



地平线
Horizon Robotics

X2J2/X3J3

地平线系统软件OTA参考实现原理和使用方法

V3.0

2021-02

版权所有© 2020 Horizon Robotics

保留一切权利

免责声明

本文档信息仅用于帮助系统和软件使用人员使用地平线产品。本文档信息未以明示或暗示方式授权他人基于本文档信息设计或制造任何集成电路。

本文档中的信息如有更改，恕不另行通知。尽管本文档已尽可能确保内容的准确性，本文档中的所有声明、信息和建议均不构成任何明示或暗示的保证、陈述或担保。

本文档中的所有信息均按“原样”提供。地平线不就其产品在任何特定用途的适销性、适用性以及不侵犯任何第三方知识产权方面做出任何明示或暗示保证、陈述或担保。地平线不承担产品使用所引起的任何责任，包括但不限于直接或间接损失赔偿。

买方和正在基于地平线产品进行开发的其他方（以下统称为“用户”）理解并同意，用户在设计产品应用时应承担独立分析、评估和判断的责任。用户应对其应用（以及用于其应用的所有地平线产品）的安全性承担全部责任，并保证符合所有适用法规、法律和其它规定的要求。地平线产品简介和产品规格中提供的“典型”参数在不同应用下可能会不同，实际性能也可能随时间而变化。所有工作参数，包括“典型”参数，都必须由用户自己针对每项用户应用进行验证。

用户同意如因用户未经授权使用地平线产品或因不遵守本说明中的条款，造成任何索赔、损害、成本、损失和（或）责任，用户将为地平线及其代表提供全额赔偿。

© 2018 版权所有

北京地平线信息技术有限公司

<https://www.horizon.ai>

修订记录

版本	修订日期	修订说明
1.0	2020-08-04	地平线系统软件 OTA 参数实现原理和使用方法
2.0	2020-08-05	增加 GPT 分区表修改方法 增加 OTA 升级模式配置方法
3.0	2020-10-19	更新 gpt 信息

目录

免责声明	i
修订记录	ii
目录	iii
1 范围	6
2 术语、定义和缩略语	6
2.1 术语和定义	6
2.2 缩略语	6
3 OTA 实施方案	7
3.1 概述	7
3.1.1 OTA 介绍	7
3.1.2 OTA 系统主要组成部分	8
3.2 系统分区	9
3.2.1 单分区	9
3.2.2 双分区	11
3.3 GPT 分区配置和修改说明	13
3.3.1 分区配置说明	13
3.3.2 分区镜像生成	14
3.4 OTA 升级模式配置	15
3.4.1 初始化配置	15
3.4.2 通过 utility 修改模式	15
3.5 OTA 标志位	16
3.5.1 概述	16
3.5.2 update_success	16
3.5.3 flash_success	16
3.5.4 first_try	16
3.5.5 app_success	18
3.5.6 reset_reason	18
3.5.7 part_status	18
3.6 OTA 日志系统	19
3.6.1 概述	19
3.6.2 commend	19

3.6.3	info	19
3.6.4	progress	20
3.6.5	result.....	20
3.7	OTA 升级脚本.....	21
3.7.1	概述.....	21
3.7.2	otaupdate	21
3.7.3	hbota_utility	22
3.7.4	up_partition.sh	22
3.7.5	s9updatecheck.sh	23
3.8	OTA utility 定义	23
3.8.1	概述.....	23
3.8.2	hrut_boardid.....	23
3.8.3	hrut_eeprom	24
3.8.4	hrut_resetreason	25
3.8.5	hrut_count	25
3.8.6	hrut_otastatus	26
3.8.7	hrut_checksucces.....	26
3.8.8	hrut_checkinfo	27
3.8.9	hrut_checkprogress.....	27
3.8.10	hrut_otaextrainfo.....	28
3.9	warning 信息约定.....	28
3.9.1	功能介绍	28
3.9.2	使用方法	29
3.10	recovery 实现.....	30
3.10.1	recovery 功能介绍.....	30
3.10.2	进入 recovery 模式的方式.....	30
3.11	OTA 升级流程.....	31
3.11.1	golden 模式升级流程.....	31
3.11.2	AB 模式升级流程.....	32
3.11.3	uboot 中 OTA 实现原理.....	34
3.12	OTA secure 方案	36
3.12.1	版本校验.....	36

3.12.2 分区校验.....	36
3.12.3 签名校验.....	36
4 OTA 升级获取方法.....	38
4.1 发布的 SDK 包.....	38
4.2 ota 打包工具.....	39
5 OTA 升级方法.....	39
5.1 otawrite 接口.....	39
5.2 AP 侧升级接口.....	40
5.3 CP 侧升级接口.....	41
5.4 APP 升级方法.....	41
5.4.1 APP 签名方法.....	41
5.4.2 APP 升级约定.....	42
5.5 download 工具实现升级.....	43
6 参考文献.....	45

1 范围

本文描述了 X2J2 和 X3J3 系列芯片中 OTA 系统软件详细设计，包括系统分区、OTA 标志位、日志系统、OTA 相关 utility、warning 信息约定、recovery 系统、OTA 升级流程、OTA 升级包获取方式、以及 OTA 升级方法等。

本文主要介绍 eMMC OTA 升级方案，nor flash 升级方案类似。

2 术语、定义和缩略语

2.1 术语和定义

OTA（Over-the-Air Technology）原指空中下载技术。在地平线系统软件实际实现为不断电升级系统功能。

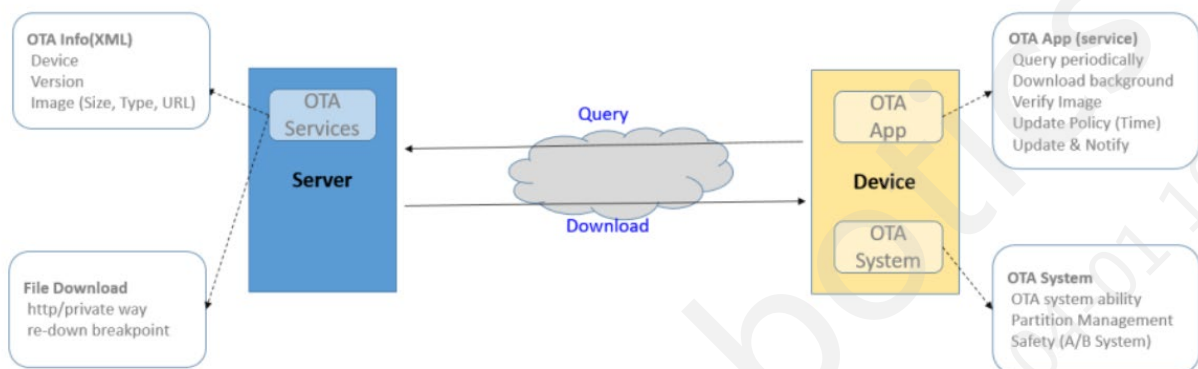
2.2 缩略语

缩略语	英文全称	中文解释
SoC	System on Chip	片上系统
BL[x]	Boot Loader Stage [x]	启动的阶段 x
SPL	Secondary Program Loader	二级程序加载器
GPT	GUID Partition Table	GUID 磁盘分区表
GUID	Globally Unique Identifier	全局唯一标识符

3 OTA 实现方案

3.1 概述

3.1.1 OTA 介绍



(1) OTA Services

- **信息查询服务：**OTA 服务器中保存各系统和软件的最新信息，设备向服务器查询时，提供相应的信息，包括 device、version、image size 等；
- **软件包下载服务：**服务器中保存各系统和软件的安装包，通过网络为设备提供软件包的下载服务。

(2) OTA App

- **查询：**定期向 OTA 服务器查询系统或软件是否有更新；
- **下载：**在查询到系统或软件有更新后，通过网络从 OTA 服务器中下载升级包到本地相应的内存中；
- **验证：**升级包下载后，通过 RSA 和 HASH 算法验证升级包的完整性和安全性，如果验证不通过，则不进行升级。

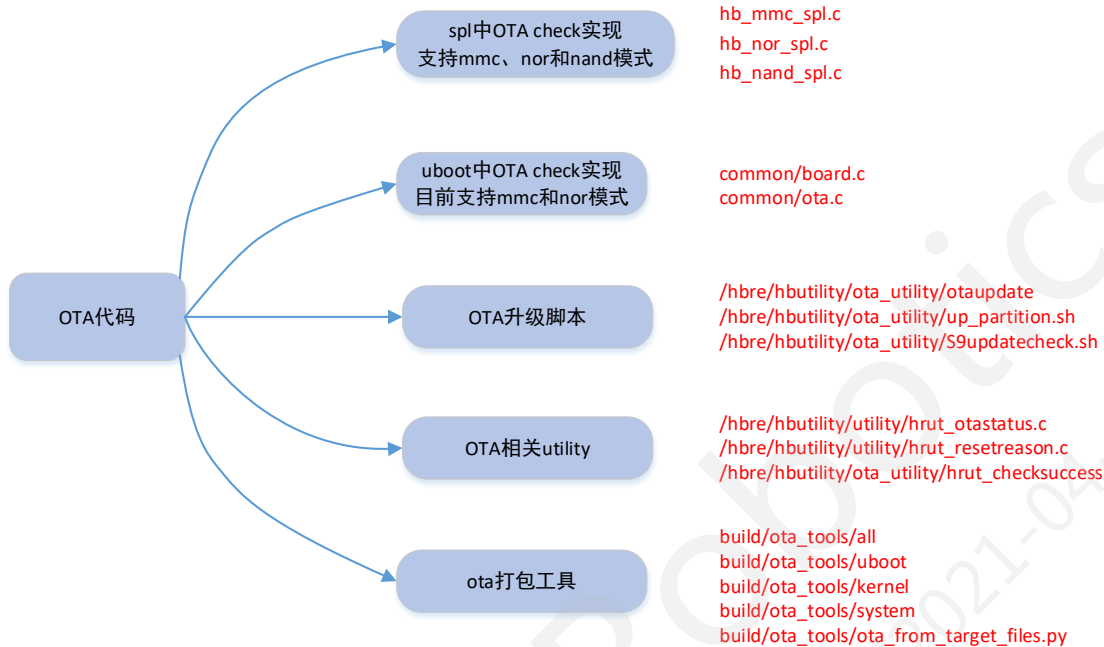
(3) OTA System

- **分区管理：**设备中有不同的分区，包括 uboot、system、recovery 和 cache 等，OTA 升级中将升级包写入对应的分区，完成系统或软件的升级；
- **鲁棒性：**OTA 要确保升级包写入正确的分区，同时可以使用 A/B System 来保证升级失败后，设备仍有可用的系统；
- **安全性：**OTA 支持对升级包进行 RSA 签名校验，确保升级包的完整性；
- **版本管理：**OTA 对升级包和系统的版本进行校验，确保系统升级由低版本到高版本。

(注：目前提供特殊 disk.img OTA 升级包，跳过版本校验，支持由高本版向低版本升级)

3.1.2 OTA 系统主要组成部分

OTA 系统主要由 5 部分组成，如下图列出了每个部分的部分相关代码。



1) SPL

主要实现 OTA 升级的校验功能，根据 OTA 的相关标志位，判断升级是否成功，成功 SPL 会正常启动 "uboot" 分区，失败 golden 模式下会启动 "ubootbak" 分区，进入 recovery 模式。AB 模式下会启动原系统（升级前使用的 "uboot" 分区）。

2) UBoot

实现 OTA 升级的校验功能和 UBoot 命令行下 otawrite 功能。根据 OTA 的相关标志位，判断升级是否成功。成功，UBoot 会正常启动 Kernel，失败 golden 模式下会启动 recovery.gz，进入 recovery 模式。AB 模式下会启动原系统（升级前使用的 kernel 和 system 分区）。

"otawrite" 功能提供 UBoot 下更新 GPT 分区的能力，方便更新系统镜像，具体使用方法见 5.1 小节。

3) OTA 升级脚本

OTA 升级的主体部分，完成 OTA 升级包的校验、系统状态判断、以及 OTA 升级的处理等功能。主要包括 otaupdate, up_partition.sh, S9updatecheck.sh。

4) OTA 相关 utility

Utility 支持对 veeprom 相关标志位的读写功能，支持对 log 日志的读功能。通过 utility，OTA 可以方便的处理升级中的状态和日志信息。

5) OTA 打包工具

打包工具用于生成 OTA 升级包，升级包中会加入 GPT 信息、版本信息、warning 信息和对应的 update.sh 升级脚本。支持生成分区升级包和 disk.img 升级包，支持生成 eMMC 和 nor flash 的升级包。

注：打包工具介绍详见 4.2 小节。

3.2 系统分区

3.2.1 单分区

● 分区设计

“veeprom, spl, ddr, bl31, vbmeta, system, app, bpu, userdata” 为单分区

其中 UBoot 和 Kernel 分区做双备份，ubootB 和 recovery (KernelB) 用于 recovery 模式，在 OTA 升级失败的情况下，进入 recovery 模式进行再升级/恢复。

veeprom	spl	ddr	bl31	ubootA	ubootB	vbmeta	kernel	recovery	system	bpu	app	userdata
---------	-----	-----	------	--------	--------	--------	--------	----------	--------	-----	-----	----------

- veeprom: 存放 OTA 相关标志位，开发板的相关信息，包括：IP，MAC，upmode，upflag 等
- spl: 存放 second boot 的镜像，对应 spl.bin;
- DDR: 存放 DDR 的初始化镜像和配置 efuse 的 bin (可配)，对应 ddr_storage.bin 和 efuse.bin
- BL31: 存放 bl31 的镜像，对应 bl31.bin
- uboot: uboot 分区做双备份，保存 ubootA 和 ubootB
 - ubootA: 存放 normal boot 过程中的 third boot 的镜像，对应 uboot.bin;
 - ubootB: 存放 recovery boot 过程中的 third boot 的镜像，对应 uboot.bin;
- vbmeta: 存放 vbmeta 的数据
- kernel: 存放 kernel 镜像，包括 Image.gz,*.dtb, dtb-mapping.conf 等
- recovery: 存放 recovery 镜像，包括与 kernel 相同的 Image.gz,*.dtb,dtb-mapping.conf 以及 cpio
- system: 存放 rootfs 相关文件，对应 etc, sbin, usr, bin, lib 等;
- bpu: 存放 bpu image 的镜像
- app: 存放相关应用程序，例如：adas-rt 的应用程序;
- userdata: 存放 user 的数据，例如：OTA 升级包，DDR dump 数据，APP 存放的数据等。

● 分区属性

分区名	分区偏移	分区大小	分区格式	加载到内存地址
Head(MBR+GPT)	0x0000 0000	17K	raw	*
veeprom	0x0000 4400	2K	raw	*
spl	0x0000 4C00	512K	raw	0x8020 0000
ddr	0x0008 4C00	512K	raw	*
bl31	0x0010 4C00	512K	raw	0x0000 0000
uboot	0x0018 4C00	2M	raw	0x0400 0000
vbmeta	0x0038 4C00	128K	raw	*

boot	0x003A 4C00	10M	raw	0x0028 0000
recovery	0x00DA 4C00	15M	raw	*
system	0x01CA 4C00	150M	ext4	*
bpu	0x0B2A 4C00	100M	raw	*
app	0x116A 4C00	256M	ext4	*
userdata	0x216A 4C00	6921M	ext4	*

● 分区实现

		offset
MBR head	MBR head + 512 info	0x0
GPT head	GPT head	
VEEPROM	veeprom	0x0000 4400
SPL	spl.bin	0x0000 4c00
DDR	ddr.img	0x0008 4C00
BL31	bl31.bin	0x0010 4C00
UBOOT	uboot.bin(normal) uboot.bin(recovery)	0x0018 4C00
VBMENTA	vbmenta	0x0038 4C00
KERNEL	Image.gz, *dtb recovery.gz, *dtb	0x003A 4C00
SYSTEM	rootfs	0x01CA 4C00
BPU	bpu image	0x0B2A 4C00
APP	app	0x116A 4C00
USERDATA	userdata	0x216A 4C00

3.2.2 双分区

- 分区设计：

spl, ddr, app, userdata 为单分区

其中 uboot 分区做双备份，ubootB 用于 recovery 模式，在 OTA 升级失败的情况下，进入 recovery 模式进行再升级。

kernel 和 system 为双分区，AB 为等价的双系统

veeprom	spl	ddr	bl31	ubootA	ubootB	vbmetaA	vbmetaB	kernelA	kernelB	systemA	systemB	bpu	app	userdata
---------	-----	-----	------	--------	--------	---------	---------	---------	---------	---------	---------	-----	-----	----------

veeprom、spl、DDR、app、userdata 介绍同单分区，这里不再赘述。

- uboot: uboot 分区做双备份 ubootA 和 ubootB
- vbmeta: 双分区 vbmetaA 和 vbmetaB
- kernel: 双分区 kernelA 和 kernelB
- system: 双分区 systemA 和 systemB

● 分区属性：

分区名	分区偏移	分区大小	分区格式	加载到内存地址
Head(MBR+GPT)	0x0000 0000	17K	raw	*
veeprom	0x0000 4400	2K	raw	*
spl	0x0000 4C00	512K	raw	0x80200000
ddr	0x0008 4C00	512K	raw	*
bl31	0x0010 4C00	512K	raw	0x0000 0000
uboot	0x0018 4C00	2M	raw	0x4000000
vbmeta	0x0038 4C00	128K	raw	*
vbmeta_b	0x003A 4C00	128K	raw	*
boot	0x003C 4C00	10M	raw	0x280000
boot_b	0x00DC 4C00	10M	raw	0x280000
system	0x017A 4C00	150M	ext4	*
system_b	0x0ADC 4C00	150M	ext4	
bpu	0x143C 4C00	100M	raw	*
app	0x1A7C 4C00	256M	ext4	*
userdata	0x2A7C 4C00	6776M	ext4	*

● 分区实现：

		offset
MBR head	MBR head + 512 info	0x0
GPT head	GPT head	
VEEPROM	veeprom	0x0000 4400
SPL	spl.bin	0x0000 4c00
DDR	ddr.bin	0x0008 4C00
BL31	bl31.bin	0x0010 4C00
UBOOT	uboot.bin(normal) uboot.bin(recovery)	0x0018 4C00
VBMENTA	vbmenta vbmenta-bak	0x0038 4C00
KERNEL	Image.gz, *dtb Image-bak.gz, *dtb	0x003C 4C00
SYSTEM	rootfs rootfs-bak	0x017A 4C00
BPU	bpu image	0x143C 4C00
APP	app	0x1A7C 4C00
USERDATA	userdata	0x2A7C 4C00

3.3 GPT 分区配置和修改说明

3.3.1 分区配置说明

- 配置文件：
 - 单分区：build/device/horizon/x3/debug-gpt.config
 - 双分区：build/device/horizon/x3/debug-gpt-dual.conf

配置文件通过“.”分割，不同的位表示不同的内容，如下是单分区的分区表

```
1. 1:veeprom:none:34s:37s:1
2. 1:sbl/${UBOOT_SPL_NAME}:none:38s:1061s:1
3. 0:sbl/${UBOOT_WARM_SPL_NAME}:none:550s:1061s:1
4. 1:ddr/${DDR_IMAGE_NAME}:none:1062s:2085s:1
5. 0:ddr/${EFUSE_IMAGE_NAME}:none:2080s:2085s:1
6. 1:bl31/${BL31_IMAGE_NAME}:none:2086s:3109s:1
7. 1:uboot/${UBOOT_IMAGE_NAME}:none:3110s:7205s:1
8. 0:ubootbak/${UBOOT_IMAGE_NAME}:none:5158s:7205s:1
9. 1:vbmeta/${VBMETA_IMAGE_NAME}:none:7206s:7461s:1
10. 1:boot/${BOOT_PART_IMAGE_NAME}:none:7462s:27941s:1
11. 1:recovery/${BOOT_RECOVERY_IMAGE_NAME}:none:27942s:58661s:1
12. 1:system:ext4:58662s:365861s:1
13. 1:bpu/${BPU_IMAGE_NAME}:none:365862s:570661s:1
14. 1:app:ext4:570662s:1094949s:0
15. 1:userdata:none:1094950s:15269751s:0
```

说明：如果需要调整分区大小，可以直接修改 GPT 分区表中的 start_sector 和 end_sector，后续的分区需要做对应调整。

- 配置文件不同的位表示内容如下：

0	0: 不需要单独分区，如 uboot 和 uboot-bak 只需要分配一个分区，uboot 指定整个分区大小，ubootbak 指定在整个分区占用的区域 1: 需要单独分区
1	分区名
2	文件系统类型；none 表示不需要格式化
3	start sector
4	end sector
5	0: 不需要打包到 disk.img（烧写镜像），分区内无初始化内容；如 app、userdata 分区 1: 需要打包到 disk.img

3.3.2 分区镜像生成

- 相关脚本

./partition.sh debug-gpt.conf <===== 这里生成分区镜像

- 添加文件到分区

该过程由 partition.sh 脚本完成，如果生成镜像的位置有文件夹名称和分区表名一样，则会在分

区过程中自动拷贝文件夹内容到镜像中。

例如：以上面 debug-gpt.conf 为例

在 partition.sh 所在的目录，如果有 kernel 目录，则会在生成镜像的时候，将这个目录下的所有内容拷贝到对应的分区中

如果 system 分区没有对应的名为 system 文件夹，则建立好镜像后，该分区没有文件。

如果文件系统格式为 none 的条目，则使用 dd 命令将目标文件导入到 disk.img 镜像中

3.4 OTA 升级模式配置

3.4.1 初始化配置

- 配置文件

路径：hbre/hbutils/ota_utility/S9updatecheck.sh

SDK 中路径：prebuilts/root/etc/rcS.d/ S9updatecheck.sh

- 初始化

模式初始化的位置如下，默认给 golden 模式，如果需要 AB 模式可以手动修改。

```
function veeprom_init()
{
    # boot success set count 10, if count <=0 ,using bak partition
    set_count 25

    local file="/usr/bin/hrut_boardid"
    if [ ! -f $file ];then
        echo "[OTA_INFO] file $file not exist"
        exit 1
    fi

    file="/usr/bin/hrut_otastatus"
    ota_upmode=$(($file g upmode))

    if [ -z "$ota_upmode" ] || [ x"$ota_upmode" != x"AB" -a x"$ota_upmode" != x"golden" ];then
        echo "[OTA_INFO] veeprom init "

        board_id=$(cat /sys/class/socinfo/board_id)
        /usr/bin/hrut_boardid s $board_id
        tmp=$(hrut_count 25)
        tmp=$(hrut_otastatus s upflag 13)
        tmp=$(hrut_otastatus s upmode golden)
        tmp=$(hrut_otastatus s partstatus 0)
        tmp=$(hrut_resetreason normal)
        tmp=$(hrut_countflag s on)
    fi
}
```

3.4.2 通过 utility 修改模式

提供 hrut_otastatus 工具修改升级模式，AB 模式需要双分区支持，否则 OTA 升级会报错。

- 查询方法

hrut_otastatus g upmode

- 设置方法

hрут_otastatus s upmode <AB | golden>

说明：utility 工具的使用方法，可以参考 3.8.6 小节

3.5 OTA 标志位

3.5.1 概述

OTA 升级的标志位保存在 veeprom 分区，用于在 OTA 升级的过程中进行完整性和鲁棒性判断，保证 OTA 升级的顺利进行。

实现的标志位有：update_success, flash_success, first_try, app_success, reset_reason 和 part_status。

3.5.2 up_flag 标志位

其中，update_success, flash_success, first_try 以及 app_success 统一作为"upflag"，占用 1 个 byte，比特分布为：

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reserved	reserved	reserved	reserved	update_success	flash_success	First_try	app_success

3.5.2.1 update_success

- 功能说明：

此标志位用于判断 OTA 升级是否成功

- 具体实现：

- 1: 升级成功
- 0: 升级失败

3.5.2.2 flash_success

- 功能说明：

此标志位用于判断升级包镜像写入 emmc 或 nor 中是否成功

- 具体实现：

- 1: emmc 镜像写入成功
- 0: emmc 镜像写入失败

3.5.2.3 first_try

- 功能说明：

此标志用于判断 OTA 升级后，是否是第一次启动。

如果是第一次启动，则会尝试启动更新的分区或系统；否则正常启动。

➤ 具体实现：

1: OTA 升级后的第一次启动

0: 正常启动

3.5.3 app_success

➤ 功能说明：

此标志位用于判断，镜像更新重启后，核心 APP 是否启动成功。

➤ 具体实现：

1: 更新后进入 rootfs，app 启动成功

0: 更新后进入 rootfs，app 启动失败

注：目前因没有具体 APP 是否启动成功的判断功能，当能正常进入 rootfs，则认为 app 启动成功。

3.5.4 reset_reason

➤ 功能说明：

用于记录此次 OTA 升级更新的具体 partition name，正常启动或强制进入 recovery 模式。

➤ 具体实现：

uboot/system/kernel: 表示此次 OTA 升级的是对应分区

all: 表示 OTA 升级所有的分区，包括 uboot、kernel 和 system

normal: 正常启动

recovery: 进入 recovery 模式

3.5.5 part_status

➤ 功能说明：

用于标记当前系统所处的分区状态（即 A 分区或 B 分区）

例如：

升级前： 00000000 在 ubootA 分区

升级 uboot 分区后： 00000010 在 ubootB 分区

➤ 具体实现：

0: 表示在 A 分区

1: 表示在 B 分区

初始状态：00000000，升级对应的分区后，在 AB 模式下会更新分区状态每一位对应的分区如下：

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
		userdata	app	system	kernel	uboot	spl

3.6 OTA 日志系统

3.6.1 概述

OTA 日志系统用于保存 OTA 升级过程中的 log 信息、出错时的错误信息和 OTA 升级的结果信息。

- 日志存放位置：

- 默认使用/userdata/cache 目录存放日志
- /userdata/cache 空间满，则使用/tmp/cache 空间临时存放日志
- /tmp/cache 空间也满，就停止升级（ps: 这时升级包也已经放不下了）

说明：

放在/tmp 空间，重启后日志丢失，这个影响不是很大，OTA 重启后会再次创建日志文件，确保 download 工具 query 时必要日志存在，只是丢失了一些 log 信息（ps: 使用/ota 目录，更新 disk.img 也会有这个问题）

3.6.2 command

- 功能说明：

保存升级分区名和升级包的绝对路径，用于 up_partition.sh 脚本或 recovery 模式获取升级的分区和升级包。

- 文件格式：

partition_name/absolute_path

partition_name: 指要升级的分区（uboot, kernel, system, all 等）

absolute_path: 升级包的绝对路径

- 实现示例：

1. uboot//userdata/up_img/uboot.zip

3.6.3 info

- 功能说明：

保存 OTA 升级过程中的 log 信息，用于查看 OTA 升级的细节和出错时定位具体的原因。

- 文件格式：

[OTA_INFO]log[OFNI_ATO]

[OTA_INFO]: log 信息前后加上 OTA_INFO，便于 download 工具获取 log 信息

- 实现示例：

1. [OTA_INFO]boot mode: emmc[OFNI_ATO]
2. [OTA_INFO]parameter check success[OFNI_ATO]
3. [OTA_INFO]get ota_upmode golden[OFNI_ATO]
4. [OTA_INFO]upgrade_need_space: 1027[OFNI_ATO]
5. [OTA_INFO]tmp_free_space: 95920[OFNI_ATO]
6. [OTA_INFO]userdata_free_space: 2924764[OFNI_ATO]
7. [OTA_INFO]get ota_upmode golden[OFNI_ATO]
8. [OTA_INFO]check update mode golden success[OFNI_ATO]

3.6.4 progress

- 功能说明：

保存 OTA 升级的进度信息，用于查看 OTA 的进度。

- 文件格式：

[OTA_PROGRESS]log[SSERGORP_ATO]

[OTA_PROGRESS]: log 信息前后加上 OTA_PROGRESS，便于 download 工具获取 log 信息

- 实现示例：

1. [OTA_PROGRESS]5%[SSERGORP_ATO]
2. [OTA_PROGRESS]10%[SSERGORP_ATO]
3. [OTA_PROGRESS]15%[SSERGORP_ATO]
4. [OTA_PROGRESS]20%[SSERGORP_ATO]
5. [OTA_PROGRESS]30%[SSERGORP_ATO]
6. [OTA_PROGRESS]90%[SSERGORP_ATO]
7. [OTA_PROGRESS]95%[SSERGORP_ATO]
8. [OTA_PROGRESS]100%[SSERGORP_ATO]

3.6.5 result

- 功能定义：

保存 OTA 升级的结果信息

- 文件格式：

使用 0,1,2 来表示升级的结果，具体定义如下

- 0: 升级进行中或未开始
- 1: 升级成功
- 2: 升级失败

- 实现示例：

1. 0 //升级开始时，写入 0
2. 1 //升级成功，写入 1

3.7 OTA 升级脚本

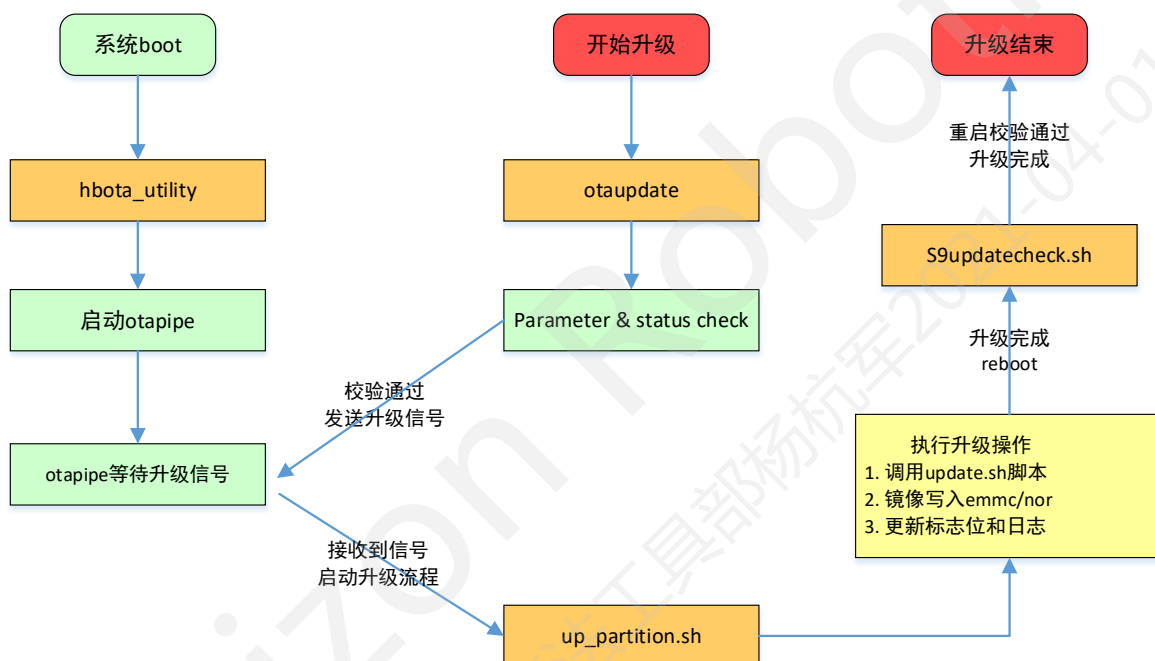
3.7.1 概述

OTA 升级的主要部分，完成 OTA 升级包校验、系统状态判断、以及 OTA 升级镜像的写入功能。主要包括 otaupdate, hbota_utility, up_partition.sh 和 S9updatecheck.sh。

代码路径：

[hbre/hbutils/ota_utility](#)

脚本之间的关系：



3.7.2 otaupdate

● 功能说明：

Otaupdate 是 CP 侧升级的接口，OTA 升级会首先在 CP 侧调用此接口，该脚本的主要功能有：

- **参数检查：**首先检查 partition 和 image 是否合法，partition 是否为系统的分区，image 升级文件是否存在，partition 和 image 是否匹配等。
- **创建日志：**升级之前先创建 info, result, progress 等日志文件，保存 OTA 升级的日志信息和升级出错时的错误信息。
- **模式检查：**检查系统是否支持 recovery 模式，如果 golden 模式下没有 recovery.gz 镜像，则停止升级。

- **剩余空间检查：**升级前查询系统空间使用情况，确定使用 `userdata` 空间或 `tmp` 空间进行 OTA 升级。如果需要的空间不足，则停止升级。
- **系统版本认证：**对升级包中的版本信息进行验证，详见 3.10.1 小节
- **GPT 分区认证：**对升级包中的 GPT 信息进行验证，详见 3.10.2 小节。
- **RSA 签名认证：**对升级包中的签名进行验证，详见 3.10.3 小节。
- **Warning 信息提取：**提取升级包中的 `warning` 信息，并输出到日志 `warning` 中，详见 3.7 小节。
- **向 `otapipe` 发送升级信号：**所有上述验证通过后，`otaupdate` 会向 `otapipe` 发送升级信息，信号格式如下：

```
1. echo "$file $up_partition $image $upgrade_file_path" > $pipe
```

● 接口定义：

```
1. root@X2SOM1V8:~# otaupdate
2. Usage:
3.     otaupdate partition_name img_file
4.     --partition_name:
5.         uboot | kernel | system | app | all
6.     --img_file:
7.         uboot.zip | kernel.zip | system.zip | all_in_one.zip
```

3.7.3 `hbota_utility`

● 功能说明：

在系统启动的时候，`hbota_utility` 会在后台创建 `otapipe` 管道，`otapipe` 接收到 OTA 升级信号后主要做 2 件事：

- 关闭在 OTA 升级前需要退出的程序，例如：HBIPC 相关进程。
- 调用 `up_partition.sh` 进行升级。

3.7.4 `up_partition.sh`

● 功能说明：

实现镜像写入 `emmc` 或 `nor flash` 的功能，支持对单分区镜像和 `all_in_one.zip` 的升级。`up_partition.sh` 在确认升级包在后，会调用升级包中的 `update.sh` 脚本来完成镜像写入。

- **参数检查：**确保升级包存在
- **调用 `update.sh`：**升级包中存在 `update.sh`，直接调用 `update.sh` 执行升级。
- **直接升级：**升级包中不存在 `update.sh`，判断是否是分区 `img` 镜像，如果是直接升级。
- **标志位和日志更新：**镜像写入成功后，更新 `upflag`, `resetreason` 等相关标志位和日志，用于重启后，系统判断升级是否成功。
- **重启：**执行 `reboot` 操作。

3.7.5 s9updatecheck.sh

- 功能说明:

初始化和 OTA 升级后检查脚本，主要功能如下：

- **初始化 veeprom:** 在系统启动时，该脚本会检查 veeprom 是否初始化，如果未初始化，则初始化 boardid, upflag, upmode, resetreason 和 partstatus 等标志位。
- **启动 otapipe:** 在系统启动时，会自动启动 hbota_utility, 创建管道 otapipe, 等待接收 OTA 升级信号。
- **检查升级是否成功:** 根据标志位检查 OTA 升级是否成功，并写入相关结果到 veeprom 的标志位和日志文件中，
- **恢复功能:** 如果升级失败，golden 双分区模式会尝试使用备份分区恢复原分区，确保系统的鲁棒性。

3.8 OTA utility 定义

3.8.1 概述

OTA 相关的 utility 用于辅助 OTA 升级，支持对 veeprom 中相关标志位的读写，支持对 OTA 日志文件的查询。

Utility 相关代码路径如下：

[hbre/hbutils/utility](#)

veeprom 分区根据存储介质不同，分区定义如下：

```
#define VEEPROM_START_SECTOR (34)
#define VEEPROM_END_SECTOR (37)

#define VEEPROM_NOR_START_SECTOR (0)
#define VEEPROM_NOR_END_SECTOR (3)
```

3.8.2 hrut_boardid

- 功能说明:

查询和设置开发板的 board id

- 实现原理:

s 和 g 读写 veeprom 标志位，标志位在 veeprom 中定义如下：

```
#define VEEPROM_BOARD_ID_OFFSET 0
#define VEEPROM_BOARD_ID_SIZE 2
```

S 和 G 读写 bootinfo 中 board id 位

- 使用方法:

X2J2:

1. root@X2SOM1V8:~# hrut_boardid
2. Usage: hrut_boardid [OPTIONS] <Values>
3. Example:
4. hrut_boardid g
5. hrut_boardid s 103
6. Options:
7. g get board id(veeprom)
8. s set board id(veeprom)
9. G get board id(bootinfo)
10. S set board id(bootinfo)
11. C clear board id
12. H display this help text
13. Values:
14. [101|103|202|203|301|302....]

X3J3:

```
root@j3dvbj3-hynix2G-3200:/userdata# hrut_boardid
Usage: hrut_boardid [OPTIONS] <Values>
Example:
    hrut_boardid g
Options:
    g  get board id(veeprom)
    s  set board id(veeprom)
    G  get board id(bootinfo)
    S  set board id(bootinfo)
    c  clear board id(veeprom)
    C  clear board id(bootinfo)
    h  display this help text
```

3.8.3 hrut_eeprom

- 功能说明:

查询和设置 veeprom 中的数据

- 实现原理:

直接对 veeprom 分区进行操作, read/write/clear/dump 分区中的数据

- 使用方法:

1. root@X2SOM1V8:~# hrut_eeprom

2. usage:
3. hrut_eeprom r ADDR SIZE : Read data in eeprom
4. hrut_eeprom w ADDR 0xAA 0xBB ** : Write data to eeprom
5. hrut_eeprom c : Clear eeprom
6. hrut_eeprom d : Display all the data in eeprom

3.8.4 hrut_resetreason

- 功能说明:

查询或设置系统的 resetreason

- 实现原理:

读写 veeeprom 中的 resetreason 标志位, 标志位定义如下:

```
#define VEEPROM_RESET_REASON_OFFSET 9
#define VEEPROM_RESET_REASON_SIZE 8
```

- 使用方法:

查询: 不带参数, 直接执行 hrut_resetreason

设置: 带参数, 设置执行 hrut_resetreason <value>

value 取值范围: uboot | kernel | system | all | normal | recovery

1. root@X2SOM3V3:~# hrut_resetreason
2. normal
3. root@X2SOM3V3:~#
4. root@X2SOM3V3:~#
5. root@X2SOM3V3:~# hrut_resetreason recovery
6. recovery

3.8.5 hrut_count

- 功能说明:

OTA 中引入 count 功能, 如果系统连续 25 次启动未成功, 会进入备份系统或 recovery 模式。

注: 未启动成功指系统启动未进入 rootfs

在 build/ddr/hb_imem_parameter.c 文件 init_ddr_header 函数中可以设置 ota_check_count, 设置为 0, 上电将不检查 count。默认设置为 25, 表示连续 25 次启动未成功, 进入备份或 recovery 模式。

- 实现原理:

Count 加操作: 每次启动 spl 中会做 count +1 操作

Count 减操作: 每次进入 rootfs 中, S9updatecheck.sh 脚本会初始化 count = 0

读写 veeeprom 中的 count 标志位, 标志位定义如下:

```
#define VEEPROM_COUNT_OFFSET 38
#define VEEPROM_COUNT_SIZE 1
```

- 使用方法:

查询：不带参数，直接执行 `hrut_count`

设置：带参数，执行 `hrut_count <value>`

value 取值范围：0 - 25

```
1. root@X2SOM3V3:~# hrut_count
2. 25
3. root@X2SOM3V3:~#
4. root@X2SOM3V3:~#
5. root@X2SOM3V3:~# hrut_count 9
6. 9
```

3.8.6 hrut_otastatus

- **功能说明：**

查询或设置 OTA 相关标志，包括：upflag, upmode 和 upstatus

- **实现原理：**

读写 veeprom 中的 upflag, upmode 和 upstatus 标志位，标志位定义如下：

```
#define VEEPROM_UPDATE_FLAG_OFFSET 8
#define VEEPROM_UPDATE_FLAG_SIZE 1

#define VEEPROM_UPDATE_MODE_OFFSET 29
#define VEEPROM_UPDATE_MODE_SIZE 8
#define VEEPROM_ABMODE_STATUS_OFFSET 37
#define VEEPROM_ABMODE_STATUS_SIZE 1
```

- **使用方法：**

```
1. root@X2SOM1V8:~# hrut_otastatus
2. Usage: hrut_otastatus [Options] <Target> <value>
3. Example:
4.   hrut_otastatus g upflag
5.   hrut_otastatus s upflag 13
6. Options:
7.   g   gain [upflag | upmode | partstatus]
8.   s   set [upflag | upmode | partstatus]
9.   h   display this help text
10. Target:
11.   upflag   OTA update flag
12.   upmode   OTA update mode [AB | golden]
13.   partstatus GPT partition status
```

3.8.7 hrut_checksucces

- **功能说明：**

查询 OTA 升级的 result 信息

- **实现原理：**

直接读取 log 文件：result 的结果，result 中信息的定义详见 OTA 日志系统
默认输出 3 次（ps: 与 download tools 之间的约定，方便获取结果）

- 使用方法：

```
1. root@X2SOM3V3:~# hrut_checksucces
2. 1
3. 1
4. 1
```

3.8.8 hrut_checkinfo

- 功能说明：

查询 OTA 升级的 info 信息

- 实现原理：

直接读取 log 文件：info 的结果，info 中信息的定义详见 OTA 日志系统
默认输出 info 文件的最后一行 log 信息，输出 3 次（ps: 与 download tools 之间的约定，方便获取结果）

- 使用方法：

参数：l 会过滤掉[OTA_INFO] [OFNI_ATO]

```
1. root@X2SOM3V3:~# hrut_checkinfo l
2. write uboot img to emmc success
3. write uboot img to emmc success
4. write uboot img to emmc success
5. root@X2SOM3V3:~#
6. root@X2SOM3V3:~#
7. root@X2SOM3V3:~# hrut_checkinfo
8. [OTA_INFO]write uboot img to emmc success [OFNI_ATO]
9. [OTA_INFO]write uboot img to emmc success [OFNI_ATO]
10. [OTA_INFO] write uboot img to emmc success [OFNI_ATO]
```

3.8.9 hrut_checkprogress

- 功能说明：

查询 OTA 升级的 progress 信息

- 实现原理：

直接读取 log 文件 progress，progress 中信息的定义详见 OTA 日志系统
默认输出 progress 文件的最后一行 log 信息，输出 3 次（ps: 与 download tools 之间的约定，方便获取结果）

- 使用方法：

参数：l 会过滤掉[OTA_INFO] [OFNI_ATO]

```
1. root@X2SOM3V3:~# hrut_checkprogress l
2. 100%
3. 100%
4. 100%
5. root@X2SOM3V3:~#
6. root@X2SOM3V3:~#
7. root@X2SOM3V3:~# hrut_checkprogress
8. [OTA_PROGRESS]100%[SSERGORP_ATO]
9. [OTA_PROGRESS]100%[SSERGORP_ATO]
10. [OTA_PROGRESS]100%[SSERGORP_ATO]
```

3.8.10 hrut_otaextrainfo

- 功能说明：

查询 OTA 升级的 warning 信息，warning 信息具体介绍见 3.7 小节

- 实现原理：

直接读取 log 文件 warning 中信息

默认输出 warning 文件的所有信息，输出 1 次

- 使用方法：

参数：l 会过滤掉[OTA_INFO] [OFNI_ATO]

```
1. root@X2SOM3V3:~# hrut_otaextrainfo l
2. wait_before_query: 51
3. estimate: 51
4. msg: The update will take about 51 seconds
5. root@X2SOM3V3:~#
6. root@X2SOM3V3:~#
7. root@X2SOM3V3:~# hrut_otaextrainfo
8. [OTA_INFO]wait_before_query: 51
9. estimate: 51
10. msg: The update will take about 51 seconds[OFNI_ATO]
```

3.9 warning 信息约定

3.9.1 功能介绍

warning 信息中定义 download 工具在升级开始后的查询等待时间、预估升级时间和提示信息。具体定义如下：

- **wait_before_query**：开始升级后，download 工具查询前的等待时间，避免系统升级过程中，download 工具查询引起 OTA 升级失败的情况出现。
(ps: 升级过程中，系统所处的一些状态不能进行查询操作，会导致系统异常。)
- **estimate**：预估的 OTA 升级所需要的总时间，estimate >= wait_before_query。

- **msg:** 提示信息，用于向用户提示该升级包中需要注意的地方，例如：升级需要等待的时间。

例如：约定文件名为 **warning.txt**，在升级包的根目录

1. wait_before_query: 60
2. estimate: 60
3. msg: the update will take about 60 minutes

3.9.2 使用方法

OTA 升级过程中会提取升级包中的 **warning** 信息，**download** 工具通过 **utility hrut_otaextrainfo** 查询并获取 **warning** 相关信息。

注：查询方法详见 3.6.10

3.10 recovery 实现

3.10.1 recovery 功能介绍

- 镜像说明：

kernel 分区的镜像如下：

```
1、image.gz          kernel partition
2、recovery.gz
3、*.dtb
4、dtb-mapping.conf
```

- ◆ **Image.gz**: kernel 镜像 image 的压缩文件，用于正常系统启动；
- ◆ **recovery.gz**: recovery 系统中 image(with cpio)的压缩文件，用于升级失败后恢复；
- ◆ ***.dtb**: dtb 文件，每个开发板的配置不同，通过 dtb 来区分不同的开发板；
- ◆ **dtb-mapping.conf**: board id, gpio id 和 dtb 的索引文件，用于通过 board id 和 gpio id 来查找 dtb 文件。

说明：recovery 系统的镜像为 recovery.gz，包括 Image + ramdisk，保存在 kernel 分区

- recovery 系统的功能：

1) 系统升级

在 golden 单分区模式下，升级失败后会进入 recovery 系统，可以再次进行系统升级，升级次数不受限制，升级成功后会回到正常系统。

可以强制进入 recovery 系统，进行系统升级。

2) 其他操作

可以强制进入 recovery 系统，进行其他操作。例如：emmc 分区操作等。

3.10.2 进入 recovery 模式的方式

1) OTA 升级失败后，golden 方式会进入 recovery 系统。

2) 通过设置标志位 reset_reason 为 recovery，重启后进入 recovery 系统。

进入方法：

1. hrut_reasetreason recovery //设置 resetreason 为 recovery
2. reboot //重启后，进入 recovery 系统

退出方法：

1. hrut_reasetreason normal //设置 resetreason 为 normal
2. reboot //重启后，返回正常系统

3) 启动过程中，通过相关 gpio 的值，可以进入 recovery 系统。（待实现）

例如：可以通过特殊的键组合 电源键 + 音量键

3.11 OTA 升级流程

3.11.1 golden 模式升级流程

golden 模式支持单分区方案，uboot 做双备份，用于 recovery 系统

A: golden 模式正常使用的系统

- 实现原理

- 1) A 为 golden 模式使用的系统，OTA 只升级 A 系统。
- 2) A 升级成功，继续在 A 系统运行；升级失败，则进入 recovery 系统；
- 3) 升级失败后，可在 recovery 系统下再次升级；
- 4) 可以强制进入 recovery 系统，进行升级或其他操作。

- 升级流程

例如：升级 all_in_one.zip

1) 升级包获取：A 系统正在运行，获取升级包 all_in_one.zip，准备升级 A 系统；

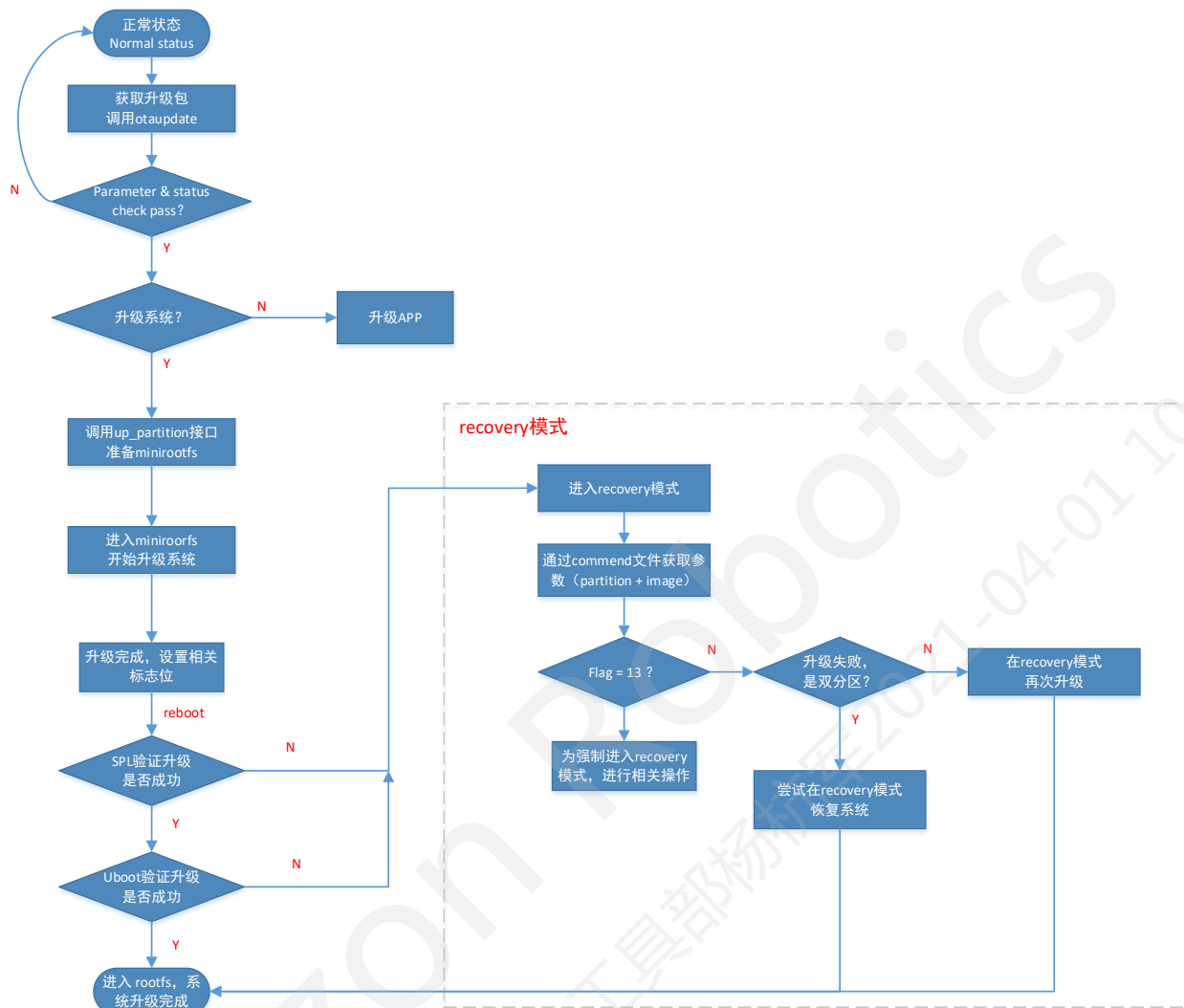
2) 升级：调用 CP 侧接口 otaupdate，使用升级包升级 A 系统，设置相关标志位，创建临时文件系统，chroot 到临时文件系统进行升级；

3) 重启验证：验证 A 是否升级成功，升级成功，继续运行 A；升级失败，进入 recovery 系统恢复系统或再次尝试进行升级；

31) 成功：继续使用 A 系统

32) 失败：重启进入 recovery 系统，在 recovery 系统下，如果为双分区，则尝试恢复系统，如果为单分区支持再次升级。

● 流程图如下



3.11.2 AB 模式升级流程

AB 模式只支持双分区方案，A 系统和 B 系统等价，互为对方的备份

A: 正在使用的系统 B: 未使用的备份系统

● 实现原理:

1) A 为当前正在运行的系统或分区，OTA 升级时升级 B 系统或分区;

注: 这个方案只针对 AB 模式

2) A 升级 B 成功，进行系统分区状态切换，进入 B 系统或分区继续运行; 升级失败，则回退到 A 系统继续运行。

3) 升级失败后，A 系统或分区可以再次升级 B 系统或分区，直到升级成功。

● 升级流程:

例如: 升级 all_in_one.zip

1) 升级包获取: A 系统正在运行，获取升级包 all_in_one.zip，准备升级 B 系统

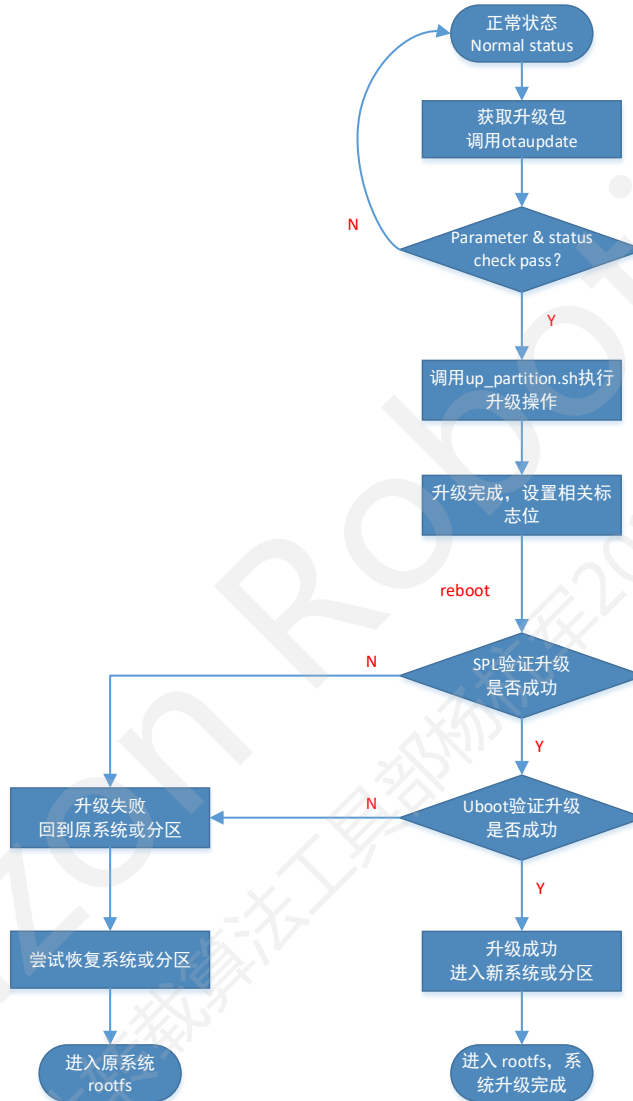
2) 升级: 调用 CP 侧接口 otaupdate，使用升级包升级 B 系统，设置相关标志位，开始升级

3) 重启验证: 验证 B 是否升级成功, 升级成功, 运行 B; 升级失败, 回退到 A 系统;

31) 成功: 切换 A(system) --> B(systembak), 使用 B 系统

32) 失败: 重启回到 A, 再次升级时升级 B, B 升级成功才会使用 B

● 流程图如下:



3.11.3 uboot 中 OTA 实现原理

- 实现原理:

1) 校验 **upmode** 位: UBoot 首先判断系统是否支持 OTA, 如果不支持 OTA, 则跳过 OTA 标志位校验, 正常启动系统。是否支持 OTA 通过 **veeprom** 中的 **upmode** 位进行判断。

- AB: AB 模式, 支持 OTA
- Golden: golden 模式, 支持 OTA
- 其它: 未初始化, 不支持 OTA

2) 校验 **boot_reason** 位: 如果支持 OTA, 则校验 **boot_reason** 位, 根据 **boot_reason** 位判断是否是升级流程。

- normal: 系统正常启动, 正常启动 **kernel** 和 **system**
- recovery: 升级失败或强制进入 **recovery** 系统, 启动 **recovery.gz**
- uboot/kernel/system/all: 升级流程, 进入步骤 3

3) 判断是否升级 **kernel** 和 **system**: 根据 **boot_reason** 判断是否是升级 **kernel** 和 **system**。

- kernel/system/all: 此次 OTA 升级了 **kernel** 或 **system**, 进入步骤 4
- uboot: 此次 OTA 未升级 **kernel** 和 **system**, 正常启动 **kernel** 和 **system**

4) 判断 **flash_success** 位: 根据 **flash_success** 位判断镜像写入是否成功。

- true: 镜像写入 **emmc/nor** 中成功, 进入步骤 5
- false: 镜像写入 **emmc/nor** 中失败, 升级失败。AB 模式启动原分区 (升级前使用的 **kernel** 或 **system** 分区), **golden** 模式启动 **recovery.gz**。

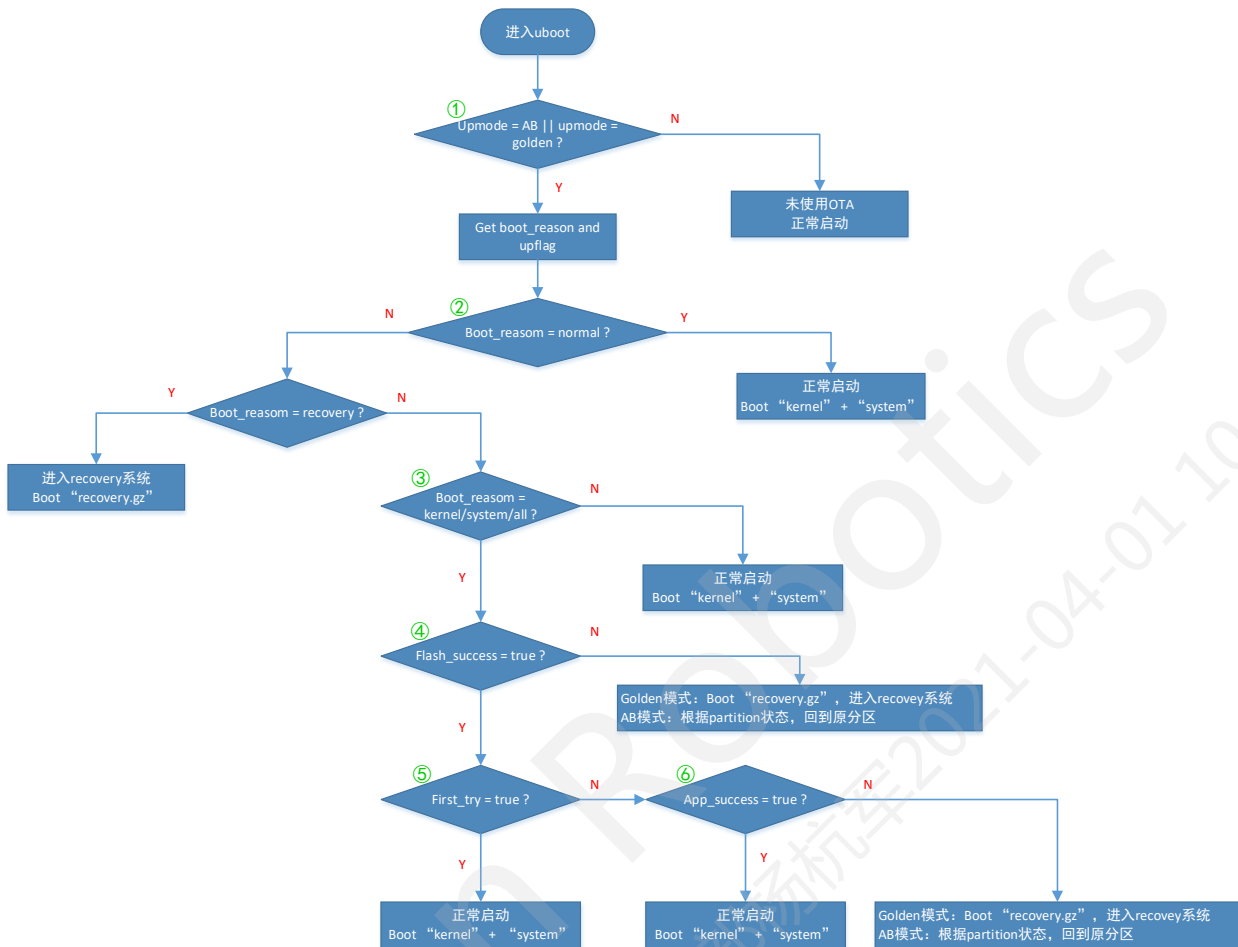
5) 判断 **first_try** 位: 根据 **first_try** 位判断是否是第一次启动。

- true: OTA 升级后第一次启动, 正常启动 **kernel** 和 **system**
- false: OTA 升级后第一次启动失败, 进入步骤 6

6) 判断 **app_success** 位: 根据 **app_success** 位判断 **app** 是否启动成功。

- true: 正常启动 **kernel** 和 **system** (ps: 目前 OTA 正常升级流程, 不会走到这个位置)
- false: OTA 升级失败, AB 模式启动原分区 (升级前使用的 **kernel** 和 **system** 分区), **golden** 模式启动 **recovery.gz**。

● 流程图如下：



3.12 OTA secure 方案

3.12.1 版本校验

1) 功能说明:

OTA 升级前会进行版本校验, 验证升级包的版本是否低于系统的版本, 如果低于系统版本, 则停止升级。(ps: 防止版本回退)

注:

- (1) 提供版本降级方案, 使用 disk.img 的升级包跳过版本校验;
- (2) 如果升级包中不包含 version 文件, 则跳过版本校验。

2) 数据定义:

约定版本文件名为 version, 位于升级包根目录, 具体内容如下:

1. x2j2_lnx_db_2019XXXXX release
2. x3j3_lnx_wb_20XXXXXX release

3.12.2 分区校验

1) 功能说明:

OTA 升级前对镜像中的 GPT 分区校验, 通过与当前系统的 GPT 分区进行对比, 验证 GPT 分区是否调整, 如果 GPT 分区有调整, 则停止升级。提醒用户 GPT 分区已经调整, 分区升级包不能升级系统。需要使用 disk.img 的升级包升级系统。

注: nor flash 中没有 GPT 分区, 升级跳过分区校验

2) 数据定义:

约定分区文件为 gpt.conf, 具体内容如下:

1. veeeprom:34s:37s
2. sbl:38s:293s
3. ddr:294s:2597s
4. uboot:2598s:4645s
5. kernel:4646s:45605s

3.12.3 签名校验

1) 功能描述:

OTA 支持对升级包的签名进行校验, 使用 RSA2048 SHA256 算法对升级包的每个文件进行签名验证, 验证不通过则停止升级, 确保升级包的完整性。

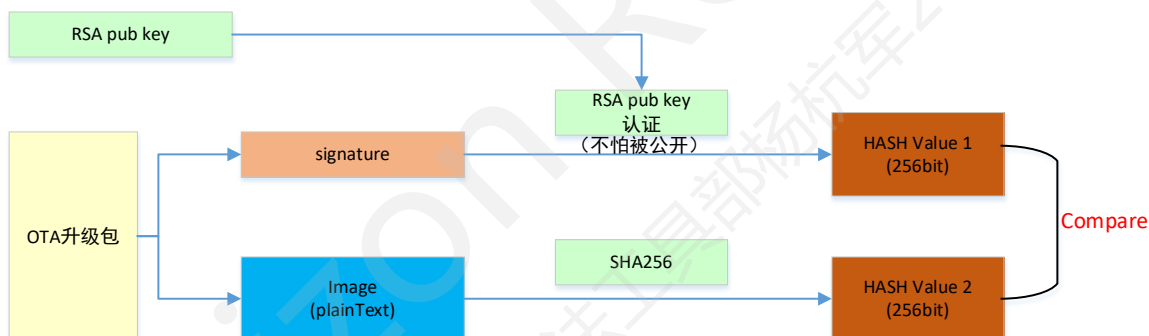
说明: 目前未开启强制校验功能, 如果升级包没有签名或签名验证失败, 只有 warning 提示信息。后面会在一个时间点开启强制校验功能。

2) 数据定义:

1. RSA-SHA256(gpt.conf)=
b0f7cd46a065529a25917167efce092ab1cc2387ca68aba1e6313108a3abf90c5162fb5e4fdf3da1617237e8c06370fe842b33b1cc79debfb8a80b92d5762e15e8d1d7a5da6c2aebb1074bcfcec6b60f6c5a29c72924f59a8b1df5c06819b5d904e0a18ab5110ea4713afec5fbc58a1a0f4255934684691409787b661f7254f95
2. RSA-SHA256(hobot-x2-som-ipc.dtb)=
1e7313f4b931e5680efcf6bd2adae28dd6f16920a83c9b072d9a8942232b2068ba34acce8e00e3d017ba30356c3b0524b9d68b878edaf9d6fad04b62641958beb9de3b92139f4d445d83c2d70706f0ddada38f6ad9b3bebd6e90f67c4511a670cd982613ce4bbafede98e37ec651c2df1746408ad61c3bfe3c04ec13b2af90a
3. RSA-SHA256(version)=
b109c3e04ccdfc23d793ef3ef5ce71d7180df6b57026d8f4cf4079131ee9fdea1dc08acc3afcedf5b15ac67fbc783e83d5c3d80a4069b2fbbcbdb6957abb77947c8034b3a8fa399ca3ee6d8b966a8181077effad74ae7190986ac6704e55a99d2304989fa156d64db30a25661b3b3fed0c09fab04e1a545a762498dcd75fe448b
4. RSA-SHA256(Image.gz)=
7bb3d740e3e26a334ec501bacc105da5583b40c0835715534d2c69c7d367e25c30bd3557d2f44e41c8bce4aae348c38b6f4c1f978a237600f4d5619595b2d1e27d985279da545d8f80a872724af591abfe1a0cbbee5b11d02c082720c987d26271baaf336a62ac4cf04dabc945b6dff5a853f050f2472b43b7142aeba420ee97

● 签名校验流程:

RSA pub key 来自系统本身，对升级包中的每个文件进行校验。



4 OTA 升级获取方法

4.1 发布的 SDK 包

1) 镜像包，可以使用如下方法获取 OTA 包

通过 gen_ota.sh 脚本获取对应板子的 OTA 升级包

gen_ota.sh 用法（执行之前先给脚本执行权限：chmod +x gen_ota.sh）：

X2J2:

1. usage: \$0 <-b board> <-t Debug/Release> [-u] [-f ddr_frequence] [-v ddr_vendor] [-o output_file] [-x]
2. available board: x2ipc/x2dev/x2dev512m/j2dev/x2somfull/x2svb/j2mono/j2sk/j2mm-s202/96board/j2quad
3. -x means generate 32bit image, without '-x' will generate 64bit image
4. ex: build x2dev Debug UT disk image: ./gen_disk.sh -b x2dev -t Debug -u
5. build j2dev Release disk image: ./gen_disk.sh -b j2dev -t Debug
6. build j2sk Debug disk image with ddr frequency 2666 and ddr vendor hynix:
7. ./gen_disk.sh -b j2sk -t Debug -f 2666 -v hynix

X3J3:

8. ./gen_ota.sh: illegal option -- h
9. usage: ./gen_ota.sh <-b board> [-s] [-u] [-t debug/release] [-l] [-o output_file] [-p][-d]
10. available board : xj3
11. xj3 support ddr4 and lpddr4 board, ddr4 development board needs burn chip id
12. optimize boot : -l reduce boot log for performance test and projects that are needed
13. dual_boot : support AB Boot image
14. when switching between normal and AB boot, -b xj3 must be used to update gpt
15. ex: build xj3 OTA UT disk pkg: ./gen_ota.sh -b xj3 -u
16. build xj3 OTA normal disk pkg: ./gen_ota.sh -b xj3
17. build xj3 OTA secure disk pkg: ./gen_ota.sh -b xj3 -s
18. build xj3 OTA UT disk pkg: ./gen_ota.sh -b xj3 -u
19. build xj3 OTA AB Boot disk pkgs: ./gen_ota.sh -b xj3 -d -p
20. build xj3 OTA partition upgrade pkgs: ./gen_ota.sh -p

2) SDK 源码包，可以直接编译 build 系统，会自动生成 OTA 包

4.2 ota 打包工具

1) OTA 打包工具编译生成 OTA 升级包

- 功能说明:

OTA 打包工具用于升级 OTA 升级包, 包括分区升级包和 disk.img 升级包, disk.img 升级包仅 eMMC 镜像支持。

- 代码路径: [build/ota_tools](#)

- 编译方法:

```
1. Usage: ./make_update_package.sh [-b emmc | nor | nand] [-e all | disk]
2. Options:
3.     -b  choose board type
4.         'emmc'  emmc board type
5.         'nor'   nor board type
6.         'nand'  nand board type
7.     -e  build image
8.         'all' means build partition update package
9.         'disk' means build disk.img update package
10.    -h  this help info
```

说明:

all: 正常的升级包, 包括 uboot.zip, kernel.zip, system.zip 和 all_in_one.zip

disk: 特殊的升级包, 支持版本升级和版本降级, 升级包包含 disk.img + update.sh

注: disk 方案升级板子中镜像会全更新, 这时 recovery 系统也会被更新, 不能保证升级失败后进入 recovery 系统。

- OTA 打包工具获取升级包的步骤如下:

a) 先编译整个工程, 生成 disk.img, 确保打包的镜像是最新的。

b) 执行脚本:

emmc all 升级文件: ./make_update_package.sh -b emmc -e all

emmc disk 升级文件: ./make_update_package.sh -b emmc -e disk

nor all 升级文件: ./make_update_package.sh -b nor -e all

nor disk 升级文件: ./make_update_package.sh -b nor -e disk

获取镜像: ota_tools 目录和 out/target/deploy/目录

5 OTA 升级方法

5.1 otawrite 接口

- 升级接口:

UBoot 下 OTA 升级接口, 用于 UBoot 模式更新系统镜像。

代码路径:

[uboot/cmd/ota.c](#)

- 使用方法:

- (1) 首先将待升级文件传输到内存中, 例如: `bifrw` 或 `tftp` 传输文件到 `0x4000000`,
- (2) 执行 `otawrite`
 1. `otawrite uboot 0x4000000 0x100000 emmc`
- (3) 调用 `otawrite` 成功后, 镜像会自动写到对应的分区

- 接口定义:

1. Hobot>otawrite
2. Usage:
3. `otawrite <partition name> <ddr addr> <image size> [emmc | nor]`
4. – emmc partition name:
5. [all | gpt-main | sbl | ddr | uboot | kernel | system | app | gpt-backup]
6. – nor partition name:
7. [all | uboot | kernel | system | app]
8. – image size:
9. bytes size [Example: 0x8000]
10. – emmc | nor:
11. options, write emmc or nor partition
12. **default:** writing device depend on bootmode
13. - example:
14. `otawrite uboot 0x4000000 0x100000`
15. - version:
16. [2019-08-02]

5.2 AP 侧升级接口

- 升级接口:

AP 侧 ota 升级工具为:

`hrsom_otaupdate`

- 使用方法:

- (1) 首先将待升级文件拷贝到 AP 侧, 例如 `/mnt/all.zip`,
- (2) 执行 `hrsom_otaupdate`
 2. `hrsom_otaupdate all /mnt/all.zip`
- (3) 调用 `hrsom_otaupdate` 成功后, CP 自动进入升级模式进行升级, 期间可以使用
 1. `hrsom_otaquery progress` //查询升级进度
 2. `hrsom_otaquery checksuccess` //查询升级结果

注意：hrsom_otaupdate 依赖 hbipc，使用之前要将相关的驱动模块载入。

5.3 CP 侧升级接口

- 升级接口：

CP 侧 ota 升级工具为：

otaupdate

说明：otaupdate 接口功能详见 3.5.2 小节

- 使用方法：

- (1) 获取升级包

根据升级的内容不同，获取不同的 OTA 升级包，升级包获取方法见本文第 4 小节。

- (2) 传镜像到 CP 侧

升级包可以传输到 userdata 目录

tftp 方式：tftp 192.168.1.187 -gr all_in_one.zip /userdata/all_in_one.zip

hbipc 方式：hbipc-utils put /root/mount/all_in_one.zip /userdata/all_in_one.zip

- (3) 调用接口开始升级

otaupdate all all_in_one.zip

说明：

OTA 执行完升级后，会重启一次系统，验证升级是否成功

例如：all_in_one.zip 升级的部分 log 如下

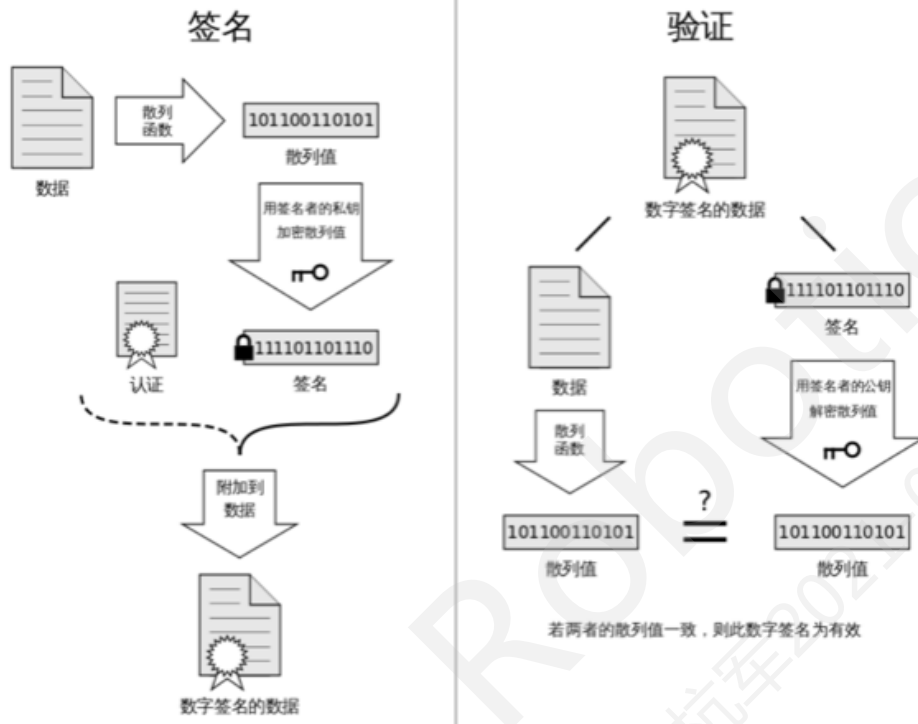
```
1. root@X2SOM1V8:~# otaupdate all /userdata/all_in_one.zip
2. [OTA_INFO] boot mode: emmc
3. [OTA_INFO] parameter check success
4. [OTA_INFO] 5%
5. [OTA_INFO] get ota_upmode golden
6. [OTA_INFO] upgrade_need_space: 33339
7. [OTA_INFO] tmp_free_space: 95916
8. [OTA_INFO] userdata_free_space: 3014412
9. [OTA_INFO] 10%
10. [OTA_INFO] get ota_upmode golden
11. [OTA_INFO] check update mode golden success
```

5.4 APP 升级方法

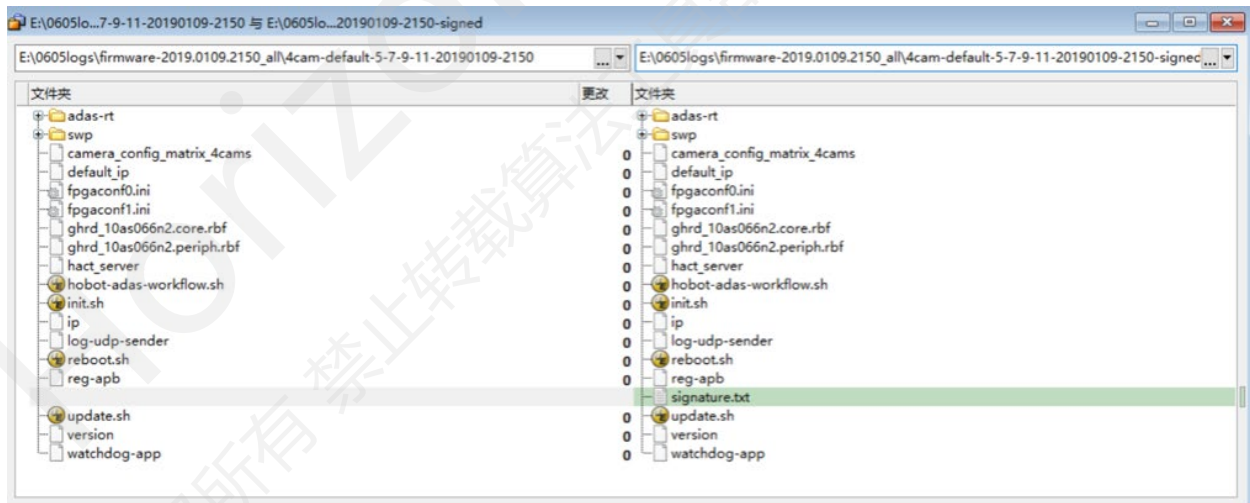
5.4.1 APP 签名方法

- 实现原理：

签名服务主要是为了确保镜像的一致性（确保其没有被 hack 或者篡改），在原文后附上签名数据。在这里，X2J2 的升级镜像使用了 RSA2048_SHA256 算法的摘要签名方式。可使用内网的签名服务，对升级包内的每个文件进行摘要计算并签名，签名结果存放到 meta 文件中嵌入到升级包中。终端使用升级服务，镜像的签名会按照验证流程进行验证。



签名和未签名包的差别如下：



5.4.2 APP 升级约定

- **app service 的脚本：**

X2J2 和 X3J3 Boot 起来通过 `appinit.sh` 来加载应用服务(app service)的脚本

下面的两个脚本由应用的开发者提供，会放置在其 app_*.zip 发布包内

(1) 启动脚本

默认约定 app 的脚本为 init.sh （此脚本 位于/mnt or /app 下）

(2) 停止脚本 （主要给 OTA 升级用, 有些 App 有自己的 watchdog, 需要合理的关闭）

默认约定 app 的脚本为 deinit.sh （位于 init.sh 同目录下）

● app-*.zip OTA 包的约定:

可使用 hupdate GUI 图形工具升级的 OTA 的包符合下面特点

1、升级包的名字带 app 前缀， 比如 app-autodrive-2019xxxx.zip

2、ZIP 包的根目录有两个重要文件（如下截图）

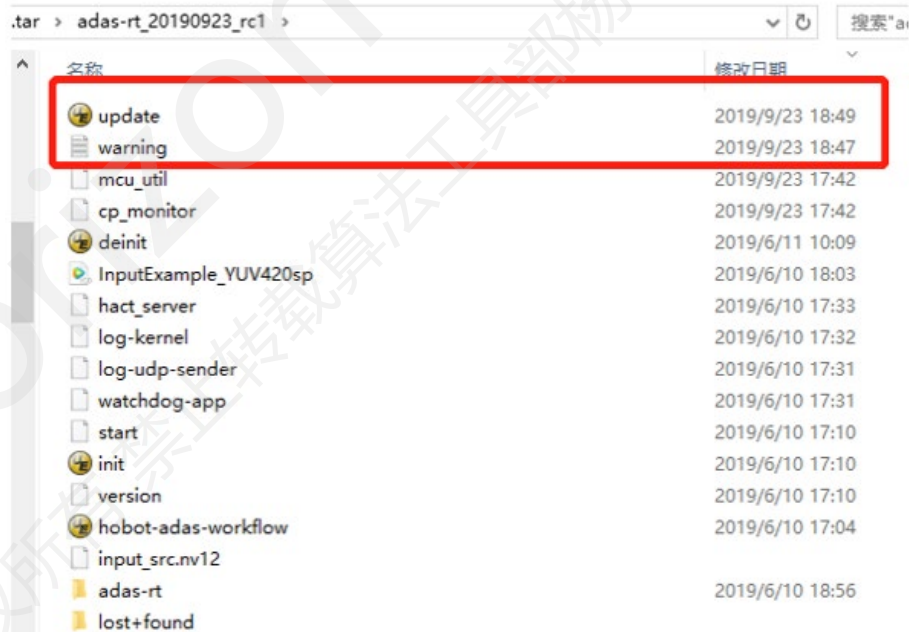
A、**update.sh** 用来控制升级要 cp 到 J2 系统的细节（使用/mnt 的应用开发者直接重用，也可加入升级需要的特殊操作）

可以参考文后的实际测试 App 包来了解里面的内容

B、**warning.txt** 用来显示升级过程中要显示的一些细节 以及 此包升级重启需要的时间（需>=真实包处理需要的时间）

默认例子的 App 升级需要 45 秒，此配置文件使用如下参数

- wait_before_query: 45 #从 zip 文件下载到板子完成后开始计算，到升级重启完成的典型时间
- estimate: 45 #从 zip 文件下载到板子完成后开始计算，到升级重启完成的最长时间，一定大于等于 wait_before_query（一般作为最大的超时时间）
- msg: the update will take about 45 seconds ##提醒给用户看的消息（单行）



名称	修改日期
update	2019/9/23 18:49
warning	2019/9/23 18:47
mcu_util	2019/9/23 17:42
cp_monitor	2019/9/23 17:42
deinit	2019/6/11 10:09
InputExample_YUV420sp	2019/6/10 18:03
hact_server	2019/6/10 17:33
log-kernel	2019/6/10 17:32
log-udp-sender	2019/6/10 17:31
watchdog-app	2019/6/10 17:31
start	2019/6/10 17:10
init	2019/6/10 17:10
version	2019/6/10 17:10
hobot-adas-workflow	2019/6/10 17:04
input_src.nv12	
adas-rt	2019/6/10 18:56
lost+found	

5.5 下载工具中的 OTA 实现升级

● 功能说明:

下载工具支持 UART, UBoot 和 OTA 3 种升级模式，支持 IPC, J2dev, X2dev, J2mono 等板子的镜像更新，

支持 X3J3 开发板更新。方便对开发板中的镜像进行烧写。

模式说明：

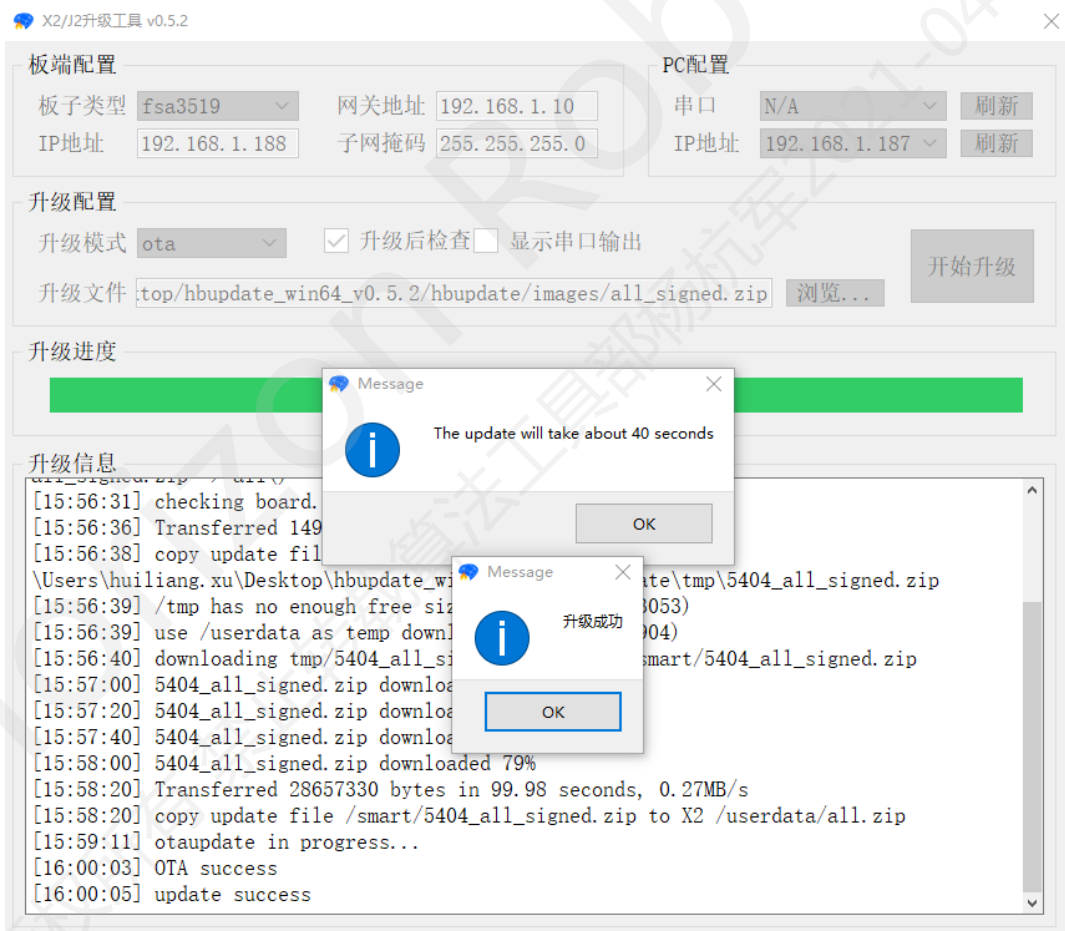
- **UART 模式：**UART 模式用于空板烧写，或开发板变砖后烧写镜像。UART 模式速度较慢，需要通过 UART 传输镜像到 CP 侧，一步步启动系统，然后在 UBoot 下烧写镜像。应优先使用 UBoot 和 OTA 模式。
- **UBoot 模式：**板子能启动到 UBoot，则可使用 UBoot 模式，更新单分区或系统镜像。
- **OTA 模式：**板子能启动到 Kernel 并进入命令行，可以使用 OTA 模式更新系统镜像。OTA 升级包的获取方式详见第 4 小节。

● 使用说明：

具体使用方法参考文档：《地平线产品升级工具使用手册》

● 使用示例：

使用下载工具在 OTA 模式下升级 FSA3519 的 all_in_one.zip 镜像



6 参考文献

- 1) [Power State Coordination Interface PDD v1.1 DEN0022D](#)
- 2) [Trusted Firmware-A - version 2.1](#)