全志科技
Allwinner Technology

Tina

OTA 使用和开发 v1.5

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2016/8/19		初始版本
V1.1	2016/10/27		R40 方案第二次 review
V1.2	2017/11/06		Tina v2.5
V1.3	2018/01/16		R30 方案
V1.4	2018/04/13		Tina v3.0
V1.5	2018/06/07		Tina v3.0.1

目　录

1. 概述概述.....4
1.1. 编写目的.....4
1.2. 适用范围.....4
1.3. 相关人员.....4
2. OTA 方案选择.....5
2.1. 小容量方案.....5
2.2. 大容量方案.....5
2.3. misc-upgrade.....5
2.4. OTA 升级流程.....6
2.4.1. 基本步骤.....6
2.4.2. 特殊步骤.....6
2.4.3. 中途掉电.....6
3. 分区处理.....7
3.1. 分区定义.....7
3.2. 分区大小配置.....7
3.2.1. 配置 boot 分区大小.....7
3.2.2. rootfs 分区的大小.....8
3.2.3. extend 分区的大小.....8
3.2.4. 其他分区.....8
3.2.5. UDISK 分区.....8
3.2.6. 其他说明.....8
4. misc-upgrade 升级.....9
4.1. misc-upgrade 构成.....9
4.2. OTA 镜像包编译.....9
4.3. 小机端 OTA 升级命令.....10
4.3.1. 大容量 flash 方案.....10
4.3.2. 小容量 flash 方案.....11
5. 脚本接口说明.....12
5.1. 实现联网逻辑.....12
5.2. 请求下载目标镜像.....12
5.3. 开始烧写分区状态.....12
5.4. 写分区完成.....12
5.5. -f (-n)调用顺序.....13
5.6. -p 调用顺序.....13
6. 相关系统状态读写.....14
7. OTA 配置.....15
8. 对 overlayfs 的处理.....16
9. 对 busybox-init 的处理.....17
10. 常见问题.....18
10.1. OTA 时出现 SQUASHFS ERROR.....18
10.2. 编译 OTA 包之后，正常编译出错.....18
10.3. 是否可更新 boot0/uboot.....18
11. Declaration.....19

1. 概述概述

1.1. 编写目的

本文主要服务于使用 Tina 软件平台的广大客户，以冀帮助客户使用 Tina 平台的 OTA 升级系统并做二次开发。

1.2. 适用范围

Allwinner 软件平台 Tina
Allwinner 硬件平台 R6 R11 R16 R18 R30 R40

1.3. 相关人员

适用 Tina 平台的广大客户和关心 OTA 的相关人员。

2. OTA 方案选择

OTA 是 Over The Air 的简称，顾名思义就是通过无线网络从服务器上下载更新文件对本地系统或文件进行升级，便于客户为其用户及时更新系统和应用以提供更好的产品服务，这对于客户和消费者都极其重要。

由于在实际应用中，存储操作系统和持久文件的存储介质（如 nand、emmc、spinor）大小各异，在 OTA 中需要单独在存储介质上开辟 recovery 分区[1]，以防备在更新中意外断电，造成系统更新失败无法重启的问题。

所以在选择 OTA 方案时一定要考虑到 recovery 分区大小对分区规划的影响，避免在小容量时 recovery 分区太大导致分区规划难题。

综上因素，我们在 Tina 上针对大容量和小容量设计了不同的方案。

2.1. 小容量方案

小容量介质一般指存储介质容量小于 32M（一般为 spi nor）。

在命令行中进入 Tina 根目录，执行命令进入配置主界面：

```
source build/envsetup.sh    (详见①)
lunch                       (详见②)
make menuconfig             (详见③)
```

详注：

① 加载环境变量及 tina 提供的命令

② 输入编号，选择方案

③ 进入内核配置主界面(对一个 shell 而言，前两个命令只需要执行一次)

配置路径：

```
Target Images
  └> *** Image Options ***
       [*] For storage less than 32M, enable this when using ota
```

选中该配置项后，rootfs 的 /usr 会被分拆出一部分生成 usr.squashfs(usr.img)，并建立软链接 usr.fex。通过配置分区表将 usr.fex 放在 extend 分区，开机后自动挂载到 usr 目录。这种设置的目的是可与 recovery 镜像（boot_initramfs）复用该分区，以此起到节省存储空间的作用。

因此小容量方案中，并无单独的 recovery 分区，而是在 OTA 升级时与 extend 分区复用。

2.2. 大容量方案

大容量介质一般指存储介质容量大于 32M（一般是 nand、emmc）。

对于大容量方案，不建议选中小容量方案中所述配置项，即不需要 usr.img 和 extend 分区，而只需要添加 recovery 分区，这样在 OTA 升级时会省去很多麻烦。

2.3. misc-upgrade

不管是小容量还是大容量，都要在 make 之前选中应用包 misc-upgrade：

```
make menuconfig
  └> Allwinner
       └> <*> misc-upgrade..... read and write the misc partition
```

misc-upgrade 包主要功能是从指定服务器下载更新镜像到本地，然后升级相应分区，期间会向 misc 分区[2]写入升级阶段的标志，出现意外无法重启时 uboot 或内核（如果能够启动）可以根据 misc 分区的状态标志进行下一步的决策。

[1] 需修改 sys_config.fex，见本文 3.2.1 配置 boot 分区大小 章节

[2] misc 分区用于存储状态信息，在 OTA 升级过程中并不会擦除，该分区读写见本文第 6 章相关系统状态读写

2.4. OTA 升级流程

2.4.1. 基本步骤

Tina3.0 及之前版本

1 .备份 busybox 等资源到 ram 中，使得 OTA 过程不依赖 rootfs
2 .设置开始 OTA 的标志，upgrade_pre
3 .将 recovery 系统写入 recovery 分区
4 .设置标志 boot-recovery
5 .更新 boot 和 rootfs 分区
6 .设置标志 upgrade_post
7 .对于小容量方案，更新 usr 分区
8 .设置标志 upgrade_end
9 .重启，重启后为新系统

Tina3.0.1

1.设置开始 OTA 的标志，upgrade_pre
2.将 recovery 系统写入 recovery 分区
3.设置标志 boot-recovery
4.主动重启，重启后进入 recovery 系统
5.更新 boot 和 rootfs 分区
6.设置置标志 upgrade_post
7.对于小容量方案，更新 usr 分区
8.设置标志 upgrade_end
9.重启，重启后为新系统

2.4.2. 特殊步骤

对于使用 busybox init 启动的平台，如 R6 等，其 rootfs_data 分区是启动后拷贝 rootfs 中的 etc 目录生成的，OTA 之后，rootfs 发生了变化，需要重新生成 rootfs_data 分区的内容。

因此，在 upgrade_end 之前新增了一个状态，upgrade_etc.重启后 busybox-init 的启动脚本会判断此标志，如果启动是标志为 upgrade_etc，则进行 etc 分区文件的更新，更新后设置系统状态为 upgrade_end。

2.4.3. 中途掉电

OTA 中途掉电后，下次启动会根据标志，继续完成 OTA

当设置 upgrade_pre 标志之后掉电，重启后仍是旧系统，从该标志之后的步骤开始执行

当设置 boot-recovery 标志之后掉电，重启时，uboot 会判断到这个标志，并直接启动 recovery 系统，启动后，从该标志之后的步骤开始执行

当设置 upgrade_post 标志之后掉电，重启时后为新系统，从该标志之后的步骤开始执行

3. 分区处理

3.1. 分区定义

升级分区	
boot 分区	存内核镜像
rootfs 分区	基础系统镜像分区（/lib, /bin, /etc, /sbin 等非/usr，非挂载其他分区的路径，wifi 支持环境，alsa 支持环境、OTA 环境）
extend 分区	扩展系统镜像分区（/usr 应用分区）[仅小容量方案有]
recovery 分区	存放恢复系统镜像 [仅大容量方案有]
不升级分区（包括但不限于如下分区）	
private 分区	存储 SN 号分区
misc 分区	系统状态、刷机状态分区
UDISK 分区	用户数据分区（/mnt/UDISK）
overlayfs 分区	存储 overlayfs 覆盖数据

注:升级分区指升级时会更新的分区

3.2. 分区大小配置

3.2.1. 配置 boot 分区大小

boot 分区镜像的大小依赖内核配置，必须小于等于 sys_partition.fex/sys_partition_nor.fex[3]中定义的 boot 分区大小。

boot 分区镜像大小设定：

```
make menuconfig
  └-> Target Images
    └-> *** Image Options ***
      (4) Boot (SD Card) filesystem partition size (in MB)
```

boot 分区大小设定：

```
[partition]
    name            = boot
    size            = 8192[4]
    downloadfile    = "boot.fex"
    user_type       = 0x8000
```

对于大容量方案，需要在 sys_partition.fex 中添加 recovery 分区：

```
;   recovery 分区说明
;   如果启用了 OTA 升级，需要去掉下面 recovery 分区的注释以提供恢复分区存储恢复系统镜像，默认以 boot_initramfs.img 作为 recovery.fex

;[partition]
;    name            = recovery
;    size            = 32768
;    downloadfile    = "recovery.fex"
;    user_type       = 0x8000
```

其中 recovery.fex 是生成的 boot_initramfs.img 软链接而成。

[3] sys_partition.fex/sys_partition_nor.fex 位于 tina//target/allwinner/xxx/configs/

[4] 单位：扇区，此处扇区大小为 512Bytes，因此分区大小为 8192*512/1024/1024=4MB

3.2.2. rootfs 分区的大小

rootfs 分区不需要通过 make menuconfig 去设定，直接根据镜像大小修改分区文件即可。

3.2.2.1. 小容量

对于一些小容量 flash 的方案（如 16M），需在/bin 下存放联网逻辑程序、版本控制程序、下载镜像程序、播报语音程序以及语音文件（这些文件在编译时应该 install 到/bin 或者/lib 下），可以在固件编译完后，查看 rootfs.img 的大小再决定 sys_partition.fex 中 rootfs 分区的大小。

3.2.2.2. 大容量

对于大容量 flash 的方案（如 128M 以上，或者有足够的 flash 空间存相关镜像），不需要小容量中那些 OTA 额外的程序，直接查看 rootfs.img 的大小设定分区文件即可。

3.2.3. extend 分区的大小

extend 分区用于小容量方案下 boot_initramfs.img 和 usr.img 的复用，其大小需要考虑多个方面：

1. 编译后 usr.img 的大小
2. make_ota_image 后 initramfs 镜像的大小[5]

因此，extend 分区略大于 boot_initramfs.img 和 usr.img 两个的最大值，并把 extend 分区的大小值设置为 initramfs 镜像的大小：

```
make menuconfig
  └─> Target Images
      └─> *** Image Options ***
          (8) Boot-Recovery initramfs filesystem partition size (in MB)
```

3.2.4. 其他分区

如 private、misc 等使用默认的大小即可。

3.2.5. UDISK 分区

sys_partition.fex 中不指定 UDISK 分区大小，则剩下的空间全部自动分配进入 UDISK 分区。需要注意的是，因为 OTA 过程会在里面写一些中间文件，所以一定要留取一定空间给 UDISK 分区，至少可以格式化，挂载，而小容量 flash 的方案，也要保证有 256K~512K 的空间。

3.2.6. 其他说明

在 OTA 升级过程中并不能修改上述分区的修改，因此应在满足分区大小条件限制(如 3.2.1-3.2.3)的情况下尽可能留有足够的空余空间，以满足 OTA 升级添加内容的需求。

修改分区大小时，尽量对齐到所用存储介质的块大小。

对于大容量，使用 recovery 分区，对应的，其 env 定义中，boot_recovery 命令需定义为从 recovery 分区启动系统。对于小容量，使用 extend 分区，对应的，其 env 定义中，boot_recovery 命令需定义为从 extend 分区启动系统。

[5] 见本文第 4 章 misc-upgrade 升级

4. misc-upgrade 升级

4.1. misc-upgrade 构成

misc-upgrade 是 Tina 下的一个应用，其路径为：

```
tina/package/allwinner//misc-upgrade
```

Makefile 符合 tina 安装包的书写规范。

misc-upgrade 目录结构如下：

```
.
├─── aw_fstab.init
├─── aw_reboot.sh
├─── aw_upgrade_autorun.init
├─── aw_upgrade_image.sh
├─── aw_upgrade_log.sh
├─── aw_upgrade_normal.sh
├─── aw_upgrade_process.sh
├─── aw_upgrade_utils.sh
├─── aw_upgrade_vendor_default.sh
├─── Makefile
├─── readme.txt
└─── tools
    ├─── Makefile
    ├─── misc_message.c
    ├─── misc_message.h
    ├─── read_misc.c
    └─── write_misc.c
```

其中 aw_upgrade_normal.sh 是专用于大容量的方案，而 aw_upgrade_process.sh 是用于小容量的方案。

4.2. OTA 镜像包编译

在命令行中进入 Tina 根目录，执行命令进入配置主界面（环境配置）：

```
source build/envsetup.sh          (详见①)
lunch                             (详见②)
make ota_menuconfig（可选）        (详见③)
```

详注：

① 加载环境变量及 tina 提供的命令

② 输入编号，选择方案

③ 进入 OTA config 配置界面。直接保存退出即可。

此步骤可选，目的是解决开发过程中 defconfig_ota 未能及时更新而可能引发的编译问题

编译命令：

```
make_ota_image                    (详见①)
make_ota_image --force            (详见②)
```

详注：

① 在新版本代码已经成功编译出烧录固件的环境的基础上，打包 OTA 镜像

② 重新编译新版本代码，然后再打包 OTA 镜像

执行 make_ota_image 之前，可通过 make ota_menuconfig 对 ota 的恢复系统镜像 boot_initramfs.img 进行配置，可根据实际情况，配置 ota 恢复系统包含的功能。

如以下配置支持 ramdisk 并选用 xz 压缩 cpio：

```
make ota_menuconfig
```

```
└> target Images
    └> [*] ramdisk
        └> --- ramdisk
            └> Compression (xz)    --->
```

OTA 镜像包路径为：

```
tina/out/xxx/ota/
```

目录结构如下：

```
.
├── ramdisk_sys
│   └── boot_initramfs.img.md5
├── ramdisk_sys.tar.gz
├── target_sys
│   ├── boot.img
│   ├── boot.img.md5
│   ├── rootfs.img
│   └── rootfs.img.md5
└── target_sys.tar.gz
```

其中，".img.md5" 是 ".img" 的校验值文件

4.3. 小机端 OTA 升级命令

必选参数：-f -p 二选一

```
aw_upgrade_process.sh -f    升级完整系统（内核分区、rootfs 分区、extend 分区）
aw_upgrade_process.sh -p    升级应用分区（extend 分区）
```

可选参数: -l，-d -u, -n

注：对于大容量，用 aw_upgrade_normal.sh 替代 aw_upgrade_process.sh，且一般用-f 参数而不用-p

4.3.1. 大容量 flash 方案

可以使用本地镜像测试，如主程序下载校验好 2 个镜像后（ramdisk_sys.tar.gz、target_sys.tar.gz），存在/mnt/UDISK/misc-upgrade 中，调用如下命令。

OTA 升级期间掉电，重启后升级程序也能自动完成烧写，不需要依赖联网重新下载镜像。

4.3.1.1. -l 选项

```
-l <路径>
```

如：

```
aw_upgrade_normal.sh -f -l /mnt/UDISK/misc-upgrade
```

(注：mnt 前的根目录"/"最好带上，misc-upgrade 后不要带"/")（-l 参数，默认使用压缩镜像包）

不使用-n 参数的方案需要部署上服务器上的镜像是：ramdisk_sys.tar.gz、target_sys.tar.gz

4.3.1.2. -n 选项

如：

```
aw_upgrade_normal.sh -f -n -l /mnt/UDISK/misc-upgrade
```

一般情况下不使用-n 选项，而是下载 ramdisk_sys.tar.gz、target_sys.tar.gz。但对于某些 ram 较小的平台，如 R6 spinand 的情况，flash 的容量足够放下大容量的 OTA 包，但升级过程可能因为 ram 不足而失败。对于这种情况，可以选择下载未压缩的数据，即 ramdisk_sys.tar.gz、target_sys.tar.gz 解压出来的所有内容。

将多个分区数据文件下载到/mnt/UDISK/misc-upgrade 中，调用上述命令。

使用-n 参数的方案需要部署上服务器上的镜像是：boot_initramfs.img、boot.img、rootfs.img 及其对应的 md5 后缀的校验文件

4.3.2. 小容量 flash 方案

不能使用-l 参数，升级区间出错重启后，需要联网下载程序获取镜像。

4.3.2.1. -d -u -n 选项

-d arg1 -u arg2

同时使用，-d 参数为可以 ping 通的 OTA 服务器的地址、-u 参数为镜像的下载地址

-n

-n 一些小 ddr 的方案（如剩余可使用内存在 20m 以下的方案），可以使用这个参数，shell 会直接请求下载不压缩的 4 个 img 文件，这样子设备下载后不需要 tar 解压，减少内存使用。

如：

```
aw_upgrade_process -f -d 192.168.1.140 -u http://192.168.1.140/
```

升级 shell 会先 ping -d 参数（ping 192.168.1.140），ping 通过后，会根据升级命令和系统当前场景请求下载：

无-n 参数：

```
http://192.168.1.140/ramdisk_sys.tar.gz
http://192.168.1.140/target_sys.tar.gz
http://192.168.1.140/usr_sys.tar.gz
```

有-n 参数：

```
http://192.168.1.140/boot_initramfs.img
http://192.168.1.140/boot.img
http://192.168.1.140/rootfs.img
http://192.168.1.140/usr.img
```

使用-n 参数的方案需要部署上服务器上的镜像是：boot_initramfs.img、boot.img、rootfs.img、usr.img 及其对应的 md5 后缀的校验文件

不使用-n 参数的方案需要部署上服务器上的镜像是：ramdisk_sys.tar.gz、target_sys.tar.gz、usr_sys.tar.gz

5. 脚本接口说明

对于小容量 flash 的方案，没有空间存储镜像，相关镜像只会存在 ram 中，断电就会丢失。
假如升级过程断电，需要在重启后重新下载镜像。
aw_upgrade_vendor.sh 设计为各个厂家实现的钩子，SDK 上只是个 demo 可以随意修改。

5.1. 实现联网逻辑

```
check_network_vendor(){
    return 0  联网成功（如：可以 ping 通 OTA 镜像服务器）
    return 1  联网失败
}
```

5.2. 请求下载目标镜像

$1：ramdisk_sys.tar.gz $2：/tmp

```
download_image_vendor(){
    # $1 image name    $2 DIR    $@ others
    rm -rf $2/$1
    echo "wget $ADDR/$1"
    wget $ADDR/$1 -P $2
}
```

5.3. 开始烧写分区状态

```
aw_upgrade_process.sh -p  主动升级应用分区的模式下，返回 0 开始写分区    1 不写分区
aw_upgrade_process.sh -f  不理会这个返回值
upgrade_start_vendor(){
    # $1 mode: upgrade_pre,boot-recovery,upgrade_post
    #return      0 -> start upgrade;    1 -> no upgrade
    #reutrn value only work in nornal mode
    #nornal mode: $NORMAL_MODE
    echo upgrade_start_vendor $1
    return 0
}
```

5.4. 写分区完成

```
upgrade_finish_vendor(){
    #set version or others
    reboot -f
}
```

5.5. -f (-n)调用顺序

1. check_network_vendor ->
2. upgrade_start_vendor ->
3. download_image_vendor (ramdisk_sys.tar.gz, -n 为 boot_initramfs.img)->
4. 内部烧写、清除镜像逻辑（不让已经使用镜像占用内存） ->
5. download_image_vendor(target_sys.tar.gz, -n 为 boot.img rootfs.img) ->
6. 内部烧写、清除镜像逻辑（不让已经使用镜像占用内存） ->
7. download_image_vendor(usr_sys.tar.gz, -n 为 usr.img) ->
8. 内部烧写、清除镜像逻辑（不让已经使用镜像占用内存） ->
9. upgrade_finish_vendor

5.6. -p 调用顺序

1. check_network_vendor ->
2. download_image_vendor (usr_sys.tar.gz) ->
3. upgrade_start_vendor ->
4. 检测返回值，烧写 ->
5. upgrade_finish_vendor

6. 相关系统状态读写

相关的信息存储在 misc 分区，OTA 升级不会清除这个分区（重新烧写镜像会擦除）

读：

```
read_misc [command] [status] [version]
```

其中：

command 表示升级的系统状态（shell 脚本处理使用）
status 自由使用，表示用户自定义状态
version 自由使用，表示用户自定义状态

写：

```
write_misc [ -c command ] [ -s status ] [ -v version ]
```

其中：

-c 不能随意修改，只能有 aw-upgrade shell 修改
-s -v 自定义使用

7. OTA 配置

一般来说，默认方案目录下会有一份 defconfig_ota 配置文件，该文件用于编译生成一个带 ramdisk 的 kernel，即 boot_initramfs.img，作为 OTA 期间烧写到 recovery 分区或 extend 分区的备份系统。

用户可以基于原有的 defconfig_ota 进行配置，也可以自行拷贝 defconfig 为新的 defconfig_ota，然后进行适当修改和配置。

执行 make ota_menuconfig 可进行 OTA 配置。

如果方案进行了较多的修改，建议删除原本的 defconfig_ota，然后拷贝 defconfig 为新的 defconfig_ota，再进行配置。

如上文所述，必须选上 misc-upgrade 包，以及 ramdisk 选项，保留 wifi 功能。其他选项可以关掉，以对生成的 boot_initramfs.img 进行裁剪。

8. 对 overlayfs 的处理

Tina 默认使用 overlayfs，则用户对原 rootfs 的修改会记录在 rootfs_dafa 分区中。而 OTA 更新的是 rootfs 分区，默认不会修改 rootfs 分区。则用户对 rootfs 文件的增加，删除，修改操作都会保留，假如原本的 rootfs 中有文件 A，用户将其修改为 A1，而 OTA 更新将该文件修改为 B，则最终看到的仍然是 A1。这是由 overlayfs 的特性决定的，上层目录的文件会屏蔽下层目录。

如果希望 OTA 之后，以 OTA 更新的文件为准，移除所有用户的修改。则可以在 OTA 之后，重新格式化 rootfs_data 分区。

9. 对 busybox-init 的处理

有些平台使用了 busybox-init，以 R6 为例子：

R6 方案将启动方式从 procd 修改为 busybox-init，不再使用 procd 和 overlayfs，由此带来一系列变化。

OTA 脚本通过 1 号进程的名字来判断启动方式。并根据结果。在后续脚本中做区别处理。

对于 busybox-init，在第一次启动之后，会将 etc 的文件拷贝到 rootfs_data 分区中，并在后续挂载该分区作为 etc 目录。OTA 的过程不会更新 rootfs_data 分区。为了支持更新 etc 目录，增加了一个系统状态，upgrade_etc，并在原本设置 upgrade_end 的地方，改为设置成 upgrade_etc。而 busybox-init 的启动脚本会判断此标志，如果启动是标志为 upgrade_etc，则进行 etc 分区文件的更新，更新后设置系统状态为 upgrade_end。

主系统指定了启动脚本 init=/pseudo_init. 对于 OTA 使用的 recovery 系统也需要指定启动脚本，若所使用的方案未配置，可自行修改 env，在 cmdline 中传递 rdinit=/pseudo_init 进行指定。若所使用的文件系统中，已经有 rdinit 作为 pesudo_init 的软链接，则可使用 rdinit=/rdinit

10.常见问题

10.1. OTA 时出现 SQUASHFS ERROR

此问题是由于 Tina3.0 及更早版本的 OTA，在更新 rootfs 分区时，只是将 busybox 等备份到 ram 中，rootfs 分区尚处于挂载状态，此时如果有进程访问 rootfs，则会出现错误。

解决方式

1. OTA 之前，关闭其他进程，避免有进程访问 rootfs。如果确有进程需要保留，将其依赖的资源提前备份到 ram 中（相关代码可参考 OTA 脚本中的 prepare_env 函数）

2. 参照 3.1 的做法，在更新完 recovery 分区后，主动 reboot，进入 recovery 系统，在 recovery 系统中更新 boot 和 rootfs 分区

3. 参考原生 openwrt 的做法，在内存中构建 ram 文件系统后，执行切换根文件系统的操作，再进行更新

10.2. 编译 OTA 包之后，正常编译出错

出现编译问题，可能是由于 defconfig 被替换了

请检查下 target 仓库中方案对应目录的 defconfig 和 defconfig_ota 的状况。当前的 OTA 包编译过程，实质上是备份 defconfig ，使用 defconfig_ota 替换 defconfig，执行 make，最终还原 defconfig.
如果此过程中断，则 defconfig 未被还原，会导致下次正常编译出问题。

解决方式

还原方案目录下的 defconfig 文件。

建议 make menuconfig 和 make ota_menuconfig 之后，及时在 target 仓库下将修改提交入 git 仓库中，避免修改丢失，方便在必要时进行还原。

10.3. 是否可更新 boot0/uboot

目前默认的方案中不支持升级 boot0/uboot.

有需求可 make menuconfig 选上 ota-burnboot 的软件包，并使用该包进行升级。

11.Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgment to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.