Linux A/B System

发布版本：1.0

作者邮箱：jason.zhu@rock-chips.com

日期：2019.01

文件密级：公开资料

前言

概述

Linux A/B System介绍。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

产品版本

修订记录

日期	版本	作者	修改说明
2019-01-25	V1.0	Jason Zhu	初始版本

---

1 引用参考

《Rockchip-Secure-Boot2.0.md》

《Rockchip-Secure-Boot-Application-Note.md》

《Android Verified Boot 2.0》

2 术语

3 简介

所谓的A/B System即把系统固件分为两份，系统可以从其中的一个slot上启动。当一份启动失败后可以从另一份启动，同时升级时可以直接将固件拷贝到另一个slot上而无需进入系统升级模式。

4 AB数据格式及存储

存储位置为misc分区偏移2KB位置。

```
/* Magic for the A/B struct when serialized. */
#define AVB_AB_MAGIC "\0AB0"
#define AVB_AB_MAGIC_LEN 4

/* Versioning for the on-disk A/B metadata - keep in sync with avbtool. */
#define AVB_AB_MAJOR_VERSION 1
#define AVB_AB_MINOR_VERSION 0

/* Size of AvbABData struct. */
#define AVB_AB_DATA_SIZE 32

/* Maximum values for slot data */
#define AVB_AB_MAX_PRIORITY 15
#define AVB_AB_MAX_TRIES_REMAINING 7

typedef struct AvbABSlotData {
  /* Slot priority. Valid values range from 0 to AVB_AB_MAX_PRIORITY,
   * both inclusive with 1 being the lowest and AVB_AB_MAX_PRIORITY
   * being the highest. The special value 0 is used to indicate the
   * slot is unbootable.
   */
  uint8_t priority;

  /* Number of times left attempting to boot this slot ranging from 0
   * to AVB_AB_MAX_TRIES_REMAINING.
   */
  uint8_t tries_remaining;
```

```
  /* Non-zero if this slot has booted successfully, 0 otherwise. */
  uint8_t successful_boot;

  /* Reserved for future use. */
  uint8_t reserved[1];
} AVB_ATTR_PACKED AvbABSlotData;

/* Struct used for recording A/B metadata.
 *
 * When serialized, data is stored in network byte-order.
 */
typedef struct AvbABData {
  /* Magic number used for identification - see AVB_AB_MAGIC. */
  uint8_t magic[AVB_AB_MAGIC_LEN];

  /* Version of on-disk struct - see AVB_AB_{MAJOR,MINOR}_VERSION. */
  uint8_t version_major;
  uint8_t version_minor;

  /* Padding to ensure |slots| field start eight bytes in. */
  uint8_t reserved1[2];

  /* Per-slot metadata. */
  AvbABSlotData slots[2];

  /* Reserved for future use. */
  uint8_t reserved2[12];

  /* CRC32 of all 28 bytes preceding this field. */
  uint32_t crc32;
} AVB_ATTR_PACKED AvbABData;
```

对于小容量存储，没有misc分区，有vendor分区，可以考虑存储到vendor。

在此基础上增加lastboot，标记最后一个可启动固件。主要应用于低电情况或工厂生产测试时retry次数用完，而还没有进入系统调用boot_ctrl服务。

参考如下：

```
typedef struct AvbABData {
  /* Magic number used for identification - see AVB_AB_MAGIC. */
  uint8_t magic[AVB_AB_MAGIC_LEN];

  /* Version of on-disk struct - see AVB_AB_{MAJOR,MINOR}_VERSION. */
  uint8_t version_major;
  uint8_t version_minor;

  /* Padding to ensure |slots| field start eight bytes in. */
  uint8_t reserved1[2];

  /* Per-slot metadata. */
  AvbABSlotData slots[2];
```

```
  /* mark last boot slot */
  uint8_t last_boot;
  /* Reserved for future use. */
  uint8_t reserved2[11];

  /* CRC32 of all 28 bytes preceding this field. */
  uint32_t crc32;
} AVB_ATTR_PACKED AvbABData;
```

同时在AvbABSlotData中增加is_update标志位，标志系统升级的状态，更改如下：

```
typedef struct AvbABSlotData {
  /* Slot priority. Valid values range from 0 to AVB_AB_MAX_PRIORITY,
   * both inclusive with 1 being the lowest and AVB_AB_MAX_PRIORITY
   * being the highest. The special value 0 is used to indicate the
   * slot is unbootable.
   */
  uint8_t priority;

  /* Number of times left attempting to boot this slot ranging from 0
   * to AVB_AB_MAX_TRIES_REMAINING.
   */
  uint8_t tries_remaining;

  /* Non-zero if this slot has booted successfully, 0 otherwise. */
  uint8_t successful_boot;

  /* Mark update state, mark 1 if the slot is in update state, 0 otherwise. */
  uint8_t is_update : 1;
  /* Reserved for future use. */
  uint8_t reserved : 7;
} AVB_ATTR_PACKED AvbABSlotData;
```

最后表格来说明各个参数的含义：

AvbABData：

参数	含义
priority	标志slot优先级，0为不可启动，15为最高优先级
tries_remaining	尝试启动次数，设置为7次
successful_boot	系统启动成功后会配置该参数，1：该slot成功启动，0：该slot未成功启动
is_update	标记该slot的升级状态，1：该slot正在升级，0：该slot未升级或升级成功

AvbABSlotData：

参数	含义
magic	结构体头部信息：\0AB0
version_major	主版本信息
version_minor	次版本信息
slots	slot引导信息，参见AvbABData
last_boot	上一次成功启动的slot，0：slot A上次成功启动，1：slot B上次成功启动
crc32	数据校验

5 启用配置

5.1 pre-loader说明

目前pre-loader支持A/B slot分区和单slot分区。

5.2 uboot配置

```
CONFIG_AVB_LIBAVB=y
CONFIG_AVB_LIBAVB_AB=y
CONFIG_AVB_LIBAVB_ATX=y
CONFIG_AVB_LIBAVB_USER=y
CONFIG_RK_AVB_LIBAVB_USER=y
CONFIG_ANDROID_AB=y
```

5.2 system bootctrl参考

目前system bootctrl设计两套控制逻辑，bootloader全支持这两种逻辑启动。

5.2.1 successful_boot模式

正常进入系统后，boot_ctrl依据androidboot.slot_suffix，设置当前slot的变量：

```
successful_boot = 1;
priority = 15;
tries_remaining = 0;
is_update = 0;
last_boot = 0 or 1;     :refer to androidboot.slot_suffix
```

升级系统中，boot_ctrl设置：

```
升级的slot设置：
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 1;
lastboot = 0 or 1;     :refer to androidboot.slot_suffix

当前slot设置：
successful_boot = 1;
priority = 15;
tries_remaining = 0;
is_update = 0;
last_boot = 0 or 1;     :refer to androidboot.slot_suffix
```

升级系统完成，boot_ctrl设置：

```
升级的slot设置：
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
lastboot = 0 or 1;     :refer to androidboot.slot_suffix

当前slot设置：
successful_boot = 1;
priority = 14;
tries_remaining = 0;
is_update = 0;
last_boot = 0 or 1;     :refer to androidboot.slot_suffix
```

5.2.2 reset retry模式

正常进入系统后，boot_ctrl依据androidboot.slot_suffix，设置当前slot的变量：

```
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
last_boot = 0 or 1;     :refer to androidboot.slot_suffix
```

升级系统中，boot_ctrl设置：

```
升级的slot设置：
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 1;
lastboot = 0 or 1;     :refer to androidboot.slot_suffix

当前slot设置：
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
last_boot = 0 or 1;     :refer to androidboot.slot_suffix
```

升级系统完成，boot_ctrl设置：

```
升级的slot设置：
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
lastboot = 0 or 1;     :refer to androidboot.slot_suffix

当前slot设置：
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
last_boot = 0 or 1;     :refer to androidboot.slot_suffix
```

5.2.3 两种模式的优缺点

1. successful_boot模式

优点：只要正常启动系统，不会回退到旧版本固件，除非system bootctrl配置

缺点：设备长时间工作后，如果存储某些颗粒异常，会导致系统一直重启

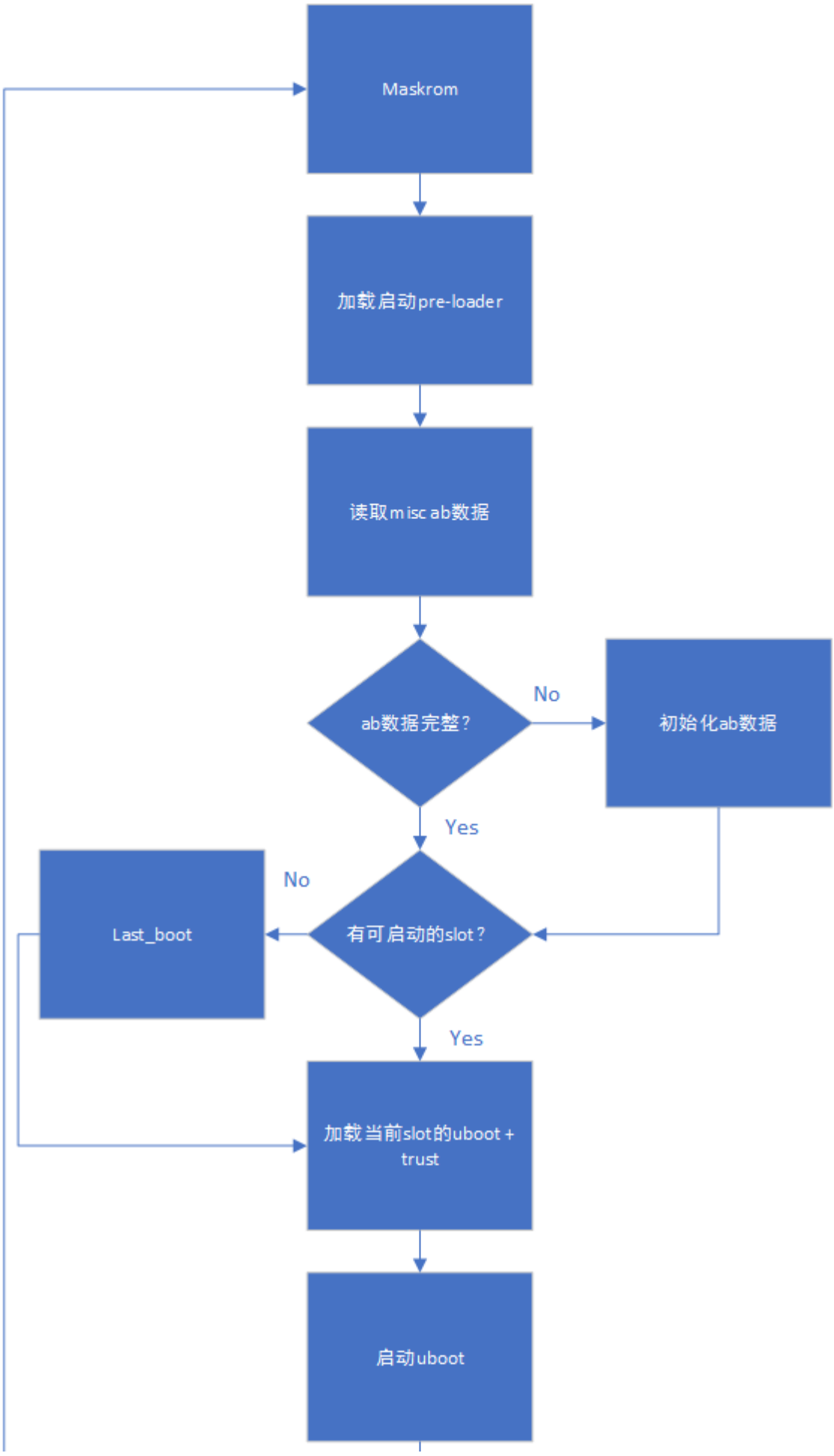
2. reset retry模式

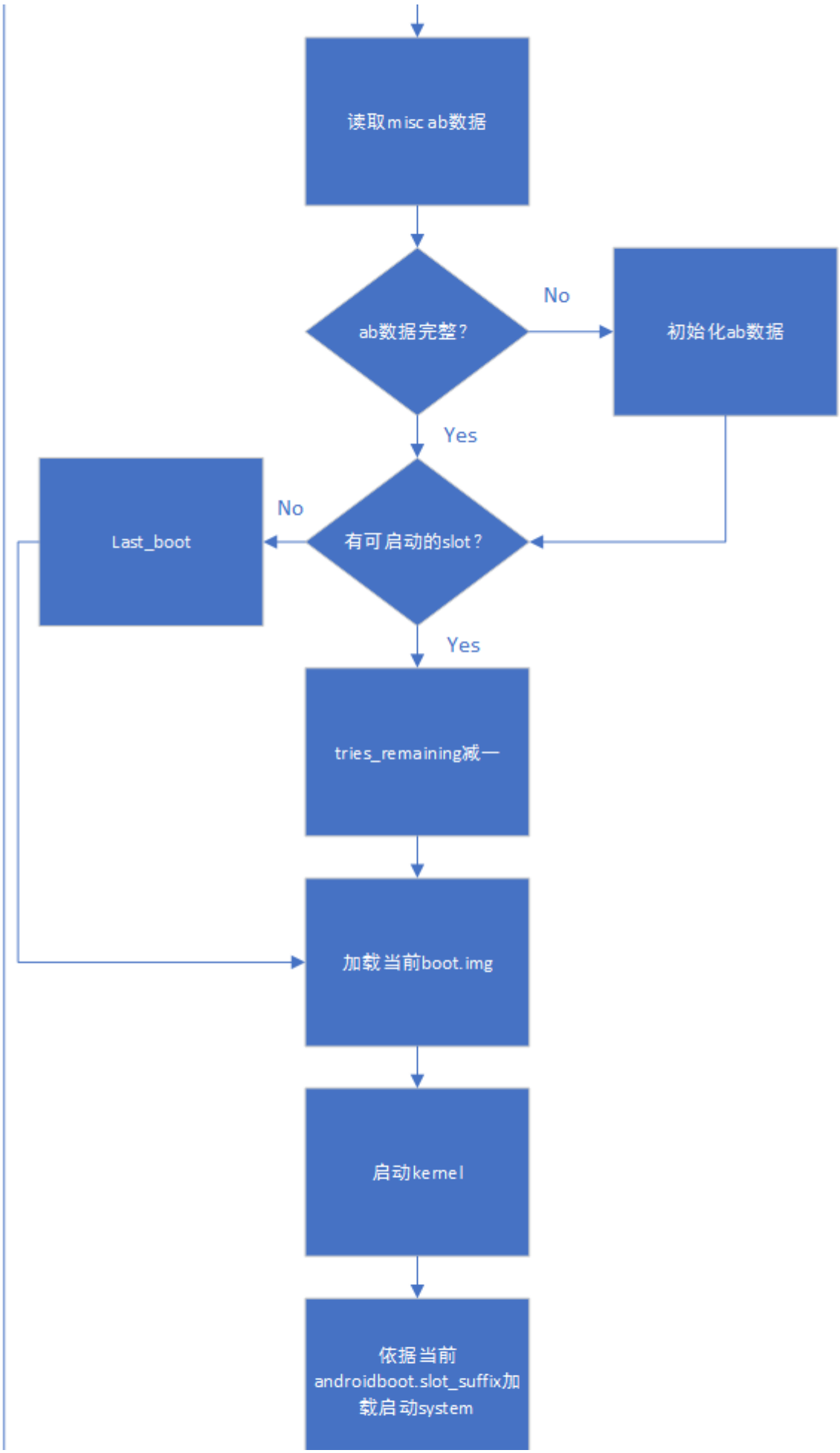
优点：始终保持retry机制，可以应对存储异常问题

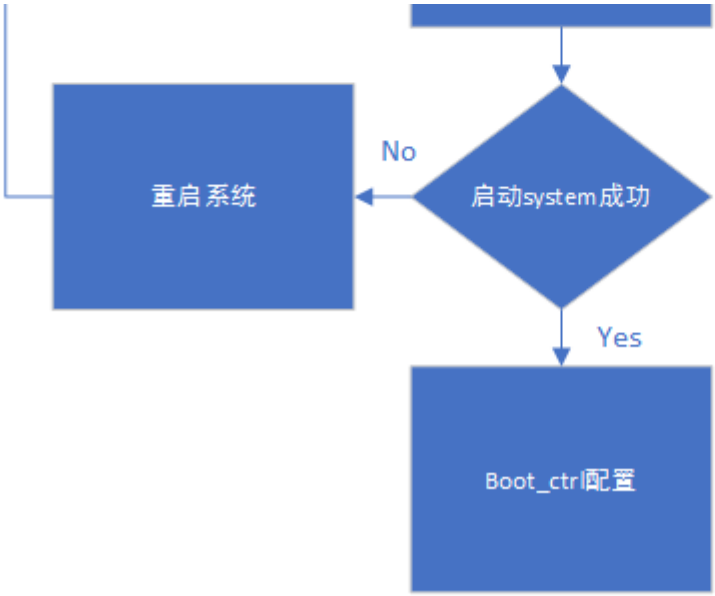
缺点：会回退到旧版本固件

6 流程

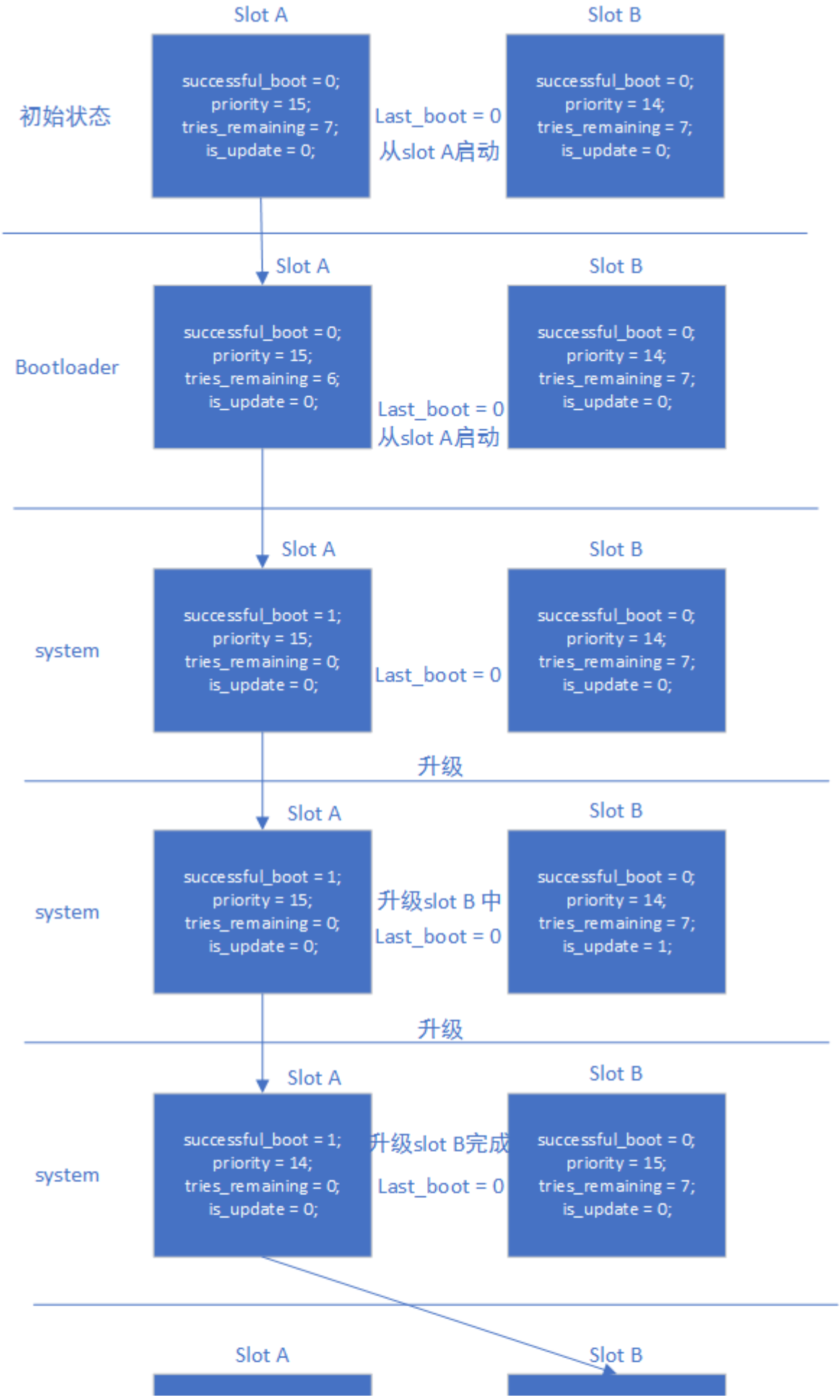
启动流程：

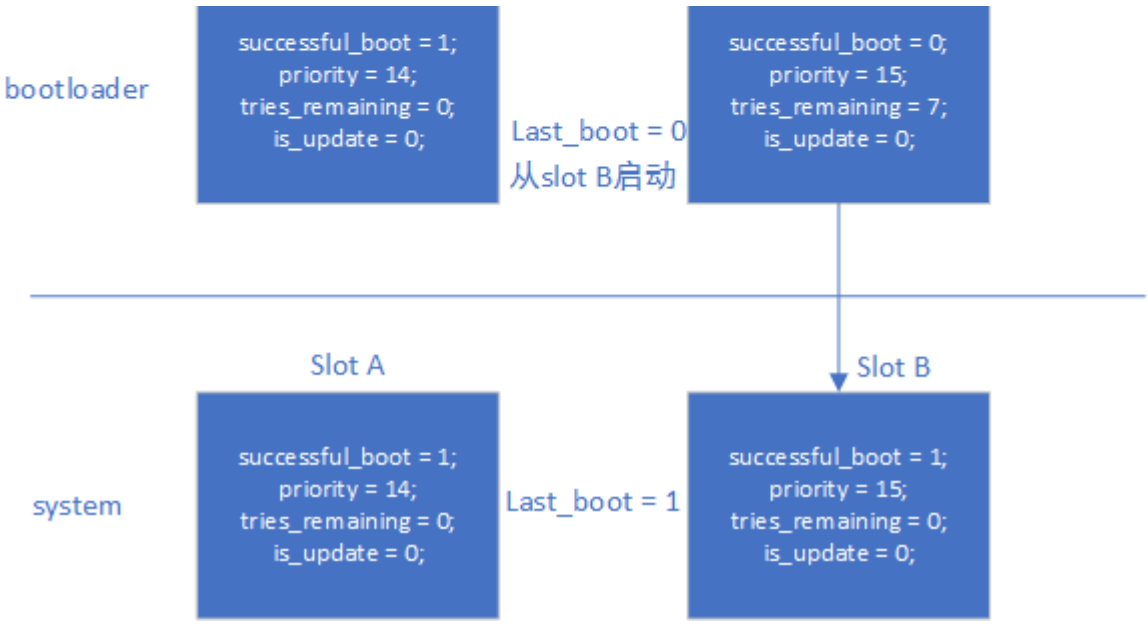

Maskrom
加载启动pre-loader
读取misc ab数据
ab数据完整？
No
初始化ab数据
Yes
有可启动的slot？
No
Last_boot
Yes
加载当前slot的uboot + trust
启动uboot


读取misc ab数据
ab数据完整？
No
初始化ab数据
Yes
有可启动的slot？
No
Last_boot
Yes
tries_remaining减一
加载当前boot.img
启动kernel
依据当前androidboot.slot_suffix加载启动system

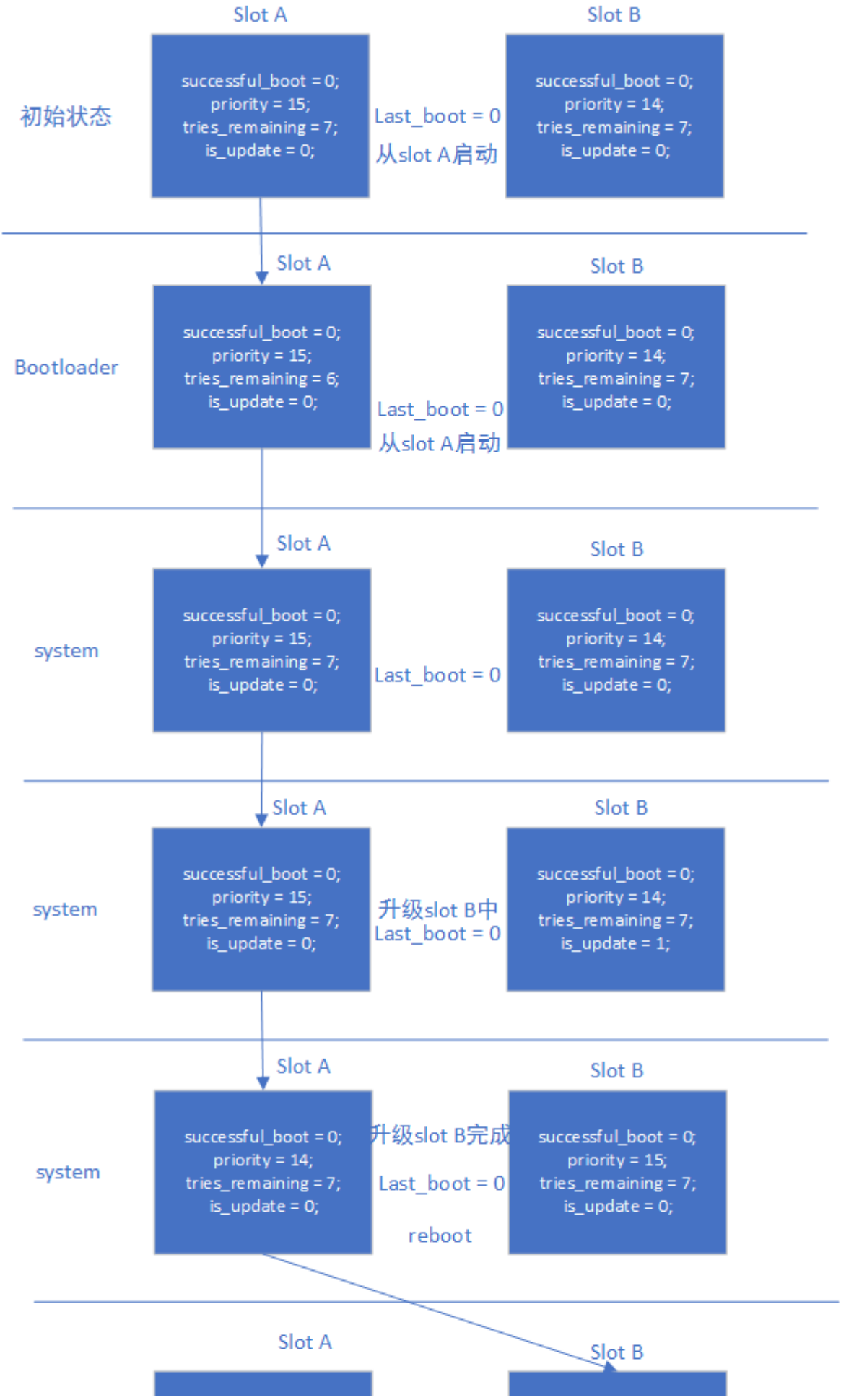


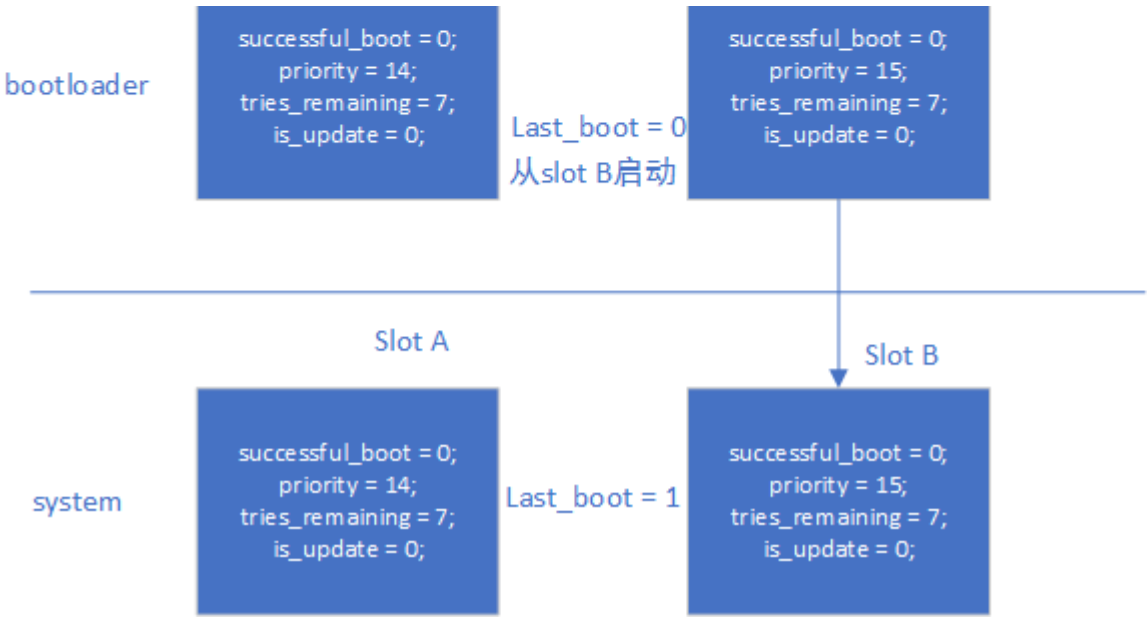
AB successful_boot模式数据流程：


Slot A
Slot B
初始状态
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
Last_boot = 0
从slot A启动
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B
Bootloader
successful_boot = 0;
priority = 15;
tries_remaining = 6;
is_update = 0;
Last_boot = 0
从slot A启动
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B
system
successful_boot = 1;
priority = 15;
tries_remaining = 0;
is_update = 0;
Last_boot = 0
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
升级
Slot A
Slot B
system
successful_boot = 1;
priority = 15;
tries_remaining = 0;
is_update = 0;
升级slot B 中
Last_boot = 0
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 1;
升级
Slot A
Slot B
system
successful_boot = 1;
priority = 14;
tries_remaining = 0;
is_update = 0;
升级slot B完成
Last_boot = 0
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B



AB reset retry模式数据流程：


Slot A
Slot B
初始状态
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
Last_boot = 0
从slot A启动
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B
Bootloader
successful_boot = 0;
priority = 15;
tries_remaining = 6;
is_update = 0;
Last_boot = 0
从slot A启动
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B
system
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
Last_boot = 0
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B
system
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
升级slot B中
Last_boot = 0
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 1;
Slot A
Slot B
system
successful_boot = 0;
priority = 14;
tries_remaining = 7;
is_update = 0;
升级slot B完成
Last_boot = 0
reboot
successful_boot = 0;
priority = 15;
tries_remaining = 7;
is_update = 0;
Slot A
Slot B



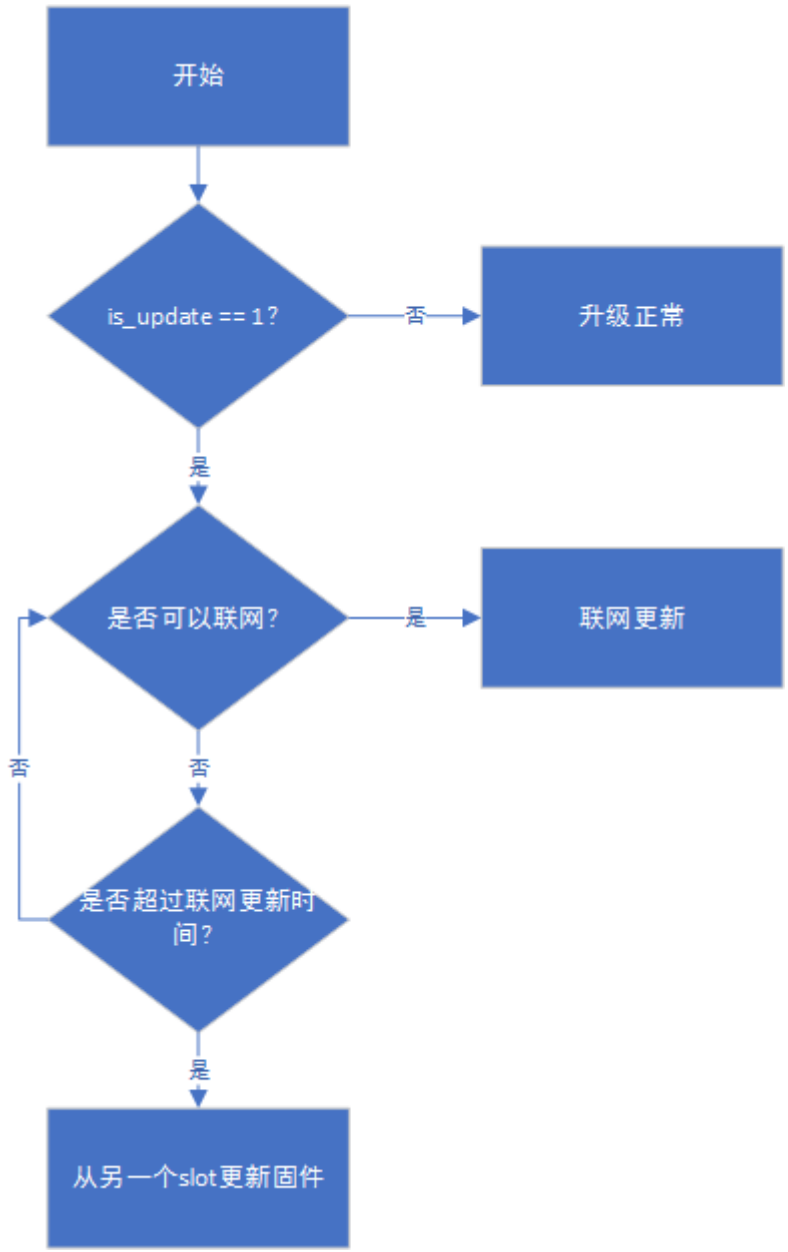
7 升级及升级异常处理参考

7.1 从系统升级

系统升级时，参数配置参考章节5.2 system bootctrl参考。

当升级异常时，is_update为1，可以依据此标志位来再更新系统。

异常更新处理流程参考：



7.2 从recovery升级

AB system不考虑支持recovery升级。

8 分区参考

```
FIRMWARE_VER:8.1
MACHINE_MODEL:RK3326
MACHINE_ID:007
MANUFACTURER: RK3326
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 3326
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
CMDLINE:
mtdparts=rk29xxnand:0x00002000@0x00004000(uboot_a),0x00002000@0x00006000(uboot_b),0x00002000@0x00008000(trust_a),0x00002000@0x0000a000(trust_b),0x00001000@0x0000c000(misc),0x00001000@0x0000d000(vbmeta_a),0x00001000@0x0000e000(vbmeta_b),0x00020000@0x0000e000(boot_a),0x00020000@0x0002e000(boot_b),0x00100000@0x0004e000(system_a),0x00300000@0x0032e000(system_b),0x00100000@0x0062e000(vendor_a),0x00100000@0x0072e000(vendor_b),0x00002000@0x0082e000(oem_a),0x00002000@0x00830000(oem_b),0x0010000@0x00832000(factory),0x00008000@0x842000(factory_bootloader),0x00080000@0x008ca000(oem),-@0x0094a000(userdata)
```

9 测试

准备一套可测试AB的固件。

9.1 successful_boot模式

1. 只烧写slot A，系统从slot A启动。设置从slot B启动，系统从slot A启动。测试完成，清空misc分区
2. 烧写slot A与slot B，启动系统，当前系统为slot A。设置系统从slot B启动，reboot系统，当前系统为slot B。测试完成，清空misc分区
3. 烧写slot A与slot B，迅速reset系统14次后，retry counter用完，还能从last_boot指定的系统启动，即能正常从slot A启动。测试完成，清空misc分区
4. 烧写slot A与slot B，启动系统，当前系统为slot A。设置系统从slot B启动，reboot系统，当前系统为slot B。设置系统从slot A启动，reboot系统，当前系统为slot A。测试完成，清空misc分区

9.2 reset retry模式：

1. 只烧写slot A，系统从slot A启动。设置从slot B启动，系统从slot A启动。测试完成，清空misc分区
2. 烧写slot A与slot B，启动系统，当前系统为slot A。设置系统从slot B启动，reboot系统，当前系统为slot B。测试完成，清空misc分区
3. 烧写slot A与slot B，迅速reset系统14次后，retry counter用完，还能从last_boot指定的系统启动，即能正常从slot A启动。测试完成，清空misc分区
4. 烧写slot A与slot B，其中slot B的boot.img损坏，启动系统，当前系统为slot A。设置系统从slot B启动，reboot系统，系统会重启7次后，从slot A正常启动系统。测试完成，清空misc分区
5. 烧写slot A与slot B，启动系统，当前系统为slot A。设置系统从slot B启动，reboot系统，当前系统为slot B。设置系统从slot A启动，reboot系统，当前系统为slot A。测试完成，清空misc分区