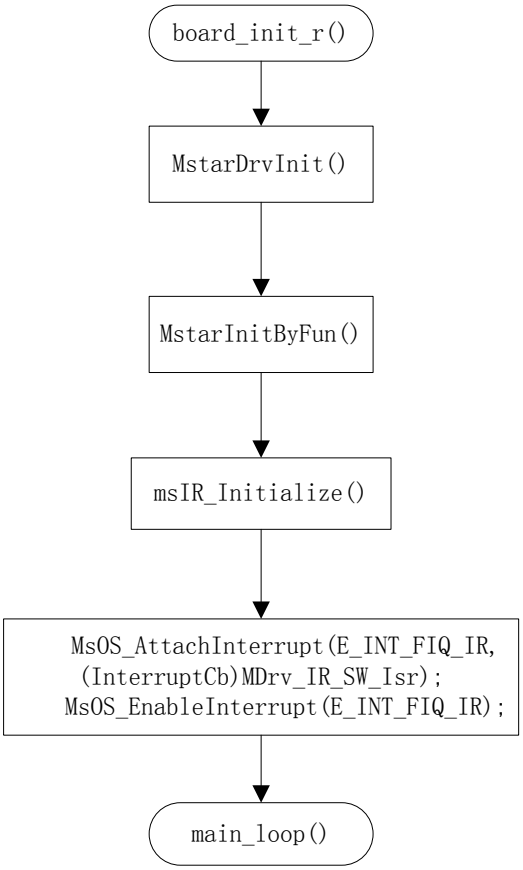


Figure 4-4: IR 流程图

4.1.4. DEBUG

4.1.4.1. IR 键值的定义

在 MstarCustomer/include/IR_MSTAR_DTV.h 中的 IrCommandType 枚举类型有定义每个按键对应的键值：

```
typedef enum _IrCommandType
{
    IRKEY_TV_ANTENNA          = 0x0C,
#if (ENABLE_DMP == DISABLE)
    IRKEY_TV_RADIO            = 0x03,//same as IRKEY_PAGE_UP
#endif
    IRKEY_CHANNEL_LIST        = 0x10,
    IRKEY_CHANNEL_FAV_LIST    = 0x08,
    IRKEY_CHANNEL_RETURN      = 0x5C,
    IRKEY_CHANNEL_PLUS        = 0x1F,
    IRKEY_CHANNEL_MINUS       = 0x19,
```

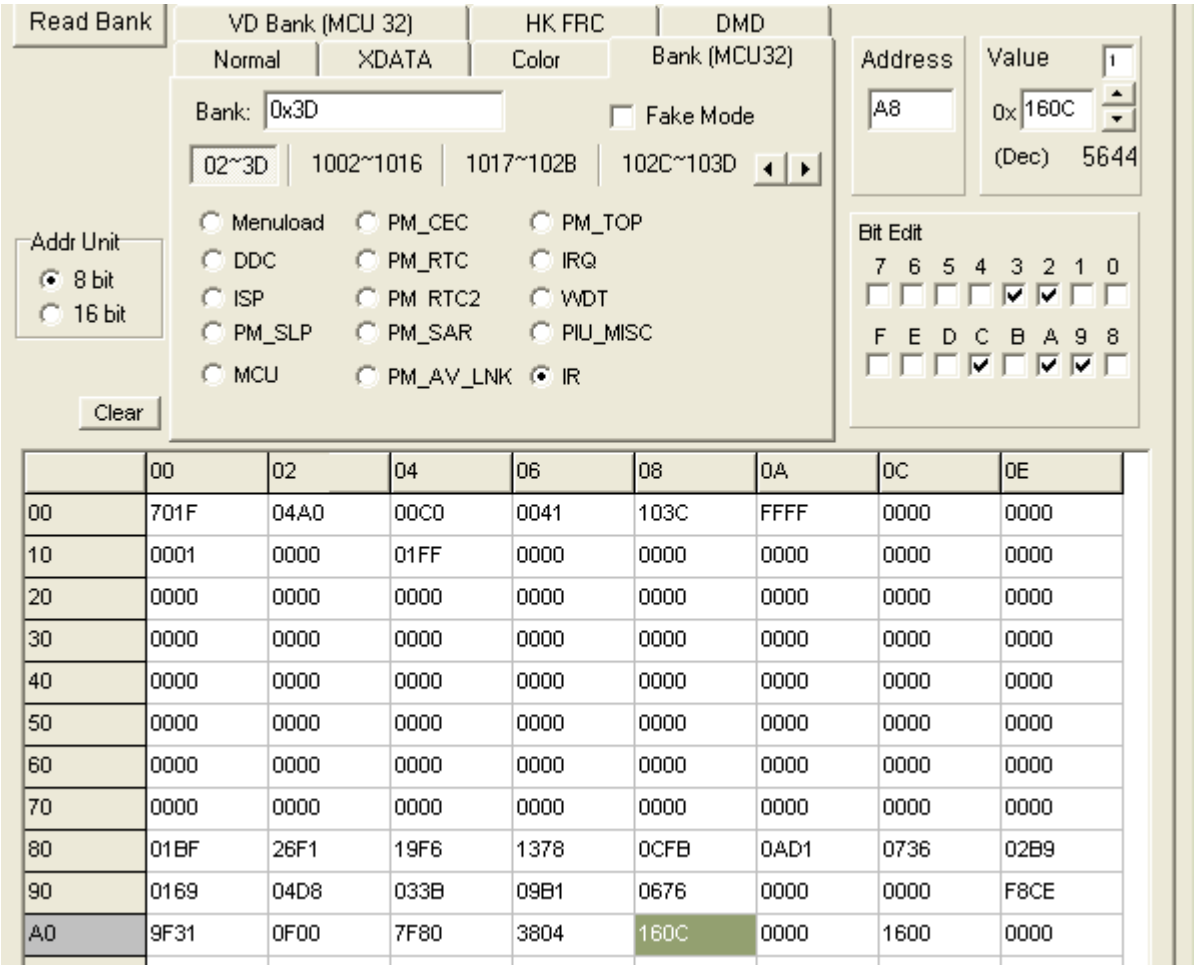
```
    IRKEY_AUDIO                 = 0x44,
    IRKEY_VOLUME_PLUS           = 0x16,
    IRKEY_VOLUME_MINUS          = 0x15,

    IRKEY_UP                    = 0x52,
    IRKEY_POWER                 = 0x46,
    IRKEY_EXIT                  = 0x1B,
    IRKEY_MENU                  = 0x07,
    IRKEY_DOWN                  = 0x13,
    IRKEY_LEFT                  = 0x06,
    IRKEY_SELECT                = 0x0F,
    IRKEY_RIGHT                 = 0x1A,
    ......
}IrCommandType;
```

4.1.4.2. 读取 IR 键值

IR 在 MSTV tool 中的 bank 值为 3D_A9，以 Kaiser 为例在 MBoot_STB_Kaiser/MstarCore/src/drivers/ir/kaiser/msIR.h 中定义了 IR 的 sub bank

```
#define IR_KEY                                  ( RIUBASE_IR + 0xA9 )   //0x3DA9
```

当 IR 按键 Vol+之后使用 MSTV tool 读取 3D_A9 值为 0x16。



	00	02	04	06	08	0A	0C	0E
00	701F	04A0	00C0	0041	103C	FFFF	0000	0000
10	0001	0000	01FF	0000	0000	0000	0000	0000
20	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
40	0000	0000	0000	0000	0000	0000	0000	0000
50	0000	0000	0000	0000	0000	0000	0000	0000
60	0000	0000	0000	0000	0000	0000	0000	0000
70	0000	0000	0000	0000	0000	0000	0000	0000
80	01BF	26F1	19F6	1378	0CFB	0AD1	0736	02B9
90	0169	04D8	033B	09B1	0676	0000	0000	F8CE
A0	9F31	0F00	7F80	3804	160C	0000	1600	0000

Figure 4-5: MSTV 读取 IR 键值

4.2. Keypad 模式

4.2.1. 使能 BOOT_KEYPAD 功能

在 make menuconfig 中将 BOOT IR 功能打开之后 mboot 才会通过获取 KEYPAD 的 ADC 值做相应的升级动作

具体步骤如下：

- make menuconfig
- 进入 Module Options

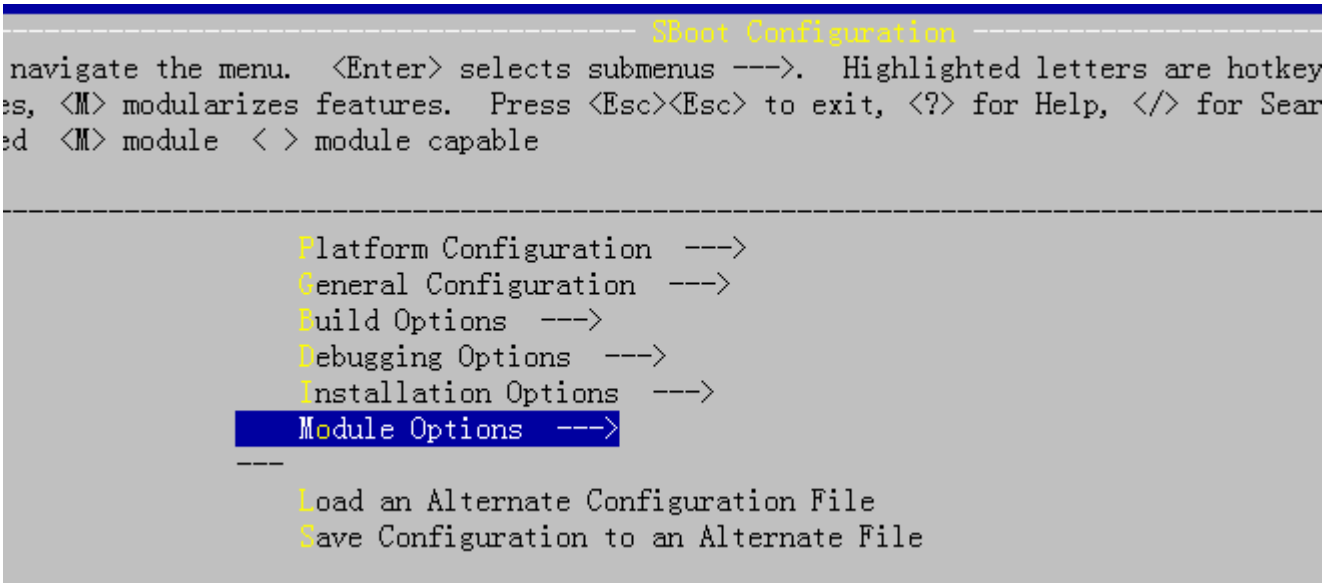


Figure 4-6: MBoot 使能 keypad-1

- 勾选 Keypad 并且勾选 Keypad 中的 BOOT_KEYPAD

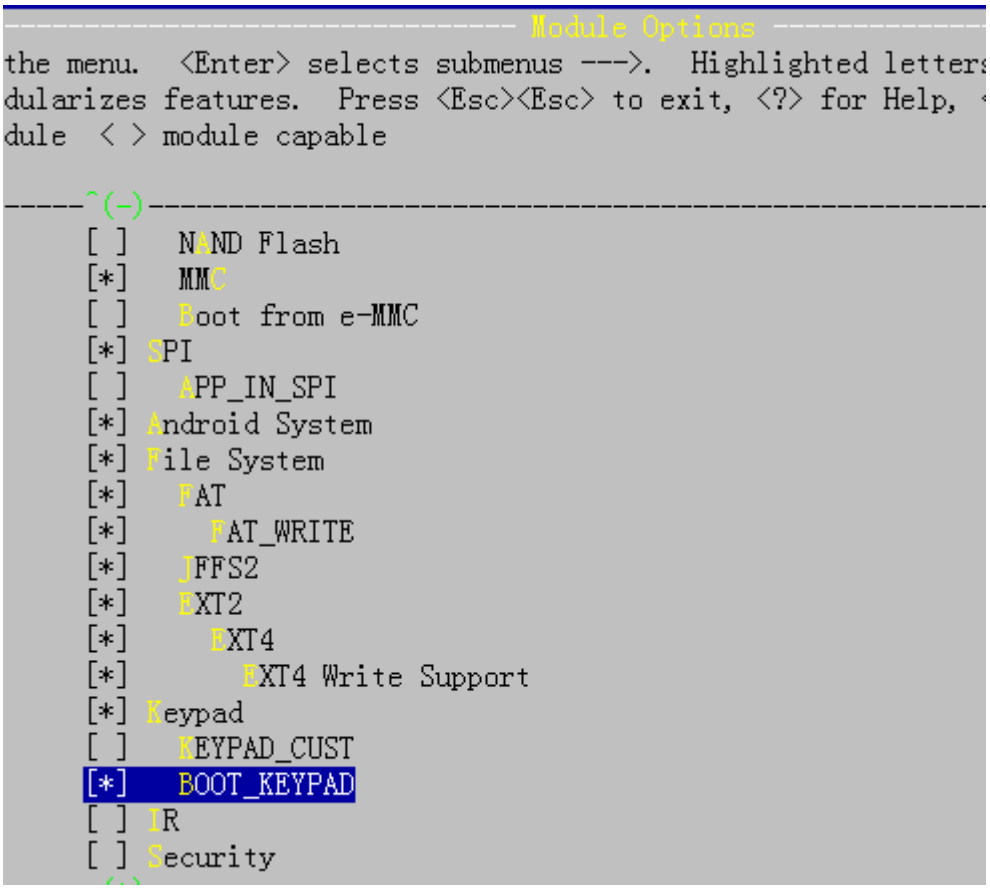


Figure 4-7: MBoot 使能 keypad-2

4.2.2. 工作原理

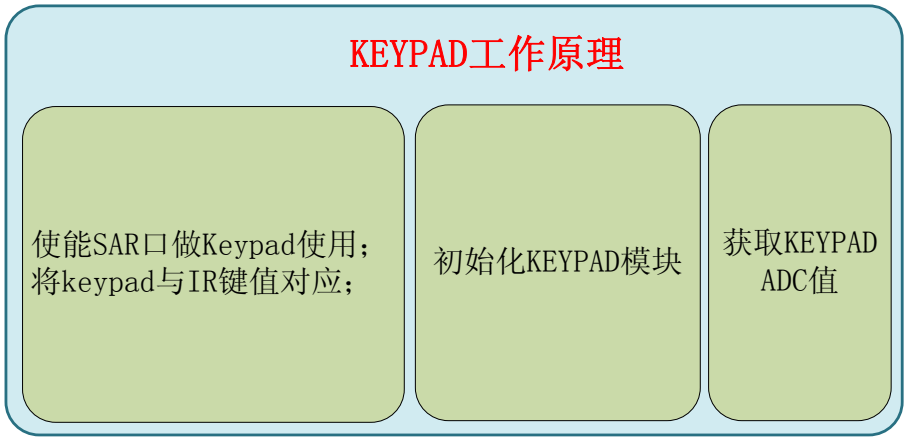


Figure 4-8: keypad 工作原理

1. 在 sboot 中配置 sar 口做 keypad 使用，并将 keypad 的值与 IR 键值进行对应
2. 初始化 KEYPAD 时会首先获取当前的 SAR 口是否做 keypad 使用，如果作为 keypad 使用会操作相应的寄存器使其作为一个模拟的输入：

```
//select pad as analog input
MDrv_WriteByte(REG_SAR_AISEL, MDrv_ReadByte(REG_SAR_AISEL)|(SAR_AISEL_CH0_MSK));
//select pad direction as input mode
MDrv_WriteByte(REG_SAR_GPIOOEN,
MDrv_ReadByte(REG_SAR_GPIOOEN)|(SAR_GPIOOEN_CH0_MSK));
```

4.2.3. 流程图

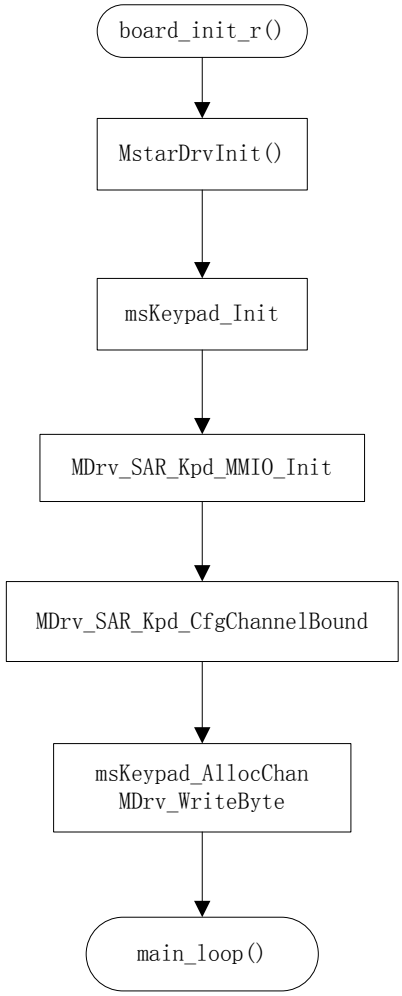


Figure 4-9: keypad 流程图

4.2.4. DEBUG

4.2.4.1. SAR 口定义

在 sboot/board/kaiser/BD_MST170A_D01A_SZ.h 中使能 SAR 口为 Keypad（K3S 为例）

```
#define ENABLE_KPDCHAN_1        ENABLE      //SAR0 定义为 keypad
#define ENABLE_KPDCHAN_2        DISABLE
#define ENABLE_KPDCHAN_3        DISABLE
#define ENABLE_KPDCHAN_4        ENABLE      //SAR3 定义为 keypad
```

4.2.4.2. keypad 与 IR 的对应

在 sboot/board/kaiser/BD_MST170A_D01A_SZ.h 定义 keypad 的键值与 IR 的对应（K3S 为例）

```
#define ADC_KEY_1_L0_FLAG           IRKEY_POWER
```

```
#define ADC_KEY_1_L1_FLAG          IRKEY_MENU
#define ADC_KEY_1_L2_FLAG          IRKEY_LEFT
#define ADC_KEY_1_L3_FLAG          IRKEY_MUTE

#define ADC_KEY_2_L0_FLAG          IRKEY_UP
#define ADC_KEY_2_L1_FLAG          IRKEY_INPUT_SOURCE
#define ADC_KEY_2_L2_FLAG          IRKEY_RIGHT
#define ADC_KEY_2_L3_FLAG          IRKEY_DOWN
```

- 前四个为 SAR0 支持的 4 个 keypad 按键值与 IR 的对应；
- 后四个为 SAR3 支持的 4 个 keypad 按键值与 IR 的对应；
- Keypad 的值是顺序识别；

4.2.4.3. 读取按键 keypad 值时的 ADC 的值

由于 keypad 的 ADC 值是由电压控制的，会有一个界定值的浮动，因此定义了一个边界值，在 ADC 值落在某个区间的时候都认为他是对应的同一个 IR 的值（sboot/board/kaiser/BD_MST170A_D01A_SZ.h）

```
#define LK_CH_MINUS_UB          0x11
#define LK_CH_MINUS_LB          0x0D
#define LK_CH_PLUS_UB           0x11
#define LK_CH_PLUS_LB           0x0D
#define LK_INPUT_UB             0x09
#define LK_INPUT_LB             0x05
#define LK_MENU_UB              0x09
#define LK_MENU_LB              0x05
#define LK_OK_UB                0x18
#define LK_OK_LB                0x14
#define LK_POWER_UB             0x03
#define LK_POWER_LB             0x00
#define LK_VOL_MINUS_UB         0x18
#define LK_VOL_MINUS_LB         0x14
#define LK_VOL_PLUS_UB          0x03
#define LK_VOL_PLUS_LB          0x00
```

以 power 键为例，ADC 落在 LK_POWER_UB 和 LK_POWER_LB 之间都对应 IR 的 power 键。

4.2.4.4. 读取 keypad 的 ADC 值

在 UTPA-17.0.x_Kaisers/mxlib/drv/sar/drvSAR.h 中定义了 SAR0 对应的寄存器的值 14_1A

#define REG_SAR_ADC_CH1_DATA ((0x00*0x10000)+ (SAR_REG_BASE + 0x0D*2))

按键板子上的 keypad 之后读取当前寄存器的值

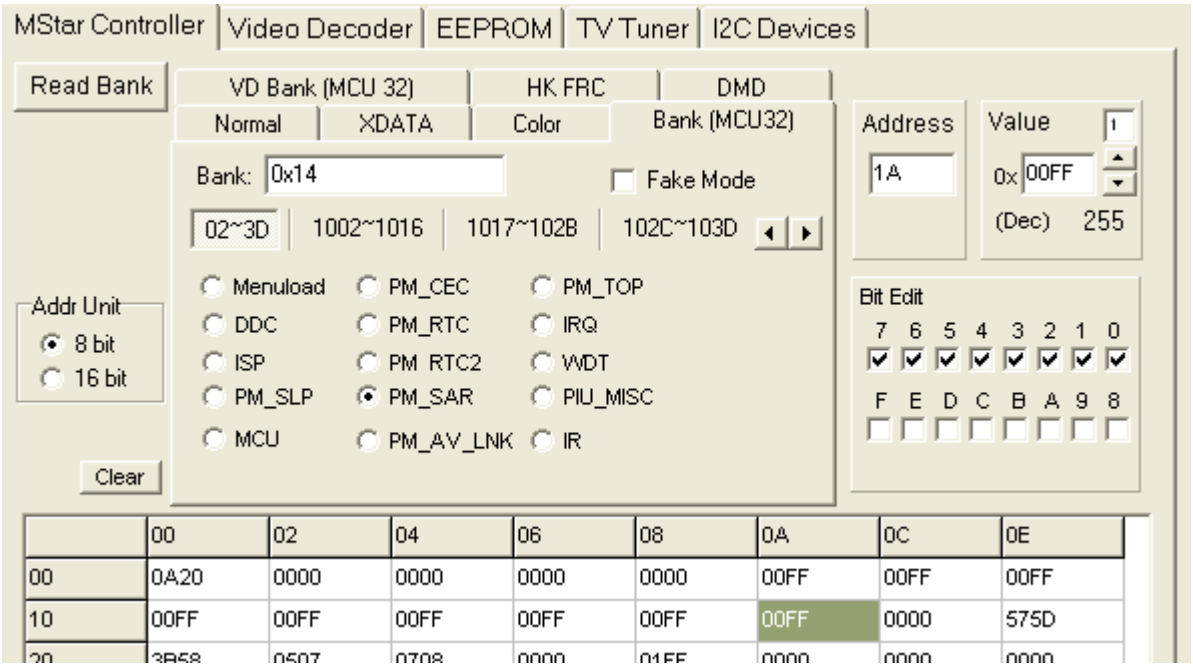


Figure 4-10: MSTV 读取 keypad ADC 值

通过 2.2.4.3 中定义的 ADC 区间以及 2.2.4.2 中定义的 flag 确认为此时按键做 power 键值使用，如果想开机按键进行 USB 升级则需要做如下的修正：

#define ADC_KEY_1_L0_FLAG IRKEY_POWER

改为

#define ADC_KEY_1_L0_FLAG IRKEY_VOLUME_PLUS

并且要修改一下位置

```
.u8PmWakeIR =
{    //IR wake-up key define
     IRKEY_POWER, 0xFF, 0xFF, 0xFF,
0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0XFF,
}
```

➔

```
.u8PmWakeIR =
{    //IR wake-up key define
     IRKEY_POWER, IRKEY_VOLUME_PLUS, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
     0xFF, 0xFF, 0xFF, 0xFF,
}
```

即可实现此功能

4.3. ENV 模式

4.3.1. 使能 BOOT_ENV 功能

BOOT_ENV 是在 code 中写死的，默认打开此功能（MstarApp/src/MsBoot.c）

#define ENABLE_MODULE_ENV_BOOT 1

4.4. 流程图

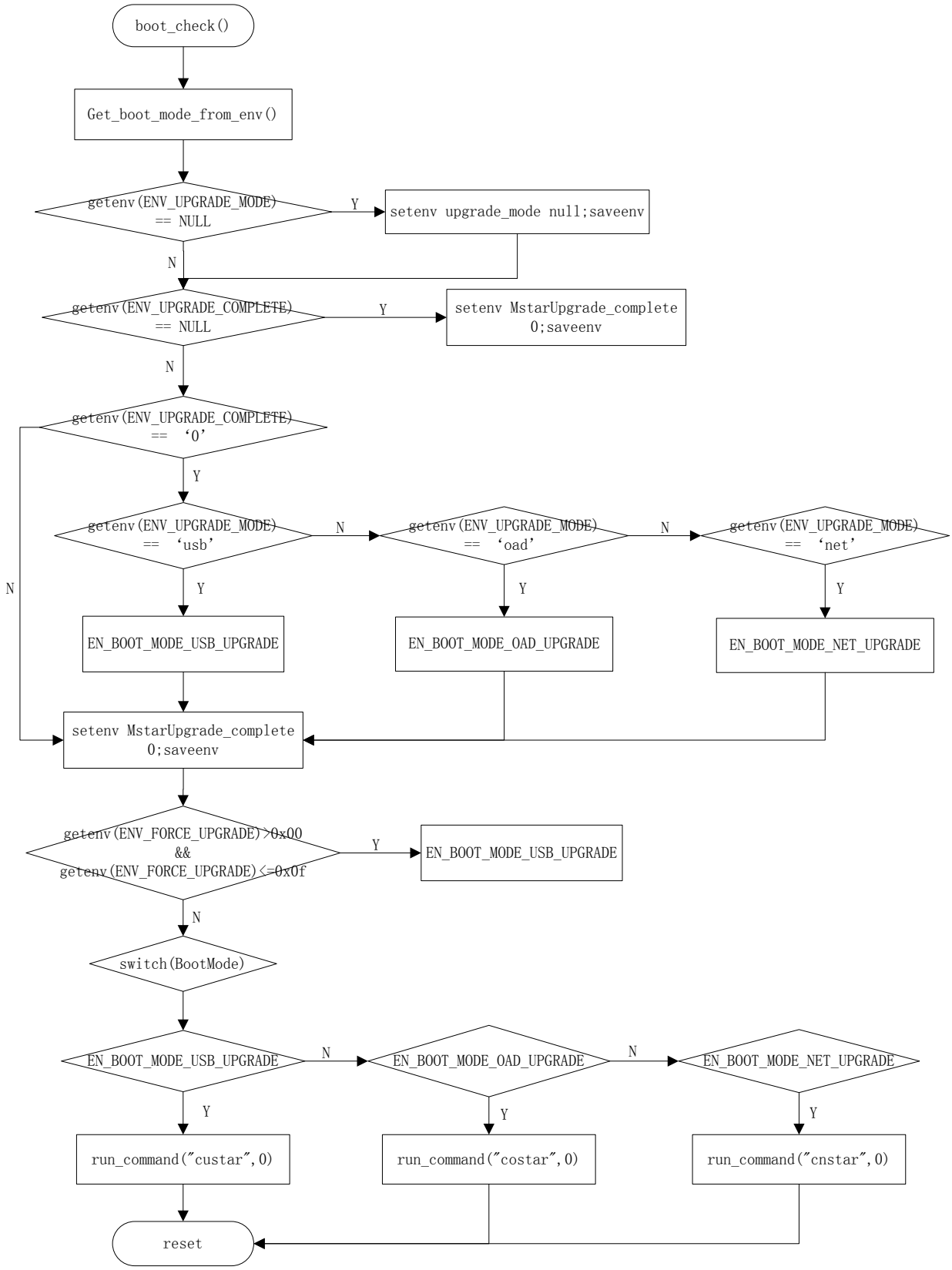


Figure 4-11: env 升级流程图

1. 进入 boot_check 函数之后会检测 MstarUpgrade_complete、upgrade_mode 以及 force_upgrade 三个环境变量的值；
2. upgrade_mode 用于判断是何种升级方式；
3. force_upgrade 的用于判断是否是升级中途就终止了升级（例断电模式）或者是同时存在几个升级包需要升级；
4. 如果 MstarUpgrade_complete = 0 则表示此时可以再次进行升级；如果 MstarUpgrade_complete = 1 则表示此时刚刚进行过升级，跳至第 7 步；
5. 当 upgrade_mode = usb 或者 force_upgrade 在 0x00-0x0F 中间时表示此时需要进行 USB 升级；
6. 升级完成后，重启系统；
7. 正常启动系统。

4.5. 开机自动升级

4.5.1. 使能开机自动升级

目前公版支持 AC / AC&DC 自动升级

在 make menuconfig 中将开机自动升级功能打开之后 mboot 才会通过获取当前开机的方式做相应的升级动作

具体步骤如下：

- make menuconfig
- 进入 Module Options

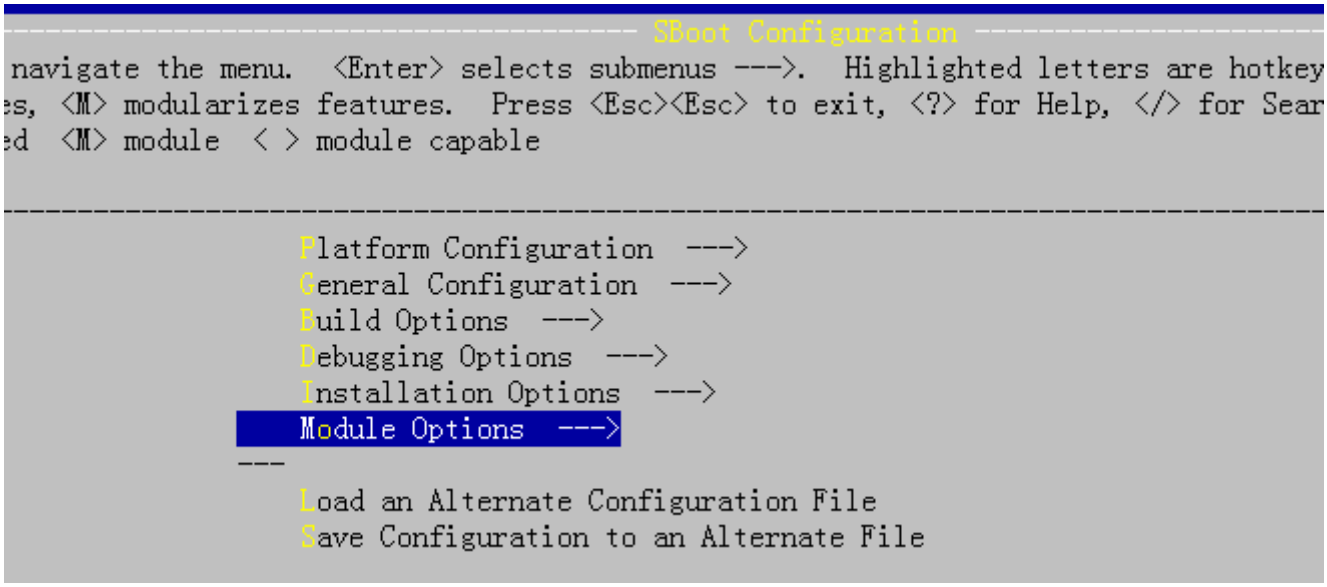


Figure 4-12: MBoot 使能开机自动升级-1

- 勾选 USB 并且勾选 USB 中的 AUTO_USB_UPGRADE

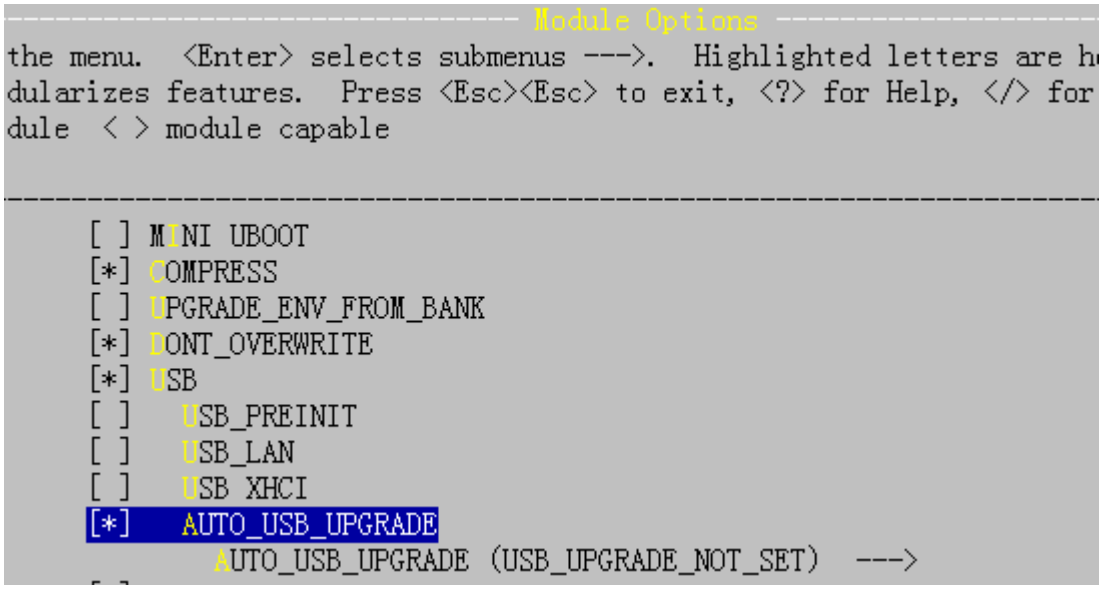


Figure 4-13: MBoot 使能开机自动升级-2

- 进入 AUTO_USB_UPGRADE，选择何种开机模式升级

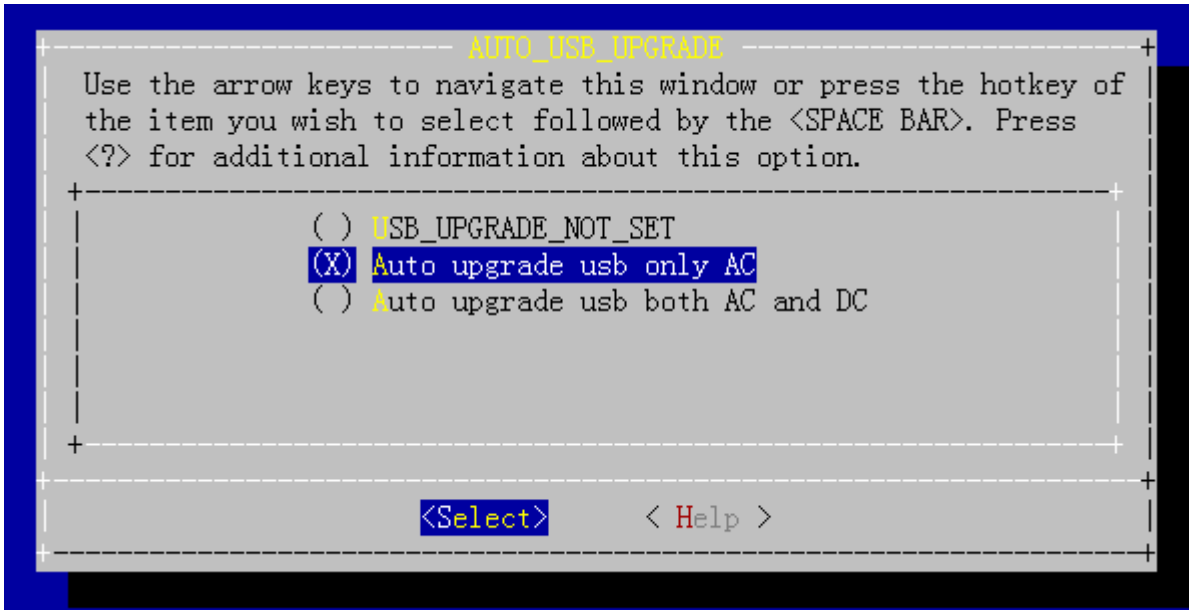


Figure 4-14: MBoot 使能开机自动升级-3

4.5.2. 流程图

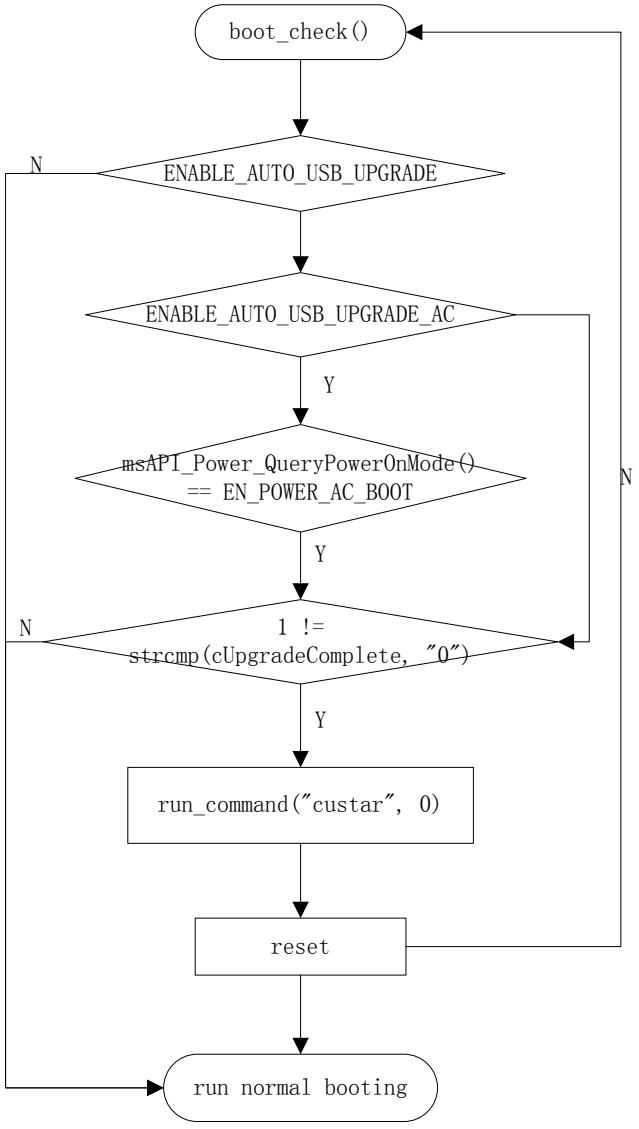


Figure 4-15:开机自动升级流程

1. 在 bootcheck 中检测是否打开 ENABLE_AUTO_USB_UPGRADE 开关，如果打开则继续执行第 2 步否则执行第 7 步；
2. 检测是否打开 ENABLE_AUTO_USB_UPGRADE_AC 开关，如果打开则代表只在 AC 上电时做 USB 升级，继续执行第 3 步；否则不论 AC 还是 DC 均进行 USB 升级，跳转至第 4 步；
3. 在打开 ENABLE_AUTO_USB_UPGRADE_AC 的情况下判断此时的开机模式是否为 AC 上电，如果为 AC 上电则执行第 4 步；否则执行第 7 步；
4. 检测此时的环境变量 MstarUpgrade_complete 的值是否为 1，如果为 1 则代表此时已经进行过 USB 升级且是刚刚升级完成时候的 AC 上电，则此时会执行第*步，否则会支持第 5 步；
5. 运行 custar 命令；
6. 系统重新启动，在重新从第 1 步进行检测；

7. 继续执行 bootcheck 的其他的命令，正常启动系统。

4.6. 串口模式

4.6.1. 串口进入 USB 升级的方法

- 电路板开机上电之后长按回车键，进入 mboot 控制台：
- 输入 custar，并按回车执行即可进入串口升级的模式；

4.6.2. 串口进入 OAD 升级的方法

- 电路板开机上电之后长按回车键，进入 mboot 控制台：
- 输入 costar，并按回车执行即可进入串口升级的模式；

5. CUSTAR 函数解析

custar 中主要包括以下三个部分：

- 检测 USB
- OSD 显示
- ustar 升级

5.1. Custar

5.1.1. 流程图

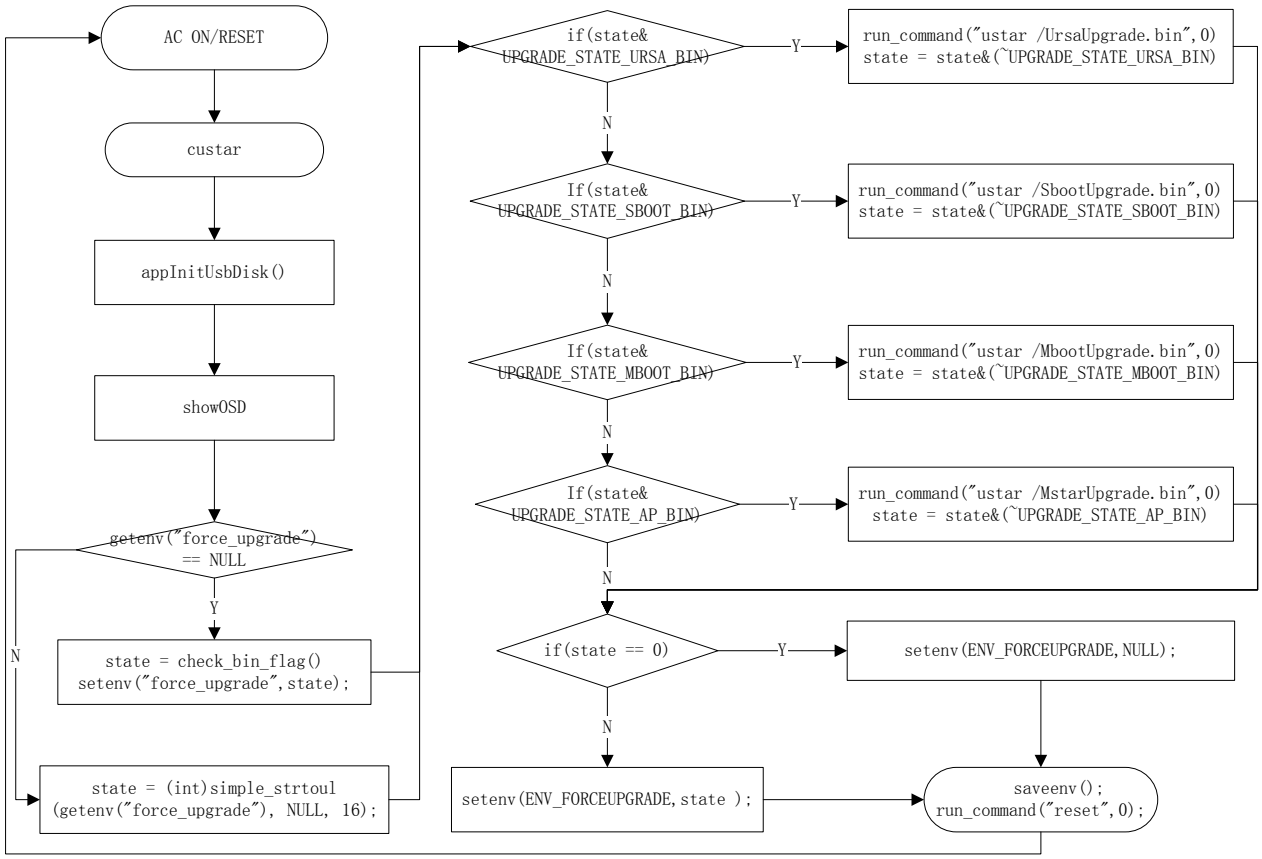


Figure 5-1:custar 流程

- custar 函数里面首先会检测 USB，此时不会修改到环境变量，检测不到 USB 之后会自动跳出这个函数，

启动正常系统；

- 检测到 USB 之后会判断此时的 OSD 开关是否打开，如果开关打开则需要显示 OSD；
- 获取 force_upgrade 这个环境变量的值，如果不为空，跳至第 5 步；
- 如果 force_upgrade 为 null，检测当前 U 盘根目录下有哪些升级包，并设置对应的 bit 值到 force_upgrade 中；
- 使用 ustar 进行升级，升级成功后消除对应的 bit 值；
- 如果 U 盘中存在多种升级包，在升级完一个升级包之后，再次开机的时候 force_upgrade 中还有其他升级包对应的 bit 值，此时会继续升级其他升级包，直至所有升级包升级完毕，启动正常模式；

5.2. USB

Mboot 中检测 USB 有如下限制：

- 只检测 USB 的主分区中是否含有 USB 升级包；
- 支持 USB2.0 口，不支持 USB3.0 口；
- 只支持 FAT32 格式的 U 盘，不支持 EXT4 以及 NTFS 格式的移动存储设备；

5.3. OSD 显示

5.3.1. 使能显示 OSD

目前公版支持 USB 升级时显示进度条。

在 make menuconfig 中将 OSD 功能打开之后，mboot 才会通过在升级时显示进入条。

具体步骤如下：

- make menuconfig
- 进入 Module Options

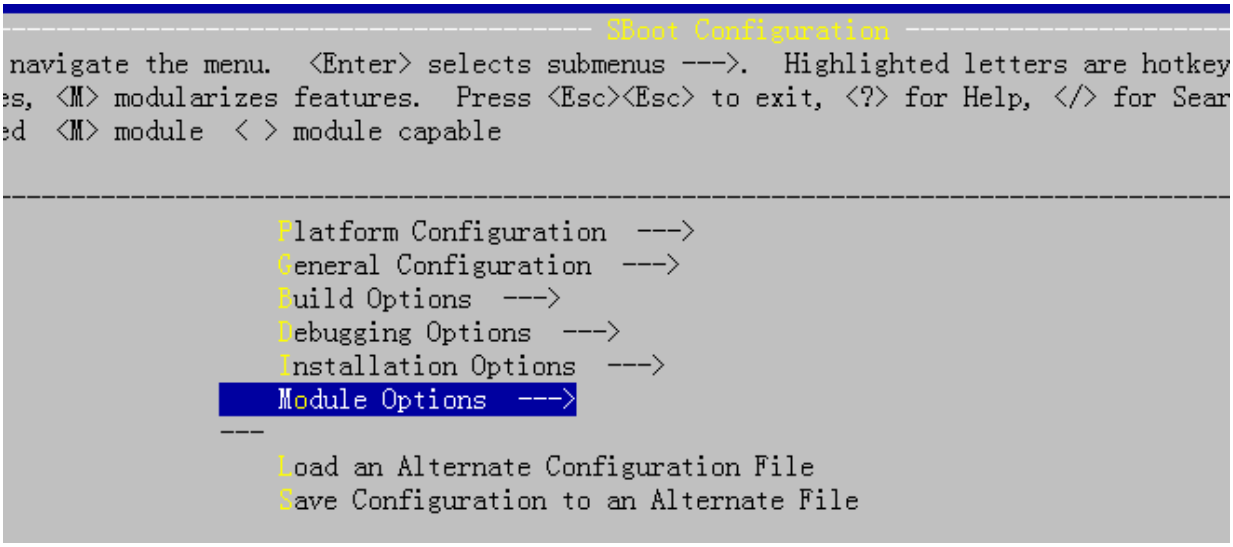


Figure 5-2:MBoot 使能 OSD-1

- 勾选 DISPLAYOSD

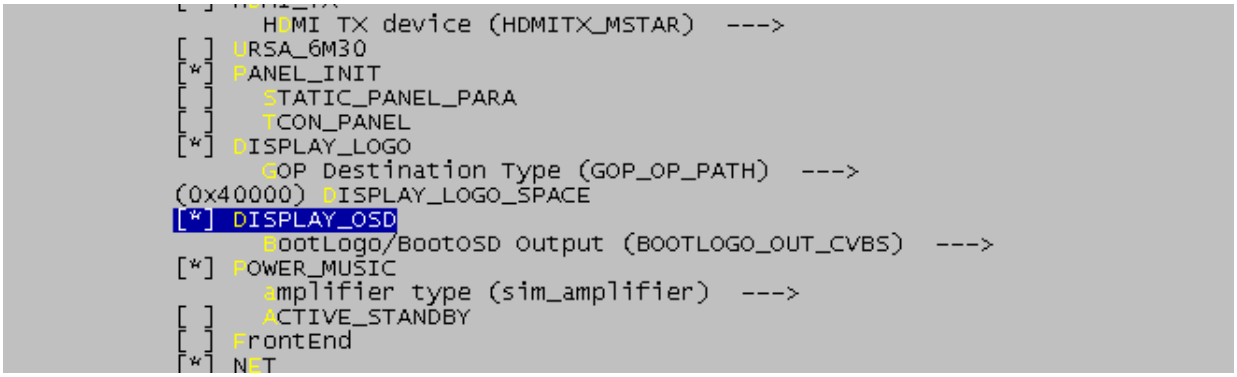


Figure 5-3:MBoot 使能 OSD-2

- 效果图如下：

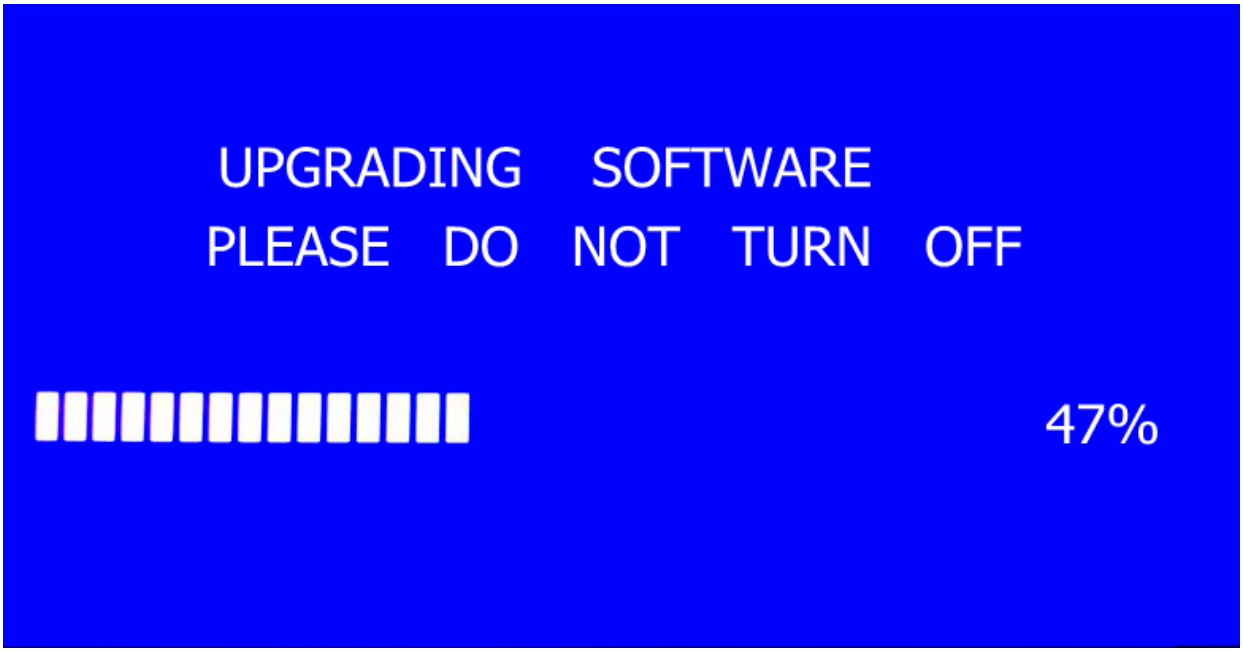


Figure 5-4:OSD 升级效果图

5.3.2. OSD 流程

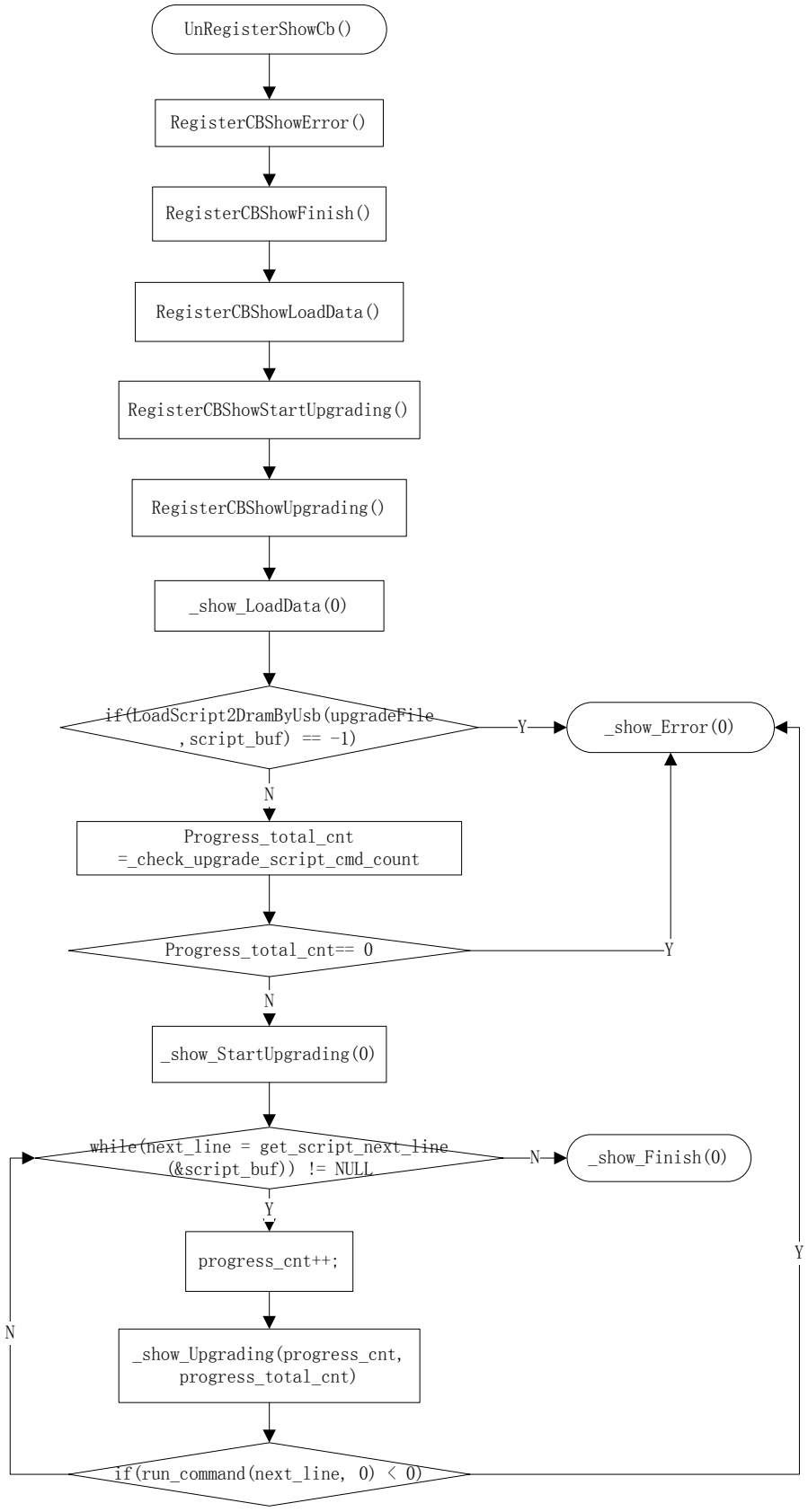


Figure 5-5:OSD 流程

1. 在 custar 指令中注册函数指针做初始化动作
2. 在 ustar 中调用_show_LoadData 函数画图显示 OSD 开始 load USB 中的数据如下图：

Figure 5-6:OSD LOAD DATA

3. 在 load 数据过后获取当前 USB 升级脚本中共有多少条语句需要执行，目的为更新 OSD 进度条的时候计算百分比；
4. Load 数据失败或者执行语句的个数为 0 的时候则显示软件升级失败，跳出升级过程，如下图：

Figure 5-7:OSD UPGRADE ERROR

5. 开始更新刷新显示进度条，进入升级过程；
6. 每执行一个语句之后会更新一次当前的 OSD 进度条；

7. 如果当前指令运行失败，则显示如第 4 步，否则如第 6 步一样，更新 OSD 进度；
8. 获取到最后一条进度的时候会显示升级完成，如下图：

Figure 5-8:OSD UPGRADE COMPLETE

5.3.2.1 osd_create

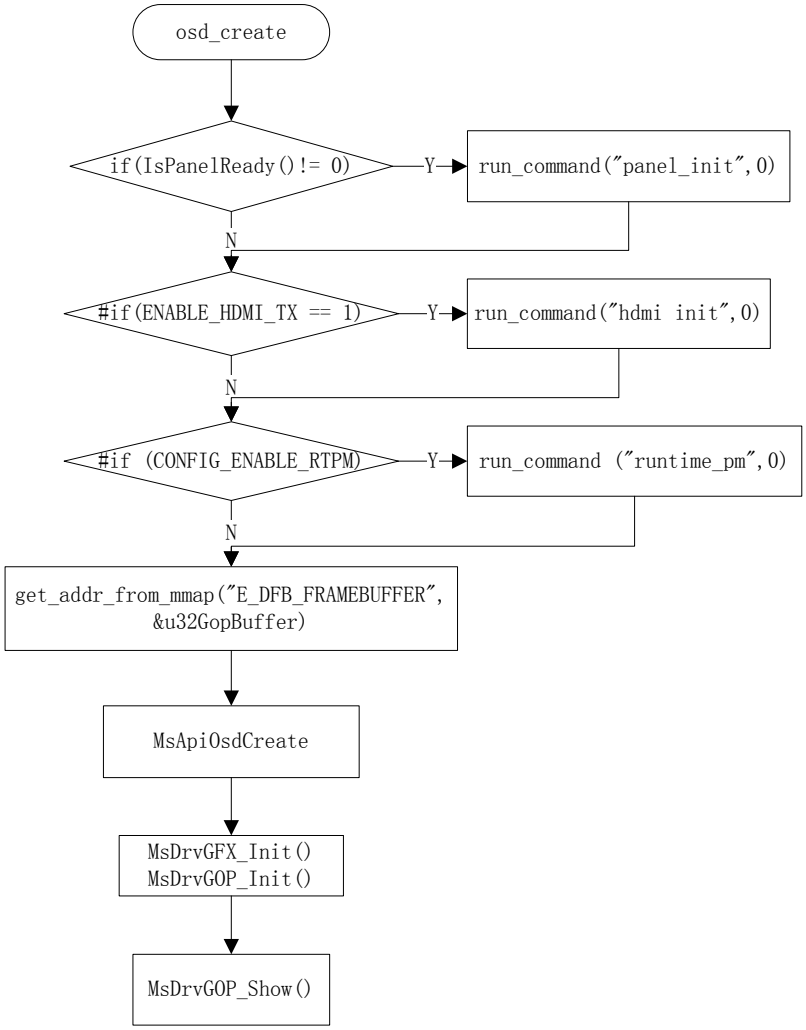


Figure 5-9:OSD create 流程

- 检测 panel 是否已经初始化，如果没有初始化，则需要运行 panel_init，点屏与背光；
- 如果有开启 HDMITX，则需要做 HDMI init；
- 如果有开启 RTPM，则需要做 runtime_pm；
- 获取 DFB 的地址做 OSD 画图区域；
- 初始化 GFX 与初始化 GOP；
- 设置 GOP 显示的地址，以及显示格式等相关信息；

5.3.2.2 draw_rect

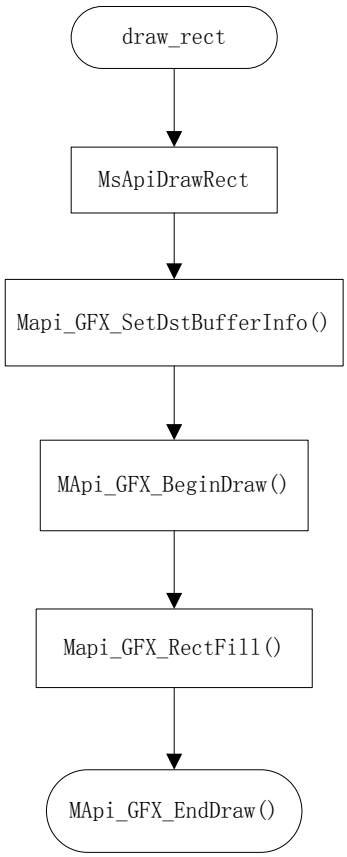


Figure 5-10:draw_rect 流程

- MApi_GFX_SetDstBufferInfo 设置 GFX 画图的目的地址 CANVAS 的信息给 GFX;
- 通过设定 GFX 得到的需要画图的长、宽、颜色等信息透过 MApi_GFX_RectFill 来进行画图;

5.3.2.3 draw_string

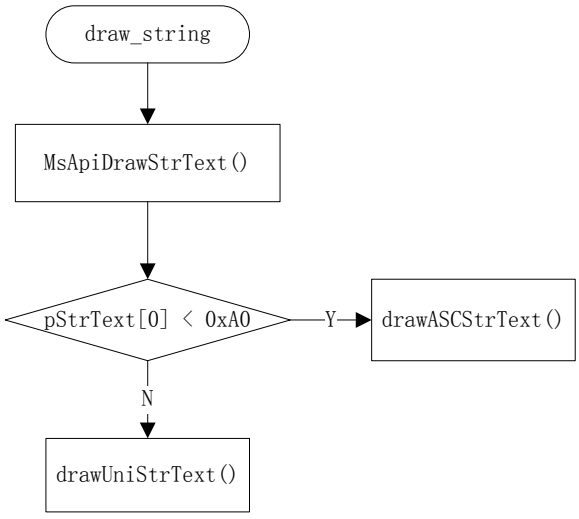


Figure 5-11:draw_string 流程

- 字符串显示的时候可以显示简单的中文和英文；
- drawASCStrText()显示英文；
- drawUniStrText()显示中文；

5.3.2.4 draw_progress

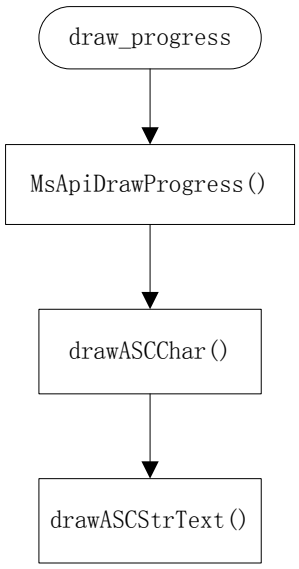


Figure 5-12:draw_process 流程

- drawASCChar 用来画进度条的白色小方块；
- drawASCStrText 用来画当前进度的百分比；

5.3.2.5 osd_flush

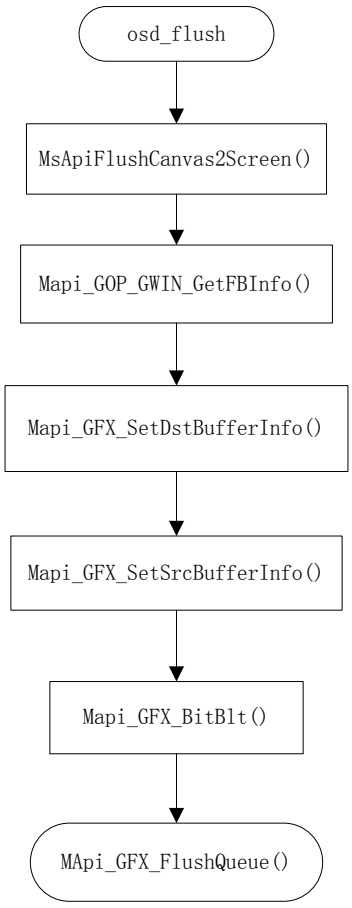


Figure 5-13:osd_flush 流程

- MApi_GFX_SetDstBufferInfo 获取 osd_create 中在 DFB 上创建 GWIN 的相关信息到 GFX 中，设置其为显示的目的地址；
- MApi_GFX_SetSrcBufferInfo 设置 CANVAS 的信息到 GFX 中，因为相关画图的地址为 CANVAS；
- MApi_GFX_BitBlt 设置画图的大小；
- MApi_GFX_FlushQueue 从画图的地址刷新到 DFB 上，此时就可以显示到屏幕上；

5.4. ustar

5.4.1. 升级流程

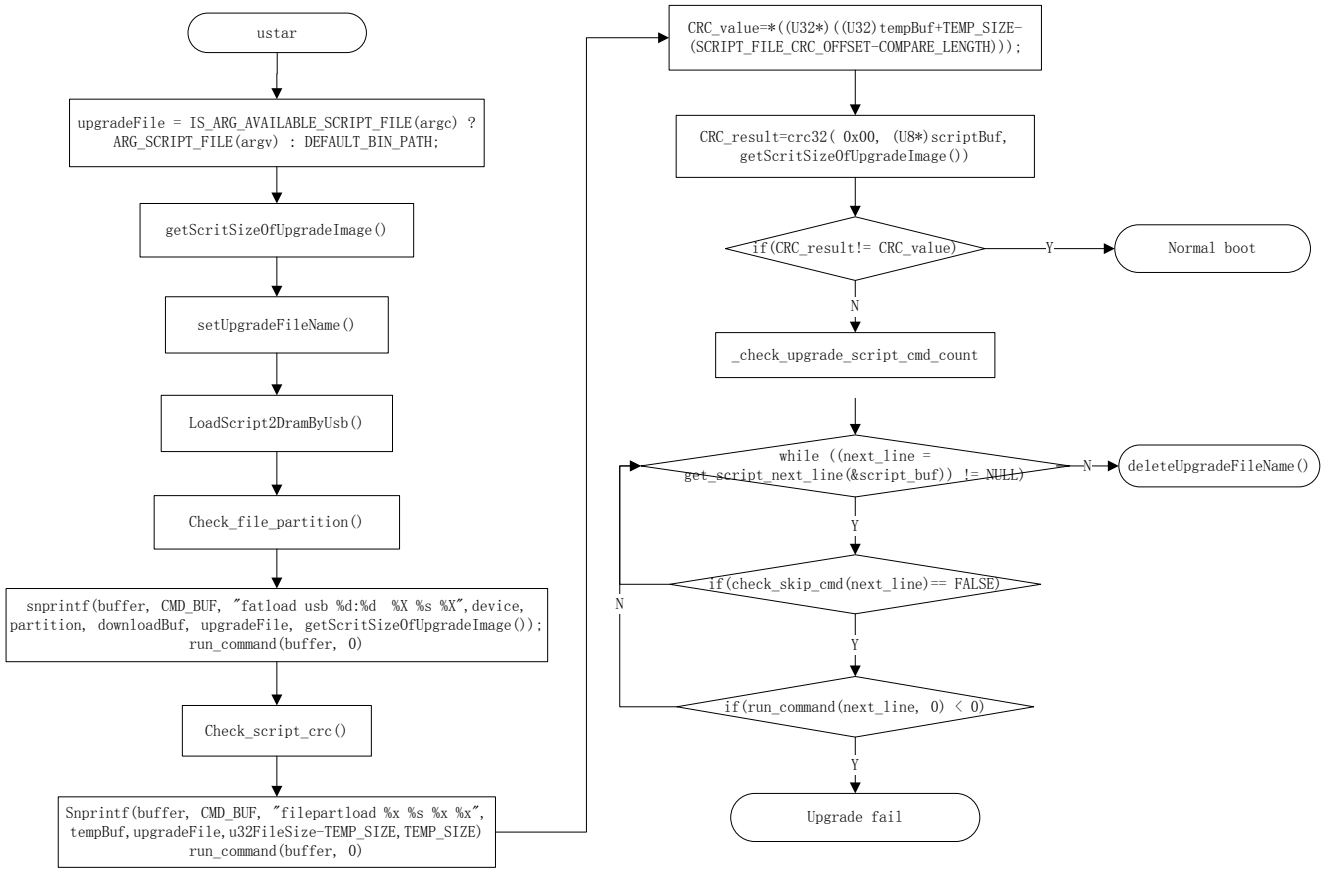


Figure 5-14:ustar 流程

1，进入 ustar 命令之后会首先判断传入的升级包名字是否是可用的，如果是不可用的则默认设置为 MstarUpgrade.bin；

2，获取升级脚本的大小 0x4000；

3，将升级包的名字设置至 env 中的 UpgradeImage 中；

4，将升级脚本的内容全部 load 至内存中，并计算其 crc 的值与制作升级包时对脚本部分计算的 crc 值是否相同，如果相同则继续执行第 5 步，否则启动到正常系统中；

5，计算脚本中的指令一共有多少个，这个主要和 OSD 显示时的进度条有关；

6，获取脚本中的每条升级命令，首先判断是不是要跳过的指令，reset 指令以及 nand 中在 ENABLE_MODULE_DONT_OVERWRITE 开关打开的时候 ubi remove 的指令也会被跳过，在指令中存在#的时候会将#之后的字符去掉，只去#前面的进行执行；

7，run_command 第 6 步解析出来的指令，如果有一句返回值为-1，则表示本次升级失败，等同于升级过程中断电，在此开机时会再次进入 env 升级的流程；

8，解析完全部的脚本并且每一句指令都执行成功之后代表本次 ustar 命令已经执行成功，会将 env 中的 UpgradeImage 删除。

6. COSTAR 函数解析

6.1. costar 流程

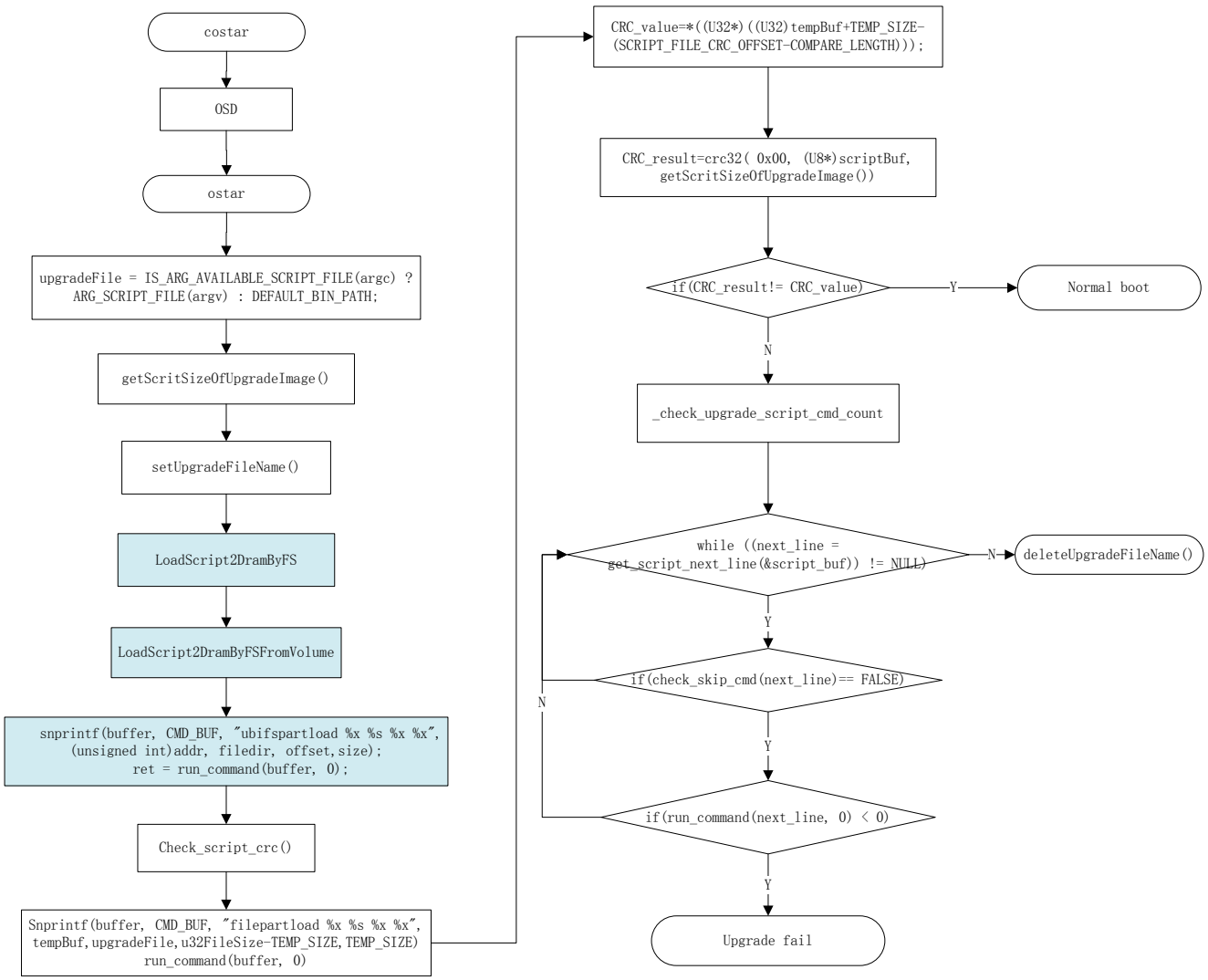


Figure 5-15:costar 流程

- costar 会首先判断此时的 OSD 开关是否打开，如果开关打开则需要显示 OSD；
- 运行 ostar 进行 OAD 升级；
- 挂载 OAD 分区，从 OAD 分区中 load OAD 升级包的执行脚本，检测 script CRC 是否一致；
- 运行升级脚本中的命令更新分区；
- 升级完毕，启动正常模式；

7. 示例分析

7.1. 分类

将 USB 升级时所有问题分类如下：

- 和 OSD 相关内容
- 和 U 盘相关内容
- nand 特有问题

7.2. 和 OSD 相关内容

7.2.1. 0523091: [Napoli][051D] FHD 平台上 MBoot 打开 OSD 后，USB 升级失败

7.2.1.1. Rootcase

打开 OSD 的时候，load system 镜像的时候有踩到 CANVAS_BUFFER_ADDR 的地址。

7.2.1.2. Solution

将 CANVAS_BUFFER_ADDR 的地址向后移，确保不会影响到 USB 升级。

7.2.1.3. Debug 思路

由 TV 的平台与以下地址有关：

- CANVAS_BUFFER_ADDR是create OSD的时候使用的地址；
- DRAM_FATLOAD_BUF_ADDR是USB升级是load分区镜像的地址；
- Box的平台还与DFB的地址有关；

  因为 box 会涉及到 CVBS 的显示，因此比 TV 多了一部分处理，就会用到 DFB 的地址；

  地址计算方法如下：

  所用空间最大的就是 load system 分区，system 分区切割一般会小于 270M，因此需要留够大约 270M 的内存，以 napoli 为例：

- DRAM_FATLOAD_BUF_ADDR = 0x20400000，物理地址为 0x20400000 - 0x20000000 = 0x400000；

- CANVAS_BUFFER_ADDR = 0x06E00000，两个地址之间的大小为 0x6E00000 – 0x400000 = 0x6A00000（106M），而 system 分区切割之后的大小一般为 270M，压缩后的大小一定比 106M 大；
- 将CANVAS_BUFFER_ADDR改为0x14000000 ,此时两个地址之间的大小为316M ,完全可以容纳下system分区切割后的大小，因此该地址是满足 USB 升级要求。

对于 K3S 来说还需要考虑 DFB 的地址

- DRAM_FATLOAD_BUF_ADDR = 0x30000000，物理地址为 0x30000000 – 0x20000000 = 0x10000000；
- E_DFB_FRAMEBUFFER_ADR = 0x001694D000；两者间的地址大小为 105M，也小于了 system 切割的大小，因此有两种方法避免这个问题，一个是将 system 分区切割小于 105M，一个是将使用更后面的地址。

7.2.2. 0531081: 烧写第一次会出现烧写失败，重启自动继续烧写就 ok 了(usb 升级)

7.2.2.1. Rootcase

错误的量产方式，在空板中烧写 mboot 之后由其自动运行到控制台之后在执行 custar 进行升级，此时以及运行了 showlogo，因此踩到 DFB 的地址，导致 out of memory。

7.2.2.2. Solution

- 建议客户使用 AC ON 的方式进行升级；
- 烧写 mboot 之后重启的过程中长按 enter 键进入控制台之后在运行 custar。

7.2.2.3. Debug 思路

- 其 mboot 自动运行之间就已经显示 logo，表示已经使用到了 DFB 的地址；
- 在升级期间发现 logo 会花屏，表示此时是踩到了 DFB 的地址，不符合正常的升级流程；
- 第二次开机升级正常是因为在正常的流程中是先走到 bootcheck 中进行 USB 升级，logo 是在后面才显示。

7.2.3. 0460570: [NikeU][Konka][U disk upgrade]没有升级主程序(或手动擦除 emmc)，ISP tool 烧 MBoot（all chip）后，进行 U 盘升级的操作，会失败

7.2.3.1. Rootcase

在空的 mmc 上执行 OSD 显示会出错，因为此时无法获取屏参等相关信息。

7.2.3.2. Solution

在显示 OSD 的时候判断 panel_init 以及 hdmi init 运行是否正常，如果执行失败，则不会显示 OSD 进度条。

7.2.3.3. Debug 思路

- 首先想到的是通过判断存放屏参分区是否存在来判断是否显示 OSD，但是发现有漏洞是如果是创建分区之后但是还没有升级存放屏参分区之前断电，这种情况是 cover 不到的，因此这种解决方式不成立；
- 设置一个全局变量，在 panel_init 以及 hdmi init 执行失败时，每次更新或者创建 OSD 的时候会首先判断这个全局变量是否满足条件，如果不满足条件则不会执行跟 OSD 相关的一些函数，此时可以照顾到裸板以及屏参获取失败的情况。

7.3. 和 U 盘相关内容

7.3.1. 0502205: [TCL][Nikon][Mboot]部分 U 盘无法进行 USB 升级

7.3.1.1. Rootcase

该 U 盘是 NTFS 格式的

7.3.1.2. Solution

USB 升级时有以下限制：

- 只支持 FAT32 格式的 U 盘，不支持移动硬盘、NTFS 格式的 U 盘以及 EXT4 格式的 U 盘；
- 只能识别 USB2.0 接口，不识别 USB3.0；
- 只支持主分区升级；

7.4. nand 特有问题

7.4.1. 0482351: [创维][DTMB]部分 U 盘无法升级程序

7.4.1.1. Rootcase

由于 nand driver 限制，升级时需要切换当前 mtd 分区。

7.4.1.2. Solution

在制作升级包脚本的时候，在创建 volume 的时候制定工作的 mtd 分区。

7.4.2. 0482180: [创维][DTMB]酷开 U 盘无法升级

7.4.2.1. Rootcase

创维的 U 盘存在两个分区，且主分区不能正常使用，升级包存放的是逻辑分区。

7.4.2.2. Solution

修改 usb.h 里的 USB_MAX_STOR_DEV = 4，设置最多一个 U 盘支持 4 个分区。

7.4.3. 0537060: [Skyworth][Nugget][ISDB 256M]烧完 MBoot 后第一次使用 USB 升级不成功

7.4.3.1. Rootcase

- 由于此时是在裸板的基础上烧写了 mboot，还没有创建分区；
- nand 在制作 MstarUpgrade.bin 的时候，在创建每个分区的之前都会先做 ubi remove partition 的操作；
- 此时进行 USB 升级就会运行 ubi remove partition 失败，导致整个升级失败。

7.4.3.2. Solution

执行 ubi remove partition 失败时不返回-1，当做执行成功处理；这种情况也使用与 nand 新增分区时的情况。