Application Notes

Amlogic Android 5.1 MBOX API

Revision 0.2

Amlogic, Inc.
2518 Mission College Blvd
Santa Clara, CA 95054
U.S.A.
www.amlogic.com

目录

修改记录

版本	日期	作者	修改
0.1	Aug 6th, 2015	Lei Qian	初稿
0.2	Oct 28th, 2015	Lei Qian	

1. 显示部分

1.1. 开机 logo 和动画

1.1.1. 实现原理

（1） uboot logo:

将一张 bmp 图片通过 fb1设备显示,直到 Android 系统启动的时候才消失。

替换路径： device/amlogic/xxx/logo_img_files/bootup.bmp

图片格式： 支持16位，24位色 bmp 图片。 16位色格式必须为 RGB 565，如果图片失真，很可能因为格式是 ARGB 555）； 24位色格式为 RGB 888. 通过修改“display_bpp=16”和“display_color_index=16”这两个 Uboot env 为“display_bpp=24”和“display_color_index=24”，可实现16位色到24位色的切换。

图片大小： 1080P 或1080P 以下

显示命令: uboot 下执行“run init_display”

（2） 开机动画：

bootanimation 动画，init.rc 中启动 bootanim 进程。bootanimation 的实现参考 frameworks/base/cmds/bootanimation/目录，这个进程会显示一段动画。

替换路径： device/amlogic/common/products/mbox

动画格式： 由 png 图片组成，以仅存储的方式打包

图片大小： 小于或等于 native UI，比如720P UI 只能用小于或等于720P 大小的图片

1.2. HDMI 和 CVBS 切换

1.2.1. 简介

Amlogic 公版默认的输出方式是单输出（HDMI 输出或 CVBS 输出），通过下面的输出规则确定 MBX 的输出模式：

① 检测到有 HDMI 接入时，自动关闭 CVBS 输出，打开 HDMI 输出；

② 检测到 HDMI 没有接入时，自动打开 CVBS 输出。

根据上面的输出规则，在拔插 HDMI 线时，MBX 会自动切换输出。

1.2.2. HDMI 插拔检测

在 vendor/amlogic/frameworks/services/systemcontrol/DisplayMode.cpp 的 HdmiPlugDetectThread()中，会开启一个轮询监听 HDMI 的 uevent 时间。在 init 进程启动不久，HDMI 插拔监测的机制就能生效。

5.1 上在开机过程中，在 uboot 和 kernel 也会检测一次 HDMI 插拔。另外以前版本惯用的 set_display_mode.sh

被挪到了 DisplayMode.cpp 的 setMboxDisplay 函数代码中。

1.2.3. 输出模式的设置 API

相关代码在 vendor/amlogic/frameworks/core/java/com/droidlogic/app/OutputModeManager.java，具体使用可参考 packages/apps/TvSettings/Settings/src/com/android/tv/settings/device/display/outputmode/OutputUiManager.java

(1) setOutputMode(final String mode)： 设置输出模式， 参数为 1080p60hz,1080p50hz,720p60hz,720p50hz,2160p24hz,2160p25hz,2160p30hz,2160p50hz420,2160p60hz420,smpte24hz,1080p24hz,576p50hz,480p60hz,1080i50hz,1080i60hz,576i50hz,480i60hz, 480cvbs,576cvbs

(2) setBestMode(String mode): 根据判断设置最佳输出模式。当 mode=null 时，设置为最佳模式；当 mode!=null 时，直接设置分辨率为 mode 所表示的值

(3) getBestMatchResolution(): 获取最佳模式

(4) isBestOutputmode()：判断自动适应最佳模式的开关是否打开

(5) getCurrentOutputMode(): 获取当前的输出模式

1.3 重显率的设置 API

相关代码在 vendor/amlogic/frameworks/core/java/com/droidlogic/app/DisplayPositionManager.java，具体 使用可参考 packages/apps/TvSettings/Settings/src/com/android/tv/settings/device/display/position/DisplayPositionActivity.java

(1) zoomIn(): 放大界面
(2) zoomOut()： 缩小界面

1.4 调试接口

如果遇到显示问题可 cat 如下节点，提供打印给相关开发人员

(1) sys/class/display/mode : 输出模式

(2) sys/class/amhdmitx/amhdmitx0/hpd_state: 如果为 1 则代表 HDMI 插上，为 0 则 HDMI 未插上

(3) sys/class/amhdmitx/amhdmitx0/disp_cap: 查看电视机的 edid, 显示 HDMI 支持的输出模式

(4) sys/class/graphics/fb0/free_scale: 0 代表 osd 缩放关闭，0x10001 代表 osd 缩放打开，默认值是 0x10001.

(5) sys/class/graphics/fb0/free_scale_axis: osd 源输出的大小，一般为 0 0 1279 719 或 0 0 1919 1079

(6) sys/class/graphics/fb0/window_axis: osd 目标输出的大小.如果 free_scale 打开，osd 则会从 free_scale_axis 所显示的大小缩放到 window_axis 显示的大小，这样才能保证界面正常.

(7) wm 命令：

wm size 1920x1080: 在线设置 android UI 的大小为 1920x1080，一般要设成跟 native UI 一样大才能保证显示正常

wm density 240： 在线设置系统的像素密度为 240

(8)dumpsys display:　打印 android 当前窗口的相关显示参数

2. 系统配置

以下都以 p201 为例

2.1. 切换 UI 大小:

将 device/amlogic/p201/files/mesondisplay.cfg 中的“1080p”改成“720p”或者“4k2k”。(必须保证在 kernel 的 dtd 文件中为 framebuffer 分配了足够的内存)。

2.2. 打开开机视频功能:

将 device/amlogic/p201/system.prop 的“service.bootvideo=0”改为“service.bootvideo=1”。

开机视频文件路径为 device/amlogic/common/products/mbox/mbox.mp4

2.3. 开机视频和动画的无缝切换调节

在 system.prop 中添加 “const.bootanim.delay=500”， “500”为 500ms, 意思是将开机 logo 消失的时间延后 500ms。原理是，等开机视频或者动画完全准备好的时候再播放，这样就可以让界面的过渡效果更好。

这个值可以根据您的实际需要添加或减少。 我们默认设置的是 100ms.

2.4. 修改默认壁纸:

更换 device/amlogic/common/products/mbox/default_wallpaper.png

2.5. 修改应用软件配置:

主要在 device/amlogic/common/products/mbox/product_mbox.mk 修改

2.6. 添加按 home 后需要保护的 APK

现在 5.1 上改成了按 home 键后，会 kill 当前的 APK，以保证此 APK 占用的资源全部释放。

如果某些 APK 不想在按 home 键后被 Kill,则可在 device/amlogic/p200/file/allowbackgroundapp.list 中按如下方式添加这些 APK 的包名，例如：

com.droidlogic.PPPoE

com.android.tv.settings

com.droidlogic.mediacenter

如果你想关闭这个功能，则只需添加如下字符:

all

2.7. 过滤不想显示的输出模式

在 device/amlogic/p201/overlay/packages/apps/TvSettings/Settings/res/values/config.xml 中修改

“<string name="display_filter_outputmode">480i60hz,576i50hz</string>”，这个字符串中添加或删除需要过滤的模式即可

2.8. Mbox Launcher 添加默认 APP 快捷方式

Mbox launcher 共有 5 个界面可更换默认 APP 快捷方式（首页快捷栏，在线视频，推荐，音乐，本地播放）。它们全都是通过 vendor/amlogic/apps/MboxLauncher2/res/raw/default_shortcut 这个文件配置的。

default_shortcut 内容解析

以下面的内容为例：

Home_Shortcut:com.farcore.videoplayer;com.android.music;app.android.applicationxc;

Video_Shortcut:com.youku.tv;com.qipo;com.togic.livevideo;com.moretv.tvapp;

Recommend_Shortcut:com.amlogic.mediacenter;com.example.airplay;com.amlogic.miracast;ol;

Music_shortcut:st.com.xiami;com.android.music;com.hycstv.android;

Local_Shortcut:com.android.gallery3d;com.farcore.videoplayer;com.fb.FileBrower;

“Home_Shortcut:”代表 首页快捷栏界面

“Video_Shortcut:” 代表 在线视频界面

“Recommend_Shortcut:”代表 推荐界面

“Music_shortcut:”代表 音乐界面

“Local_Shortcut:” 代表 本地播放界面

在这些标题的后加需要添加的 APK 的包名，就代表在对应的界面下显示这个 APK 的快捷方式。例如“Video_Shortcut:com.youku.tv;”的意思是， 将优酷显示到在线视频界面。

2.9. defenv 时保护部分 env 不被清除

我们的平台在 OTA 升级和恢复出厂设置的时候，默认会在 Uboot 执行“default env -a”来重置所有 ubootenv.

然而，ubootenv 保存了输出模式和重显率等显示方面的内容，如果想要在 OTA 升级和恢复出厂设置的时候，继续保留输出模式或者重显率，以 p201 为例：

1. 在 uboot/board/amlogic/gxb_p200_v1/gxb_p200_v1.c 的“const _env_args_reserve_[]”数组中添加 ubootenv 名字

2. 将 include/configs/gxb_p201_v1.h 中需要修改的“default env -a”改成“defenv_reserv”.

“defenv_reserv” 命令执行的时候，会将“const _env_args_reserve_[]”数组以外的所有 ubootenv 恢复到默认。

2. system control API

在 android 5.1 以及之后的版本，Amlogic 对系统进行操作的 API 都挪到了 systemcontrol 这个 service 中，我们需要通过它来控制显示，读写系统节点，修改系统 prop 和 ubootenv 等。这个 service 作为重要服务，会常驻于系统中。相关代码在 vendor/amlogic/frameworks/services/systemcontrol 和 vendor/amlogic/frameworks/core/java/com/droidlogic/app/SystemControlManager.java

3.1. c++调用

.c 文件可以通过.cpp 文件得到 API 生成方法，然后去调用.cpp 里面的函数。

3.1.1. Android.mk 中添加“LOCAL_SHARED_LIBRARIES += libsystemcontrolservice”，

3.1.2. .cpp 文件中添加“#include <systemcontrol/ISystemControlService.h>”

3.1.3. 获取 SystemControlService:

```
#include <binder/Binder.h>
#include <binder/IServiceManager.h>
#define SYST_SERVICES_NAME "system_control"

static sp<ISystemControlService> amSystemControlService;

const sp<ISystemControlService>& getSystemControlService() {
   if (amSystemControlService.get() == 0) {
      sp<IServiceManager> sm = defaultServiceManager();

      amSystemControlService = interface_cast<ISystemControlService>(sm-
>getService(String16(SYST_SERVICES_NAME)));

   }

   return amSystemControlService;
}
```

3.1.4. API 介绍：

(1) 获取 system prop 的值为字符型，不包含 uboot env
    bool getProperty(const String16& key, String16& value)
(2) 获取 system prop 的值，不包含 uboot env, 可设置返回的默认值
    bool getPropertyString(const String16& key, String16& value, String16& def)
(3) 获取 system prop 的值为整型，不包含 uboot env
    int32_t getPropertyInt(const String16& key, int32_t def)
(4) 获取 system prop 的值为长整型，不包含 uboot env

int64_t getPropertyLong(const String16& key, int64_t def)

(5) 获取 system prop 的值为布尔型，不包含 uboot env

bool getPropertyBoolean(const String16& key, bool def)

(6) 设置 system prop 的值

void setProperty(const String16& key, const String16& value)

(7) 读取系统节点的值

bool readSysfs(const String16& path, String16& value)

(8) 对系统节点写入指定值

bool writeSysfs(const String16& path, const String16& value)

(9) 设置 uboot env, 名字格式需为 ubootenv.var.xxxxx

void setBootEnv(const String16& key, const String16& value)

(10) 读取 uboot env 的值, 名字格式需为 ubootenv.var.xxxxx

bool getBootEnv(const String16& key, String16& value)

(11) 获取显示相关的信息

void getDroidDisplayInfo(int &type, String16& socType, String16& defaultUI,
int &fb0w, int &fb0h, int &fb0bits, int &fb0trip,
int &fb1w, int &fb1h, int &fb1bits, int &fb1trip)

(12) 设置显示输出模式

void setMboxOutputMode(const String16& mode)

(13) 设置当前制式的重显率

void setPosition(int left, int top, int width, int height) = 0

(14) 获取指定制式的重显率

void getPosition(const String16& mode, int &x, int &y, int &w, int &h)

3.2. java 调用

3.2.1. java 文件中添加“import com.droidlogic.app.SystemControlManager;”

3.2.2. 获取 SystemControlManager

```
SystemControlManager mSystemControlManager  = new SystemControlManager(mContext);
```

3.2.3. API 介绍：

(1) 获取 system prop 的值为字符型，不包含 uboot env

public String getProperty(String prop)

(2) 获取 system prop 的值为字符型，不包含 uboot env, 可设置返回的默认值

public String getPropertyString(String prop, String def)

(3) 获取 system prop 的值为整型，不包含 uboot env

public int getPropertyInt(String prop, int def)

(4) 获取 system prop 的值为长整型，不包含 uboot env

public long getPropertyLong(String prop, long def)

(5) 获取 system prop 的值为布尔型，不包含 uboot env

public boolean getPropertyBoolean(String prop, boolean def)

(6) 设置 system prop 的值

public void setProperty(String prop, String val)

(7) 读取系统节点的值

public String readSysFs(String path)

(8) 对系统节点写入指定值
public boolean writeSysFs(String path, String val)
(9) 读取 uboot env 的值, 名字格式需为 ubootenv.var.xxxxx
public String getBootenv(String prop, String def)
(10) 设置 uboot env, 名字格式需为 ubootenv.var.xxxxx
public void setBootenv(String prop, String val)
(11) 获取显示相关的信息
public DisplayInfo getDisplayInfo()
(12) 设置显示输出模式
public void setMboxOutputMode(String mode)
(13) 设置当前制式的重显率
public void setPosition(int x, int y, int w, int h)
(14) 获取指定制式的重显率
public int[] getPosition(String mode)