深圳市华曦达科技股份有限公司		文 档 编 号	版本号	密级
		文档编号	Vx.x	密级
文档名称	SDMC 基础平台操作说明		**日期**	2015 年-月-日

SDMC 基础平台操作说明文档

文档作者：　杨宇锋　　日期：

项目经理：　　日期：

审　核：　　日期：

批　准：　　日期：

深圳市华曦达科技股份有限公司

文档历史发放及记录

序号	变更（+/-）说明	作者	版本号	日期	批准
1	草稿	杨宇锋	V1.0	2015.11.10	

注意：最新版本放在最前面。

文档简要功能及适用范围

1. 文档的简要功能

文档简要介绍 sdmc 基础软件的操作说明

1) API 操作说明

2) Patch 管理

2. 文档的适用范围

本文档适用于 SDMC 研发部软件工程师和相关硬件工程师。

适用平台包括 amlogic m8/s905、his 等平台

目录

1 关于分区

目前最新的设计方案不再添加自定义分区，统一使用 sdk 原厂所使用的分区。烧录分区所使用的命令；这里面就归纳几种单独烧录测命令，供调试用

1.1 在 u-boot 命令行下烧录

#sdc_update bootloader u-boot.bin //单独烧录 uboot
#sdc_update boot boot.img //烧录内核
#sdc_update recovery recovery.img //烧录 recovery
#sdc_update logo logo.img //烧录 logo(开机 logo)
#sdc_update system system.img //烧录 system 分区

注意：以上命令只适用于 amlogic 平台,要烧录的文件放在 tf 卡或者 SD 卡根目录

1.2 在系统命令行烧录

#[busybox] dd if=/patch/boot.img of=/dev/block/boot
说明：[busybox]：可选 /patch/：boot.img 存放的路径 其他分区的烧录同理

注意：以上命令适用于 amlogic 平台，his 平台的分区名称不同，只需更改适用于 his 平台的分区名称即可

2 dm2016 检测

函数功能：检测 dm2016
输入参数：无
输出参数：无
返回值：0—成功 -1—失败
函数原型：int dm2016_check(void)

3 升级 API

3.1 获取平台信息

函数功能：获取平台信息
输入参数：无
输出参数：misc_msg *sys_info，详见：sdmc-libs /extra/include/extra_otapackage.h
返回值：0-成功，其他：errorcode
函数原型：int get_platform_sysinfo(misc_msg *sys_info);

3.2 获取版本信息

函数功能：获取升级包版本信息
输入参数：const char *fn -----升级包名
输出参数：char *version ----- 版本信息
返回值：0-成功，其他：errorcode
函数原型：int get_update_file_version(const char *fn, char *version);

3.3 校验包信息

函数功能：校验包信息是否正确
输入参数：const char *path—包名
输出参数：无
返回值：check_fw_result：errorcode

```
typedef enum
{
        CHKFW_SUCCESS = 0,
        CHKFW_FILE_INVALID,         /* invalid data */
        CHKFW_OLDER_THAN_CURRENT,    /* current package older than system */
CHKFW_NOT_MATCHTYPE,        /* customer type not match */
CHKFW_SAME_WITH_CURRENT,     /* current package's version is same to system */
CHKFW_INC_BASE_VER_NOT_MATCH, /* system version != incremental package base version */
CHKFW_INC_VERSION_NOT_NEWER,  /* system version >= incremental package fireware version */
CHKFW_INC_FS_NOT_MATCH,      /* system file system is not match as incrental package */
CHKFW_OTHERS_ERR,           /* others invalid */
}check_fw_result;
```

函数原型：check_fw_result sdmc_check_fireware(const char *path);

3.4 更新版本信息

函数功能：更新系统版本信息
输入参数：const char *path—包名
输出参数：无
返回值：0---成功 其他：errorcode
函数原型：int sdmc_update_fireware(const char *path);

3.5 设置 boot 模式

函数功能：设置 boot 模式
输入参数：boot_mode_type type-----指定的 boot 模式

```
typedef enum
{
        BOOT_MODE_NORMAL   = 0x00000000,
        BOOT_MODE_RECOVERY = 0x01020201,
        /* low level factory test mode */
        BOOT_MODE_FACTORY  = 0x03040403,
        /*  high level factory test mode */
        BOOT_MODE_FACTORY2 = 0x05060605,
        /* secure boot */
        BOOT_MODE_SECURE   = 0x07080807,
```

} boot_mode_type;
输出参数：无
返回值：0---成功 其他：errorcode
函数原型：int set_boot_mode(boot_mode_type type);

4 IDburn API(以下 API 均从 dm2016 读写)

4.1 读 activeCode

函数功能：读 activeCode
输入参数：无
输出参数：char* activeCode--- activeCode 值（16 bytes）
返回值：0—成功 其他：errorcode
函数原型：int readActivecode(char* activeCode);

4.2 写 activeCode

函数功能：写 activeCode
输入参数：char* activeCode--- activeCode 值（16 bytes）
输出参数： 无
返回值：0—成功 其他：errorcode
函数原型：int readActivecode(char* activeCode);

4.3 读 8 个字节 sn

函数功能：读一个字节 sn
输入参数：无
输出参数：char *sn – 8 个字节的 sn 值
返回值：0—成功 其他：errorcode
函数原型 int readSN(char* sn);

4.4 写 8 个字节 sn

函数功能：写一个字节 sn
输入参数：char *sn – 8 个字节的 sn 值
输出参数：无
返回值：0—成功 其他：errorcode
函数原型 int writeSN (char* sn);

4.5 读 cpu ID

函数功能：读 cpu ID
输入参数：char* chipId –cpu ID（16 bytes）
输出参数：无
返回值：0—成功 其他：errorcode

函数原型：int readChipID(char* chipId);

4.6 写 cpu ID

函数功能：读 cpu ID
输入参数： 无
输出参数：char* chipId –cpu ID（16 bytes）
返回值：0—成功 其他：errorcode
函数原型：int writeChipID (char* chipId);

4.7 读 mac

函数功能：读 6 个字节的 mac
输入参数：无
输出参数：char *data -- 6 个字节的 mac
返回值：0—成功 其他：errorcode
函数原型：int readMacAddr(char *data);
int readMac2Addr(char *data);

4.8 写 mac

函数功能：写 6 个字节的 mac
输入参数：char *data -- 6 个字节的 mac
输出参数：无
返回值：0—成功 其他：Errorcode
函数原型：int writeMacAddr (char *data);
int writeMac2Addr(const char *data);

4.9 读用户 id

函数功能：读用户 id
输入参数：无
输出参数：char *data--用户 id（24 bytes）
返回值：0—成功 其他：Errorcode
函数原型：int readUserDeviceID(char *data);

4.10 写用户 id

函数功能：写用户 id
输入参数：char *data--用户 id（24 bytes）
输出参数： 无
返回值：0—成功 其他：Errorcode
函数原型：int writeUserDeviceID(const char *data);

4.11 读用户私有 ID

函数功能：读用户私有 id

输入参数：无
输出参数：char *data--用户私有 id（8 bytes）
返回值：0—成功 其他：Errorcode
函数原型：int readUserPrivateID(char *data);

4.12 写用户私有 ID

函数功能：写用户 id
输入参数：char *data--用户私有 id（8 bytes）
输出参数： 无
返回值：0—成功 其他：Errorcode
函数原型：int writeUserPrivateID(const char *data);

4.13 读用户硬件版本信息

函数功能：用户硬件版本信息
输入参数：无
输出参数：char *data--用户硬件版本信息（8 bytes）
返回值：0—成功 其他：Errorcode
函数原型：int readUserHardwareVersionID(char *data);

4.14 写用户硬件版本信息

函数功能：读用户 id
输入参数：char *data--用户硬件版本信息（8 bytes）
输出参数：无
返回值：0—成功 其他：Errorcode
函数原型：int writeUserHardwareVersionID(const char *data);

5 gpio 控制接口

gpio 控制接口位于/sys/class/sdmc/目录下：
/sys/class/sdmc/ant_pwr：天线供电，可读写
/sys/class/sdmc/demo_res：demo 复位，可读写
/sys/class/sdmc/dtv_pwr：dvb 电源，可读写
/sys/class/sdmc/net_led：网络指示灯，可读写
/sys/class/sdmc/s2_antpwr：s2 供电，可读写
/sys/class/sdmc/sys_led：系统指示灯，可读写
/sys/class/sdmc/wifi_type：wifi 模组型号，只读
/sys/class/sdmc/ir：遥控器接收指示灯，可读写
例如：向/sys/class/sdmc/sys_led 节点写 1 写 0 就可以实现系统指示灯的亮灭
#echo 1 > /sys/class/sdmc/sys_led //亮

#echo 0 > /sys/class/sdmc/sys_led //灭

读/sys/class/sdmc/sys_led 节点可以查看系统指示灯状态，如 Figure 1：

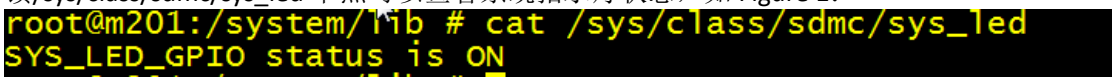

Figure 1

6 前面板

控制前面板的节点位于/sys/class/fd650s/目录下的 fd650s_ctrl，对这个节点的写操作就可以实现前面板的显示：，例如

#echo 1234 > /sys/class/fd650s/fd650s_ctrl

7 wifi &ble

打开 wifi 命令：# svc wifi enable

关闭 wifi 命令：# svc wifi disable

蓝牙关电：#echo 0 > /sys/class/rfkill/rfkill0/state

蓝牙上电：#echo 1 > /sys/class/rfkill/rfkill0/state

8 patch 管理

新增 patch 和打 patch 的前题是必须有一份与这个 sdk patch 对应的代码，本文档的前题本地有一份相对应该的代码，如果没有代码请参考 VSS 上 SDK 版本库操作说明_v1.x.doc 文档下载。

8.1 下载补丁

补丁文件下载按照以下步骤下载：[<user>是 10.10.121.100 服务器上的用户名]

mkdir ~/sdmc_patch

cd ~/ sdmc_patch

repo init -u 10.10.121.100:/home/svn/sdmc_lib/sdk_patch_tool/manifests.git -b master --repo-url=user@10.10.121.100:/usr/tools/repo.git

eg: repo init -u 10.10.121.100:/home/svn/sdmc_lib/sdk_patch_tool/manifests.git -b master --repo-url=yangyufeng@10.10.121.100:/usr/tools/repo.git

#repo sync

repo forall -c git checkout origin/master -b master

8.2 打补丁&去补丁

进入 sdmc 补丁根目录执行以下命令：（$0 是要执行的命令 $1 是 patch 路径 $2 是 SDK 的绝对路径 $4 是控制打补丁还是去补丁）

8.2.1 打补丁

./tool/patch.sh ./platform/his_kk_3796m_v60/patch_list.txt /home/yangyufeng/sdmc/hisilicon/his_v60/

8.2.2 去补丁

./tool/patch.sh ./platform/his_kk_3796m_v60/patch_list.txt /home/yangyufeng/sdmc/hisilicon/his_v60/ -r）

8.3 补丁制作

8.3.1 补丁命名

1) PATCH 文件夹的命名格式如下：

【xxxx】-【describe】

xxxx:补丁编号，占四个字节，在 patch 集合中递增
describe：patch 的功能描述
如：0062-add-AP6356-wifi-module
0062：是补丁编号，即第 0062 号 patch
add-AP6356-wifi-module：patch 的功能描述，即“add AP6356 wifi module”

2) PATCH 文件的命名格式如下：

【xxx】-【path】-【describe】

xxx：补丁编号，占三个字节，根据目录的不同来进行编号
path：合并目录，补丁打入的 sdk 路径(须是 git 的根目录)
describe：patch 的功能描述
如：003-hardware-libhardware_legacy-add-AP6356-wifi-module.patch
003：是补丁编号
hardware-libhardware_legacy：合并目录，是此补丁要打入的 sdk 路径是 hardware / libhardware_legacy，这个路径是 git 的第一级路径，也是 git 的根目录
add-AP6356-wifi-module：patch 的功能描述，即“add AP6356 wifi module”

8.3.2 patch list 说明

patch list 是需要打的补丁集合列表，格式如下：
补丁目录&合并目录|补丁&合并目录|补丁.....
补丁之间用“&”隔开，合并目录和补丁之间用“|”隔开。
合并目录和补丁表示一个单独的补丁。
如：
0042-add-demo-reset&uboot|001-uboot-add-demo-reset.patch
0042-add-demo-reset 表示存放补丁的目录，0042 是补丁的编号，后面是补丁提交的 commit 注释。用“-”进行分隔，commit 最后提交的注释是 add demo reset.
uboot|001-uboot-add-demo-reset.patch 有二个部分，分别是，uboot 合并补丁的 git 根目录，001-uboot-add-demo-reset.patch 合并的 PATCH。
注意：由于 shell 读行时是以回车符来确认的，在最后一个功能补丁后面记得回车换行，不然最后一个补丁合并不进去

8.3.3 生成补丁

在 sdk 对应文件进行修改或增加文件，修改完后进行测试，以验证问题或新增功能。可以用 git status 查看到修改的文件。

1) 用 git add 添加要修改的文件
2) git commit –m “注释” 注意注释一定要简单明了
3) git commit –m “注释” 注意注释一定要简单明了
4) 用 git format-patch 生成对应的补丁
5) 取补丁的名字新建目录
6) 拷贝补丁到新建的补丁的目录
7) 建立补丁修改的文件目录将修改好的文件拷贝到对应目录。

示例如下：

a. 修改文件，用 git status 可以查看到修改的文件，用 git diff 可以查看到具体修改内容。

Git status 结果，如 Figure 2：

```
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$ git status
# On branch sdmc_m8_20150414_patch_test
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   board/amlogic/m8_k200_v1/m8_k200_v1.c
#       modified:   board/amlogic/m8b_m201_v1/m8b_m201_v1.c
#
no changes added to commit (use "git add" and/or "git commit -a")
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
```

Figure 2

Git diff 结果，如 Figure 3：

```
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$ git diff
diff --git a/board/amlogic/m8_k200_v1/m8_k200_v1.c b/board/amlogic/m8_k200_v1/m8_k200_v1.c
index b4f2a73..98cf2a3 100755
--- a/board/amlogic/m8_k200_v1/m8_k200_v1.c
+++ b/board/amlogic/m8_k200_v1/m8_k200_v1.c
@@ -587,6 +587,9 @@ int board_init(void)

 #endif

+       gpio_amlogic_request(NULL, GPIOY_5);
+       gpio_amlogic_direction_output(NULL, GPIOY_5, 0);
+
     // LED
     clrbits_le32(P_AO_GPIO_O_EN_N, (1 << 15));
     clrbits_le32(P_AO_GPIO_O_EN_N, (1 << 31));
@@ -606,6 +609,9 @@ int board_init(void)
        wifi_power_init();
 #endif
        key_init();
+
+       gpio_amlogic_set(NULL,GPIOY_5, 1);
+
        return 0;
 }

diff --git a/board/amlogic/m8b_m201_v1/m8b_m201_v1.c b/board/amlogic/m8b_m201_v1/m8b_m201_v1.c
index bad90b8..d4e758a 100755
--- a/board/amlogic/m8b_m201_v1/m8b_m201_v1.c
+++ b/board/amlogic/m8b_m201_v1/m8b_m201_v1.c
@@ -510,6 +510,8 @@ void borad_power_init(void)
        gpio_amlogic_request(NULL, GPIODV_29);
        gpio_amlogic_direction_output(NULL, GPIODV_29, 0);

+       gpio_amlogic_request(NULL, GPIODV_24);
+       gpio_amlogic_direction_output(NULL, GPIODV_24, 1);
 }
 int board_init(void)
 {
@@ -552,7 +554,9 @@ int board_init(void)
     gpio_amlogic_direction_output(NULL, GPIOAO_13, 0);
 #endif

-
+       gpio_amlogic_set(NULL,GPIODV_24, 0);
+       msleep(15);
+       gpio_amlogic_set(NULL,GPIODV_24, 1);
        return 0;
 }

zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
```

Figure 3

b. git add 添加修改的文件，不报错表示成功，如 Figure 4。

```
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$ git add board/amlogic/m8_k200_v1/m8_k200_v1.c
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$ git add board/amlogic/m8b_m201_v1/m8b_m201_v1.c
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
```

Figure 4

c. git commit –m “注释” 进行本地提交，如 Figure 5

```
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$ git commit -m "add demo reset"
[sdmc_m8_20150414_patch_test cdc6177] add demo reset
 2 files changed, 11 insertions(+), 1 deletion(-)
zhangzicheng@ubuntu:~/jb-amlogic-m8-20150414/uboot$
```

Figure 5

d. 用 git format-patch 生成对应的补丁，如 Figure 6

Figure 6

e. 去掉生成补丁的.patch 并把当前 sdk 编号修改当前补丁的编号，在对应的 sdk patch 目录生成对应的补丁目录

a) 取.patch 前面用做文件名，如 Figure 7

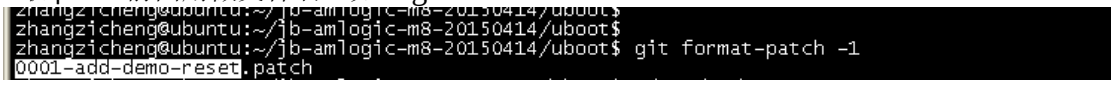

Figure 7

b) 查看当前 sdk 补丁的编号，如 Figure 8

Figure 8

c) 从 Figure 8 看 sdk 补丁最大编号是 0041,把以将新建补丁的编号改为 0042，即新补丁目录名为 0042-add-demo-reset

d) 新建补丁目录 mkdir 0042-add-demo-reset,并切换到 0042-add-demo-reset 目录，如 Figure 9

Figure 9

e) 将生成补丁拷贝到当前生成的 sdk patch 目录，如 Figure 10

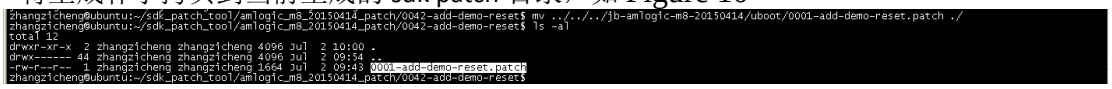

Figure 10

f) 在补丁目录新建原文件目录，以便补丁合并出错时用原文件对照查错。可以用 linux 命令行或者在 windows 下手动拷贝等方式用那种方式不做要求，如 Figure 11。注意：原文件的目录结构必须和在 sdk 中的保持一致。

Figure 11

f. 将生成的补丁按照 patch list 说明添加到 patch list 中即可

9 升级

9.1 Amlogic 平台

9.1.1 Zip 包升级

- 空片升级

如果板子是空片，就需要使用 TF 卡制作启动卡来升级，启动卡的制作方法如下:

1) 打开用于制作启动卡工具 BootcardMaker.exe，如 Figure 12

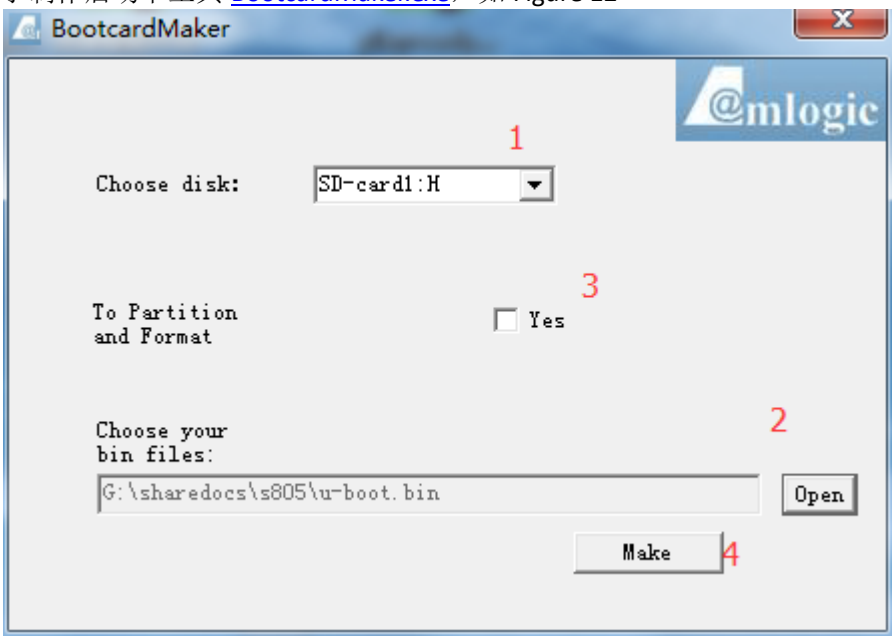

Figure 12

2) 选择 TF 卡，如 Figure 12 中红色数字 1 标注
3) 选择需要使用的 uboot 文件，如 Figure 12 中红色数字 2 标注
4) 如果 TF 卡已经损坏，请选中 Figure 12 中红色数字 3 标注的选项
5) 最后执行 make，即 Figure 12 中红色数字 4 标注的步骤，执行成功后会弹出 success 对话框，如 Figure 13

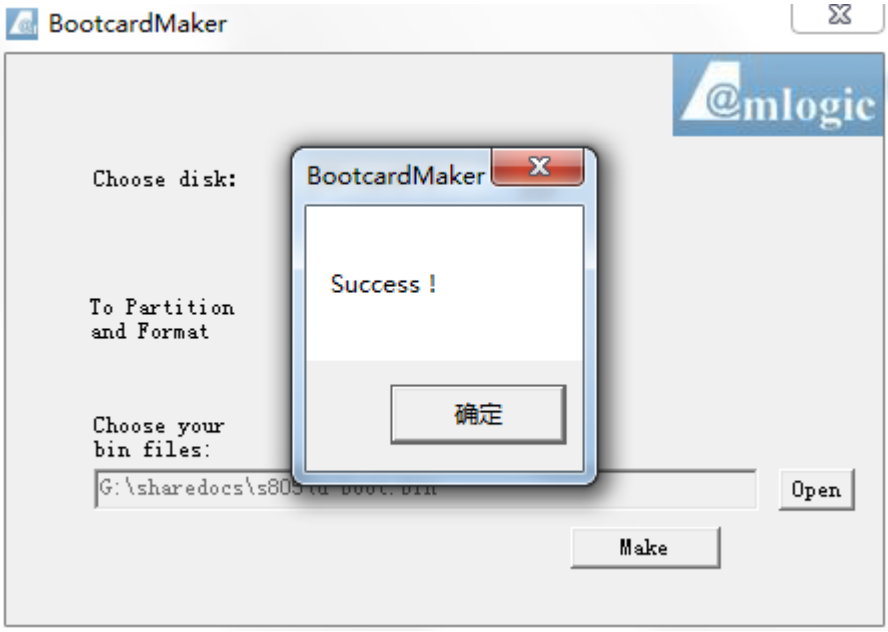

Figure 13

6) 然后把升级所需的 recovery.img 和 zip 包拷贝到 TF 卡，这时候启动卡就制作完成，接下来将制作好的启动卡，插入需要升级的盒子的 TF 卡槽，上电按 recovery 提示执行升级流程即可。

- 非空片升级

如果板子不是空片，直接在系统里升级 zip 包即可，如果升级失败或者升级后起不来，需要将板子擦空。使用以下命令将板子擦空：

$ store scrub 0

根据提示选择 yes 擦空 flash，然后按照空片升级的步骤进行升级。

9.1.2 img 包升级

img 升级主要是用来进行空片升级，工厂生产较多，同样使用 img 包来升级也需要擦空板子和制作启动卡，擦空板子参照 9.1.1Zip 包升级非空片升级。制作 img 包升级需要的启动卡如下：

1) 打开用于制作 img 升级包所需的启动卡工具 Burn_Card_Maker.exe，如 Figure 14

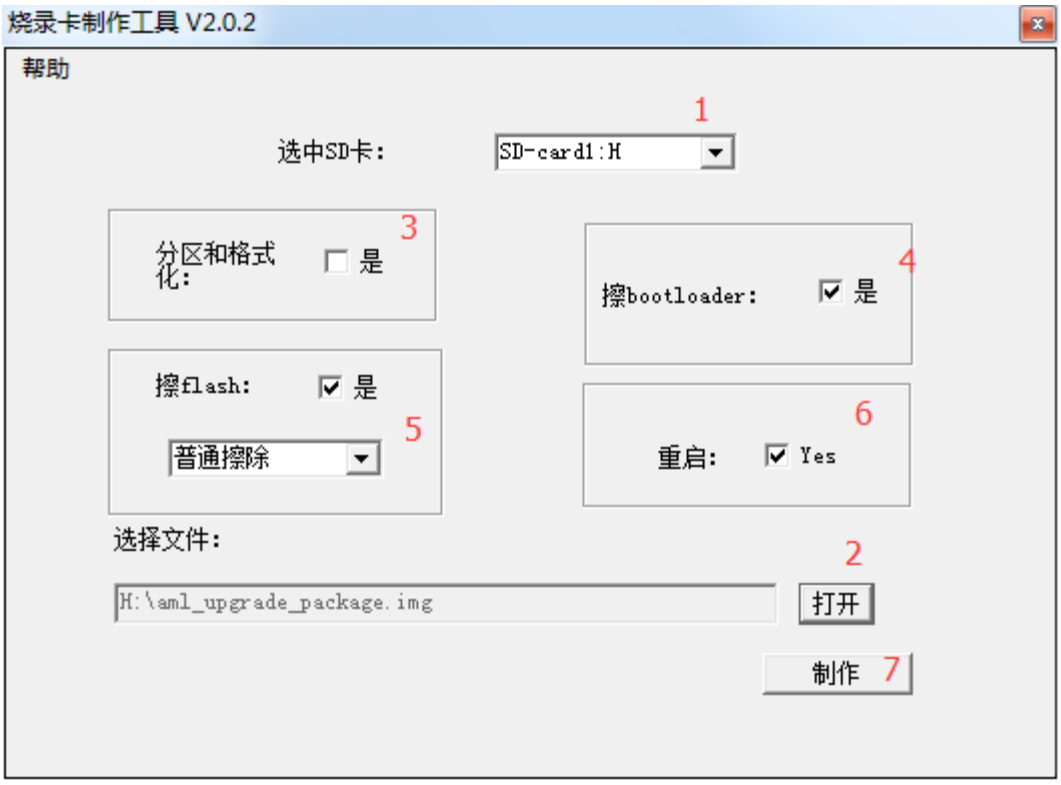

Figure 14

2) 选择配置，如 Figure 14 红色数字标注，1 是选择 TF 卡；2 是选择要升级的 img 升级包；3 是选择是否需要格式化 TF 卡，如果 TF 损坏就勾选，没有损坏就不勾选；4 是选择升级时是否要擦除 uboot 分区，默认即可；5 是选择是否需要擦空 flash，默认即可；6 是选择升级完成后是否自动重启，默认即可；7 是开始制作启动卡，点击后开始制作启动卡

3) 启动卡做好之后，将 TF 卡插入板子相应插槽，上电完成升级即可

9.2 His 平台

His 平台目前只支持 zip 包升级，暂不支持 img 包的升级方式。

- 空片升级

His 平台空片升级使用的 u 盘，步骤如下：

1) 准备 U 盘，格式化为 FAT32 文件系统，如果 U 盘本是 FAT32 则不用格式化。需要说明的是不支持 USB 读卡器、SD 卡转 USB、USB 硬盘、扩展分区 U 盘。在 U 盘中有多个分区情况下，请保证只有一个分区有 update.zip 升级包

2) 将编译好的 fastboot.bin，bootargs.bin，recovery.img，update.zip 四个镜像文件拷贝至 U 盘根目录

3) 将存有镜像文件的 U 盘插入单板 USB2.0 口，再给单板上电（如果是空板，可以不用按下按键）。约 2~5 秒后指示灯闪烁，进入 USB 升级流程，约几分钟后，指示灯常亮，升级完成

- 非空片升级

如果板子不是空片，直接在系统里升级 zip 包即可，如果升级失败或者升级后起不来，需要将板子擦空。使用以下命令将板子擦空：

```
$ mmc write 0x0 0x1000000 0x0 0x40000
```

然后按照空片升级的步骤进行升级。

10 SDK 下载

10.1 安装 repo

如果是第一次使用服务器，个人账户是没有安装 repo 的，首先就需要安装 repo 到个人账户，安装方法如下：

1. 在本地目录 repo 安装目录
   $ mkdir ~/bin
   $ cd ~/bin
2. 使用 scp 拷贝 100 服务器上的 repo，user 为自己在 100 上的账户名
   $ scp user@10.10.121.100:/bin/repo ./
3. 修改 repo 的 url, 把 REPO_URL 改为'user@10.10.121.100:/usr/tools/repo.git'，user 为自己在 100 服务器上的账户名，如红色部分
   $ vi repo

```
## repo default configuration
##
#REPO_URL='https://android.googlesource.com/tools/repo'
REPO_URL='user@10.10.121.100:/usr/tools/repo.git'
REPO_REV='stable'

# Copyright (C) 2008 Google Inc.
```

4. 导出 ~/bin 目录
   $ export PATH=~/bin:$PATH
   可把上面语句写到.bashrc 文件内，让用户登陆时自动执行
5. 生成 key 文件
   $ ssh-keygen
   直接敲回车键就可以生成 key
6. 把生成的~/.ssh/id_rsa.pub 拷贝到 100 服务器 user 帐号上,修改名字为 authorized_keys，并存到该帐户的~/.ssh 目录内.
   登陆 10.10.121.100 服务器，可用如下命令拷贝：
   $ scp -r user@10.10.121.4:/home/user/.ssh/id_rsa.pub ~/.ssh/

10.2 SDK 下载

10.2.1 S905 SDK 下载

SDMC SDK 版本管理使用 repo，工作服务器为 10.10.121.4，因此需要使用 repo 将 SDK 下载到本地目录，登录服务器个人账号后，可按照以下步骤同步 SDK：

1. $ mkdir ~/AndroidSDK
2. cd ~/AndroidSDK/
3. $ repo init -u ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_l_gx_git_mirror/l-amlogic-gx/platform/manifest.git -b l-amlogic-gx --repo-url=user@10.10.121.100:/usr/tools/repo.git （如 Figure 15）

(user:是该用户在 10.10.121.100 服务器上的账户名，如 yangufeng 的账户下载方式：

repo init -u ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_l_gx_git_mirror/l-amlogic-gx/platform/manifest.git -b l-amlogic-gx --repo-url=yangyufeng@10.10.121.100:/usr/tools/repo.git)

```
yangyufeng@ubuntu-sdmc:~/amlogic$ mkdir ~/AndroidSDK
yangyufeng@ubuntu-sdmc:~/amlogic$ cd ~/AndroidSDK/
yangyufeng@ubuntu-sdmc:~/AndroidSDK$ repo init -u ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_l_gx_git_mirror/l-amlogic-gx/platform/manifest.git -b
 l-amlogic-gx
Get yangyufeng@10.10.121.100:/usr/tools/repo.git
remote: Counting objects: 1288, done.
remote: Compressing objects: 100% (442/442), done.
```

Figure 15

4. $ repo sync

同步开始之前会提示输入账户名和邮箱地址，按提示输入即可,如 Figure 16。如果提示输入密码，如 Figure 20，请先执行 7~8，如果执行的时候还是报错，请执行跳过本步骤从 5 开始执行

```
Your Name  [yangyufeng@mail.sdmc.com]: yangyufeng
Your Email [yangyufeng]: yangyufeng@mail.sdmc.com

Your identity is: yangyufeng <yangyufeng@mail.sdmc.com>
is this correct [y/n]? y

repo initialized in /home/yangyufeng/AndroidSDK
yangyufeng@ubuntu-sdmc:~/AndroidSDK$
```

Figure 16

完成以上四步之后，只是将 repo 的版本信息更新下来，将 SDK 下载到本地好需要进行下述步骤：

5. $ cd .repo/manifests/
6. $ ls

如 Figure 17，这里面包含了各个版本的 SDK，下载指定的 sdk 方法如下：如果需要同步 0108 版本的 SDK，就把 openlinux_l-amlogic_20151031_patch_0108.xml 按照步骤 7 拷贝成默认版本，同样如果需要同步指定版本就把指定版本按照 7 拷贝成默认版本即可

```
yangyufeng@ubuntu-sdmc:~/AndroidSDK/.repo/manifests$ ls
amlogic.xml                    openlinux_l-amlogic_20150930.xml          openlinux_l-amlogic_20151031_SDK.xml  openlinux_l-amlogic_TV_20151231.xml
default.xml                    openlinux_l-amlogic_20151031_DVB.xml      openlinux_l-amlogic_20151031.xml      openlinux_l-amlogic_TV_20160122.xml
openlinux_l-amlogic_20150630.xml  openlinux_l-amlogic_20151031_patch_0108.xml  openlinux_l-amlogic_TV_20151130.xml  openlinux_l-amlogic_TV_20160304.xml
openlinux_l-amlogic_20150731.xml  openlinux_l-amlogic_20151031_patch_0401.xml  openlinux_l-amlogic_TV_20151207.xml
openlinux_l-amlogic_20150807.xml  openlinux_l-amlogic_20151031_patch_1204.xml  openlinux_l-amlogic_TV_20151225.xml
yangyufeng@ubuntu-sdmc:~/AndroidSDK/.repo/manifests$ $
```

Figure 17

7. $ cp openlinux_l-amlogic_20151031_patch_0108.xml default.xml

完成上述步骤后，需要修改 repo 的 URL 指向路径为

ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_l_gx_git_mirror/

8. vim default.xml （如 Figure 18）

```
diff --git a/default.xml b/default.xml
index 16cf73d..24c940c 100755
--- a/default.xml
+++ b/default.xml
@@ -2,7 +2,7 @@
 <manifest>

   <remote  name="openlinux"
-           fetch="ssh://git@openlinux.amlogic.com/" review="" />
+           fetch="ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_l_gx_git_mirror/" review="" />
   <default revision="l-amlogic-gx"
            remote="openlinux"
            dest-branch="l-amlogic-gx"
yangyufeng@ubuntu-sdmc:~/amlogic/s905_normal_0401/.repo/manifests$
```

Figure 18

9. $ repo sync
10. $ repo forall -c git checkout -b sdmc_s905_160108_patch（根据个人使用习惯选择是否要切分支）

完成上述步骤，SDK 的同步就完成了，注意下载下来的版本是 amlogic 的公版 SDK，如若修改，请不要提交到服务器。

10.2.2 S905X&S912&S905D SDK 下载

SDMC SDK 版本管理使用 repo，工作服务器为 10.10.121.4，因此需要使用 repo 将 SDK 下载到本地目录，登录服务器个人账号后，可按照以下步骤同步 SDK：

1. $ mkdir ~/AndroidSDK
2. cd ~/AndroidSDK/
3. $ repo init -u ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_m_gxl_git_mirror/m-amlogic/platform/manifest.git -b m-amlogic-openlinux-20160907 --repo-url=user@10.10.121.100:/usr/tools/repo.git （如 Figure 19）

(user:是该用户在 10.10.121.100 服务器上的账户名，如 yangufeng 的账户下载方式：repo init -u ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_m_gxl_git_mirror/m-amlogic/platform/manifest.git -b m-amlogic-openlinux-20160907 --repo-url=yangyufeng@10.10.121.100:/usr/tools/repo.git)

```
yangyufeng@ubuntu-121:~/AndroidSDK$ repo init -u ssh://svn@10.10.61.22/home/svn/amlogic_m_git_mirror/m-amlogic/platform/manifest.git -b m-amlogic-6.0.1 --repo-url=ssh://svn@10.10.61.22/home/svn/amlogic_m_git_mirror/repo.git
Get ssh://svn@10.10.61.22/home/svn/amlogic_m_git_mirror/repo.git
svn@10.10.61.22's password:
remote: Counting objects: 3091, done.
```

Figure 19

4. $ repo sync

同步开始之前会提示输入账户名和邮箱地址，按提示输入即可,如 Figure 16。如果提示输入密码，如 Figure 20，请先执行 7~8，如果执行的时候还是报错，请执行跳过本步骤从 5 开始执行

```
svn@ubuntu:~/amlogic_m_git_mirror$ repo sync
git@openlinux.amlogic.com's password:
```

Figure 20

完成以上三步之后，只是将 repo 的版本信息更新下来，将 SDK 下载到本地好需要进行下述步骤：

5. $ cd .repo/manifests/
6. $ ls

如 Figure 21，这里面包含了各个版本的 SDK，下载指定的 sdk 方法如下：如果我要下载 0907 版本 SDK，就把 openlinux_m-amlogic_20160907.xml 按照步骤 7 拷贝成默认版本，同样如果需要同步指定版本就把指定版本按照 7 拷贝成默认版本即可

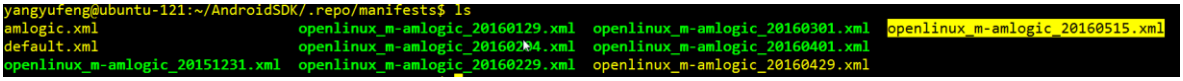

Figure 21

7. $ cp openlinux_m-amlogic_20160907.xml default.xml

完成上述步骤后，需要修改 repo 的 URL 指向路径为

ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_m_gxl_git_mirror/

8. vim default.xml （如 Figure 22）

```
diff --git a/default.xml b/default.xml
index 210673c..20463f1 100755
--- a/default.xml
+++ b/default.xml
@@ -2,7 +2,7 @@
 <manifest>

   <remote  name="openlinux"
-           fetch="../../" review="" />
+           fetch="ssh://amlogic_l_group@10.10.61.22/home/svn/amlogic_m_gxl_git_mirror/" review="" />
   <default revision="m-amlogic-openlinux-20160907"
            remote="openlinux"
            dest-branch="m-amlogic-20160907"
```

Figure 22

9. repo sync
10. $ repo forall -c git checkout -b sdmc_s905x_160907（根据个人使用习惯选择是否要切分支）

完成上述步骤，SDK 的同步就完成了，注意下载下来的版本是 amlogic 的公版 SDK，如若修改，请不要提交到服务器。

10.3 下载 sdmc patch

Patch 的下载和打 patch 按照 8 patch 管理进行，

10.4 下载 sdmc-libs

打完 patch 后，需要将 sdmc sdk 拷贝至 AndroidSDK 的 vendor 目录下：

1. $cd AndroidSDK/vendor/
2. $ ln -s sdk_patch/platform/vendor/amlogic/ sdmc

注意:以上 AndroidSDK 为用户下载的 sdk 目录路径，sdk_patch 为下载的 patch 路径

10.5 编译说明

10.5.1 S905 编译步骤

s905 平台按照以下步骤编译;

1. $ cd ~/AndroidSDK/
2. $ source build/envsetup.sh
3. $ lunch p201-user-32
4. $ make -j4

10.5.2 S905X 编译步骤

s905x 平台按照以下步骤编译;

1. $ cd ~/AndroidSDK/
2. $ source build/envsetup.sh
3. $ lunch p212-user-32
4. $ make -j4

10.5.3 S912 编译步骤

s912 平台按照以下步骤编译;

1. $ cd ~/AndroidSDK/
2. $ source build/envsetup.sh
3. $ lunch q201-user-32
4. $ make -j4

10.5.4 S905d 编译步骤

s905d 平台按照以下步骤编译;

5. $ cd ~/AndroidSDK/
6. $ source build/envsetup.sh
7. $ lunch p230-user-32
8. $ make -j4

11 打包材料发布

打包材料的发布随着打包工程的变更，需要发布的打包材料也有微小变动，下面列举 amlogic s905 平台需要发布的打包材料清单：

```
|-- Bootloader
|   |--u-boot.bin
|   |--Aml-Image
|   |   |--aml_sdc_burn.ini
|   |   |--aml_upgrade_package.conf
|   |   |--logo.img
|   |   |--manifest.xml
|   |   |--meson.dtb
|   |   |--platform.conf
|   |   |--u-boot.bin
|   |   |--u-boot.bin.sd.bin
|   |   |--u-boot.bin.usb.bl2
|   |   |--u-boot.bin.usb.tpl
|-- Dtb
```

```
|   |--gxbb_p201.dtb
|-- Kernel
|   |--kernel
|   |--ramdisk
|   |   |--root.tar.bz2
|   |   |--root-recovery.tar.bz2
|-- s9-system.tar.bz2
```

以下是各个打包材料的来源和生成方法：

Bootloader/u-boot.bin: out/target/product/p201/

Bootloader/Aml-Image: out/target/product/p201/upgrade/

Dtb/gxbb_p201.dtb: out/target/product/p201/obj/KERNEL_OBJ/arch/arm64/boot/dts/amlogic/

Kernel/kernel: out/target/product/p201/

Kernel/ramdisk/root.tar.bz2: 使用以下命令将 out/target/product/p201/root 目录压缩成 root.tar.bz2

$ tar -cvjf root.tar.bz2 root

Kernel/ramdisk/root-recovery.tar.bz2: 使用以下命令将 out/target/product/p201/recovery/root 目录压缩成 root-recovery.tar.bz2

$ tar -cvjf root-recovery.tar.bz2 root

s9-system.tar.bz2:使用以下命令将 out/target/product/p201/system 目录压缩成 s9-system.tar.bz2

$ tar -cvjf s9-system.tar.bz2 system

上述打包材料准备好并确认无误后一并发布到 svn 路径

svn://10.10.121.5/project/software/platform/s905

对应目录，写好 release note 并通知相关人员更新。