Android 解决方案

FAQ

文档版本　11

发布日期　2016-03-31

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为总部 邮编：518129

网址：http://www.hisilicon.com

客户服务邮箱：support@hisilicon.com

前 言

概述

本文档主要介绍 Android 解决方案一些功能的使用、调试方法，通过实例介绍各模块的如何使用，调试及注意事项。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3716C	V2XX
Hi3719C	V1XX
Hi3718C	V1XX
Hi3719M	V1XX
Hi3718M	V1XX
Hi3798C	V1XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX
HiSTBAndroid	V500R001
HiSTBAndroid	V600R001

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

作者信息

章节号	章节名称	作者信息
1.1	Android NDK 的使用	P00268318
2.1	WiFi	L66197
2.2	Ethernet	G00180855
3.1	Vold 挂载自定义修改	R00178058
3.2	eMMC 使用	R00178058
4.1	前面板	WKF29248/C00210106
4.2	fSD	WKF29248/C00210106
5.1.1	如何解决网页文字乱码问题	W00198192/H00182082
5.1.2	如何输入网址	W00198192/H00182082
5.1.3	如何修改首页导航网站的说明	W00198192/H00182082
5.1.4	如何让浏览器 HTML5 视频播放支持小窗口播放模式	Y00280361
5.2	多媒体	H00182082
6.1	音频	M00190072
6.2	视频	Z00180556(6.2.1)/ G00180855(6.2.2)
6.2.4	如何让视频层与图形层的显示区域保持一致	Z00272647
7.1	状态栏	W00185899
7.2	媒体扫描设置	T00194519
7.3	Android 原生动画效果	HWX201212
7.4	DLNA 和 Multiscreen	Y00227502
8.1	支持 512MB 内存设备	P00268318
8.2	1080P UI	Y00217998
8.3	海思竖屏显示特性	L00279110
8.4	支持开机优化	S00205252

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2016-03-31	11	新增 8.4 章节。
2015-01-26	10	新增 7.4 章节。
2014-12-19	09	新增 6.2.4 章节。
2014-11-18	08	删除 4.2 章节；修改 8.3 章节，增加竖屏制作方式。
2014-10-31	07	新增 6.2.3 章节；新增 4.2 章节。
2014-09-19	06	新增 5.1.4、8.3 章节。
2014-07-14	05	新增 3.1.3、5.1.2、 7.3、8.2 章节；删除原第 7 章的“后台进程”、“开发者选项”两节；修改了文中 371X 和 V5 等错误信息。
2014-03-30	04	新增第 6.2.2 章节。
2014-03-13	03	新增第 1、7.3、7.4、8 章节。
2014-01-16	02	新增 1.2、5、6 章节，修改使文档兼容 Android4.4 和 Hi3716MV400。
2013-11-20	01	新增 3、4 章节。
2013-07-31	00B01	第 1 次临时发布。

目 录

1 Android 开发环境类

1.1 Android NDK 的使用

1.1.1 版本选择

问题描述

如何选择和当前源码环境一致的 Android 版本？

解决办法

请参考表 1-1 的 Android 版本和 NDK 对应关系，截止到 Android4.4 版本。

表1-1 Android 版本和 NDK 对应关系表

NDK	Android	API	NDK EABI
n/a	1.0	1	arm-eabi-4.2.1
n/a	1.1	2	arm-eabi-4.2.1
1.5_r1	1.5	3	arm-eabi-4.2.1
1.6_r1	1.6	4	arm-eabi-4.2.1
r3	2.0	5	arm-eabi-4.4.0
	2.0.1	6	
	2.1	7	
r4	2.2	8	arm-eabi-4.4.0
r4b			
r5	2.3 2.3.3	9 10	arm-linux-androideabi-4.4.3
r5b			
r5c			

NDK	Android	API	NDK EABI
r6 r6b	3.0 3.1 3.2	11 12 13	arm-linux-androideabi-4.4.3
r7 r7b r7c	4.0 4.0.3	14 15	arm-linux-androideabi-4.4.3
r8	4.1 4.2	16 17	arm-linux-androideabi-4.4.3
r8b			arm-linux-androideabi-4.6
r8c			arm-linux-androideabi-4.6
r8d			arm-linux-androideabi-4.6
r8e			arm-linux-androideabi-4.6
r9	4.3	18	arm-linux-androideabi-4.6
r9b	4.4 4.4.x	19	arm-linux-androideabi-4.6
r9c			arm-linux-androideabi-4.6

根据上表对应关系，下载对应的 NDK 压缩包，链接地址：

```
http://dl.google.com/android/ndk/android-ndk-XX-linux-x86.tar.bz2
```

其中 XX 可以替换为 NDK 版本号。

例如：下载 r5 版本 NDK，可使用下述链接：

```
http://dl.google.com/android/ndk/android-ndk-r5-linux-x86.tar.bz2
```

1.1.2 通过 NDK 生成 EABI 工具链

问题描述

第三方开发者需要通过 C/C++代码生成动态链接库，但是不知道使用什么编译器才能编译出可以在当前 Android 版本上使用的动态链接库？

解决办法

通过 NDK 生成 EABI 工具链的操作步骤如下：

步骤 1 根据 1.1.1 章节信息，下载对应于当前 Android 版本的 NDK 包。

步骤 2 解压 NDK 包：

```
tar jxvf android-ndk-XX-linux-x86.tar.bz2
```

其中，XX 为 NDK 版本号。

步骤 3 生成可用于编译的 EABI 编译工具链工具。

```
build/tools/make-standalone-toolchain.sh --toolchain=AAA --
platform=android-BBB --install-dir=/tmp/my-android-toolchain
```

- AAA：EABI 编译工具链版本，请参考 1.1.1 章节的对照表获取正确值。
- BBB：android 版本 API 号，请参考 1.1.1 章节的对照表获取正确值。

----结束

例如：

生成可用于 Android 4.0 环境的 EABI 工具链：

下载 r7c 版本 NDK，网址：

```
http://dl.google.com/android/ndk/android-ndk-r7c-linux-x86.tar.bz2
```

解压 NDK：

```
tar jxvf android-ndk-r7c-linux-x86.tar.bz2
cd android-ndk-r7c
./build/tools/make-standalone-toolchain.sh --toolchain=arm-linux-
androideabi-4.4.3 --platform=android-14 --install-dir=/tmp/my-android-
toolchain
```

2 网络类

2.1 WiFi

2.1.1 如何配置 WiFi 编译？

问题描述

编译时是否要配置 WiFi 模块的编译选项，要如何配置？

问题分析

Android 平台支持多款 WiFi 模块，编译前需要先配置好采用的是哪种 WiFi 模块，否则 WiFi 驱动不会编译进镜像，WiFi 无法开启。

解决办法

编译前修改 BoardConfig.mk 文件，以 Hi379XXVXXX 芯片为例，如：device/hisilicon/Hi379XXVXXX/BoardConfig.mk 文件，根据采用的 WiFi 模块，打开 WiFi 驱动的编译开关。

如：

- 打开 RTL8188EUS 的编译开关：
  `BOARD_WLAN_DEVICE_RTL8188EUS = y`
- 打开 RTL8188ETV 的编译开关：
  `BOARD_WLAN_DEVICE_RTL8188ETV = y`
- 打开 AR9374 的编译开关：
  `BOARD_WLAN_DEVICE_AR9374 = y`

也可以同时打开几款 WiFi 模块的编译开关。

2.1.2 如何配置 WiFi 适用的国家或地域？

问题描述

在中国之外的市场，为何 WiFi 总会出现连接不上 AP 的情况？

问题分析

不同的国家或地域，WiFi 使用的信道不一样，比如中国使用信道 1~13，美国使用 1~11。WiFi 默认设置成中国，如果产品所发布的市场是中国之外的国家或区域，那么在编译前需要根据产品所发布的市场设置好 WiFi 的国家或区域，否则可能会出现无法连接上 AP 的问题。

解决办法

编译前要根据产品所发布的市场设置好 WiFi 的国家或区域。设置方法为修改 hardware/libhardware_legacy/wifi/wifi.c 文件，在 WiFi 的驱动参数中修改国家或区域参数。

如：

- RTL8188EUS/RTL8188ETV 设置成美国

```
#define DRIVER_MODULE_RTL8188EUS    2, \
{ \
    {"cfg80211","/system/lib/modules/cfg80211.ko","","cfg80211 "}, \
    {"rtl8188eu","/system/lib/modules/rtl8188eu.ko","ifname=wlan0
if2name=p2p0 rtw_channel_plan=0x22","rtl8188eu "} \
}
```

- AR9374 设置成美国

```
#define DRIVER_MODULE_AR9374    2, \
{ \

{"cfg80211_ath6k","/system/lib/modules/cfg80211_ath6k.ko","","cfg8021
1_ath6k "}, \
    {"ath6kl_usb","/system/lib/modules/ath6kl_usb.ko","
reg_domain=0x8348 ath6kl_p2p=0x13","ath6kl_usb "} \
}
```

国家或地域所对应的值可以询问 WiFi 厂家。

2.1.3 WiFi 待机唤醒后不能使用？

问题描述

WiFi 开启状态，待机唤醒后，WiFi 连接不上 AP，也扫描不到 AP，甚至单板无法唤醒。

问题分析

WiFi 待机有 cutpower 模式和 deepsleep 模式，cutpower 模式是在待机时将 WiFi 模块断电，而 deepsleep 模式是需要提供电源的。不同的 WiFi 支持的待机模式不一样，有些 WiFi 是默认采用 deepsleep 模式。Demo 板待机时，USB 接口是断电的，因此 deepsleep 待机模式的 WiFi 在唤醒后不能正常使用。

解决办法

为了统一方案，请在待机前将 WiFi 关闭，唤醒后将 WiFi 重新开启，版本中已采用这种方案。

2.2 Ethernet

2.2.1 WiFi 伪装如何关闭？

问题描述

系统启动后，使用有线网络或者 Pppoe 方式接入网络。在应用中，调用 Android 原生 ConnectivityManager 中的接口，查询当前可用网络的类型，发现不是 TPYE_ETHERNET 或者 TYPE_PPPOE，却是 TYPE_WIFI。

问题分析

此现象是由于代码中新增的 WiFi 伪装功能启用后导致。

解决办法

WiFi 伪装功能的加入，是为了提高应用兼容性，让那些只能在 WiFi 环境下运行的应用可以在有线和 Pppoe 环境下使用。

解决方案分为以下两种情况：

- 自研应用中需要获取到真实的网络状态

  此情况下，只需要在查询网络状态前，调用 EthernetManger 中的关闭 WiFi 伪装的接口，并在查询完成后，再调用 EthernetManger 中的启动 WiFi 伪装的接口即可。参考代码如下，请注意，参考代码仅列出代码逻辑部分，非完整代码：

  ```
  mEthernetManager.setWifiDisguise(false);
  NetworkInfo tempInfo = mConnectivityManager.getActiveNetworkInfo();
  mEthernetManager.setWifiDisguise(true);
  ```

- 不需要 Wifi 伪装功能

  此情况下，需要修改网络的 frameworks 层 EthernetService 中的相关代码，将 wifi 伪装设置和查询的相关接口的具体实现注释掉即可。参考代码如下：

  代码路径：

  ```
  frameworks/base/services/java/com/android/server/EthernetService.java
  ```

原代码：

```
    public void setWifiDisguise(boolean setEnable) {
        if(DEBUG) Log.i(TAG, "setWifiDisguise: " + setEnable);
        final ContentResolver cr = mContext.getContentResolver();
        Settings.Secure.putInt(cr, Settings.Secure.WIFI_DISGUISE,
                setEnable ? EthernetManager.WIFI_DISGUISE_ENABLED :
EthernetManager.WIFI_DISGUISE_DISABLED);
    }

    public int getWifiDisguiseState() {
        final ContentResolver cr = mContext.getContentResolver();
        try {
            if(DEBUG) Log.i(TAG, "getWifiDisguiseState: "
                    + Settings.Secure.getInt(cr,
Settings.Secure.WIFI_DISGUISE));
            return Settings.Secure.getInt(cr,
Settings.Secure.WIFI_DISGUISE);
        } catch (Settings.SettingNotFoundException e) {
            if(DEBUG) Log.i(TAG, "getWifiDisguiseState: STATE
UNKNOWN");
            return EthernetManager.WIFI_DISGUISE_STATE_UNKNOWN;
        }
    }
```

修改后的代码：

```
    public void setWifiDisguise(boolean setEnable) {
/*
//comment out these codes to disable WifiDisguise
        if(DEBUG) Log.i(TAG, "setWifiDisguise: " + setEnable);
        final ContentResolver cr = mContext.getContentResolver();
        Settings.Secure.putInt(cr, Settings.Secure.WIFI_DISGUISE,
                setEnable ? EthernetManager.WIFI_DISGUISE_ENABLED :
EthernetManager.WIFI_DISGUISE_DISABLED);
//comment out these codes to disable WifiDisguise
*/
    }

    public int getWifiDisguiseState() {
/*
//comment out these codes to disable WifiDisguise
        final ContentResolver cr = mContext.getContentResolver();
        try {
            if(DEBUG) Log.i(TAG, "getWifiDisguiseState: "
                    + Settings.Secure.getInt(cr,
```

```
Settings.Secure.WIFI_DISGUISE));
            return Settings.Secure.getInt(cr,
Settings.Secure.WIFI_DISGUISE);
        } catch (Settings.SettingNotFoundException e) {
            if(DEBUG) Log.i(TAG, "getWifiDisguiseState: STATE
UNKNOWN");
            return EthernetManager.WIFI_DISGUISE_STATE_UNKNOWN;
        }
//comment out these codes to disable WifiDisguise
*/
//always return EthernetManager.WIFI_DISGUISE_DISABLED;
        return EthernetManager.WIFI_DISGUISE_DISABLED;
    }
```

3 存储类

3.1 Vold 挂载自定义修改

3.1.1 如何更改 Vold 挂载？

问题描述

如果将其他外部存储介质（如 U 盘，SD 卡）作为默认存储，软件要如何修改？

问题分析

Android 默认将内置 Flash 作为默认存储，但有些硬件方案内置 Flash 容量配置不够大，希望将外接一个介质作为默认存储，那么就需要更改代码了。

解决办法

根据硬件确认要接的外部存储在哪个总线上，找到后修改 vold.fstab。如：需要把 /sys/devices/platform/hiusb-ehci.0/usb1/1-0:1.0 总线上的设备挂载成默认存储。步骤如下：

步骤 1 修改将[XXX].fstab 文件中挂载到/mnt/sdcard 的设备屏蔽掉。

步骤 2 增加默认存储挂载：

```
dev_mount block /mnt/sdcard auto /devices/platform/hiusb-ehci.0/usb1/1-0:1.0
```

----结束

3.1.2 UBI 虚拟 SD 卡分区变化，如何修改 vold 挂载？

问题描述

默认的内置 Flash 分区是 UBI 文件系统的第四分区。如果在前面增加了几个 UBI 分区。该怎么修改默认存储？

解决方法

确认增加的分区是在 SD Card 分区前面还是后面。

- 如果是在后面增加，不需要任何修改。
- 如果是在前面增加。则要看 SD Card 分区是第几个 UBI 分区。以 SD Card 分区是第 4 个分区为例。如果在前面增加一个，修改如下（注意：只修改“Ubifs=”后面的东西）：

  由：

  ```
  dev_mount  block  /mnt/sdcard  auto
  /devices/virtual/mtd/mtd13/mtdblock13  Ubifs=ubi3_0
  ```

  改为：

  ```
  dev_mount  block  /mnt/sdcard  auto
  /devices/virtual/mtd/mtd13/mtdblock13  Ubifs=ubi4_0
  ```

3.1.3 如何把某一个固定总线上来的外设虚拟成其他设备？

问题描述

有些单板没有 SATA 总线，但客户希望界面显示有，如何修改。

解决方法

将某一固定 USB 总线挂载上来的设备，显示成 SATA。比如：将 USB 总线“1-1:1.0”挂载上来的 USB 设备制定成 SATA 盘。

可以通过修改文件：system/vold 目录下的 DirectVolume.cpp 和 PartVolume.cpp 中函数来实现。

原来：

```
int PartVolume::addPath(const char *path) {
    mPaths->push_back(strdup(path));
     if ( strstr( path, "mmc") ) {
         mDevType = DEV_TYPE_SDCARD;
     } else if ( strstr( path, "ehci") || strstr( path, "ohci")) {
           mDevType = DEV_TYPE_USB2;
     } else if ( strstr( path, "usb3.0")) {
        mDevType = DEV_TYPE_USB3;
    } else if ( strstr( path, "ahci") ) {
        mDevType = DEV_TYPE_SATA;
    } else if ( strstr( path, "mtd") ) {
        mDevType = DEV_TYPE_FLASH;
    } else {
        mDevType = DEV_TYPE_UNKOWN;
    }
…
```

```
…
…
改为：int PartVolume::addPath(const char *path) {
    mPaths->push_back(strdup(path));
     if ( strstr( path, "mmc") ) {
         mDevType = DEV_TYPE_SDCARD;
     } else if ( strstr( path, "ehci") || strstr( path, "ohci")) {
        if (strstr( path, "1-1:1.0")) {
            mDevType = DEV_TYPE_SATA;
        } else {
            mDevType = DEV_TYPE_USB2;
        }
    } else if ( strstr( path, "usb3.0")) {
        mDevType = DEV_TYPE_USB3;
    } else if ( strstr( path, "ahci") ) {
        mDevType = DEV_TYPE_SATA;
    } else if ( strstr( path, "mtd") ) {
        mDevType = DEV_TYPE_FLASH;
    } else {
        mDevType = DEV_TYPE_UNKOWN;
}
…
```

3.2 eMMC 使用

3.2.1 如何编译 eMMC 系统镜像？

问题描述

如果板级存储介质是 eMMC 器件，如何编译出适合的系统镜像。

解决办法

以芯片 Hi379XXVXXX 单板为例，配置步骤如下：

步骤 1 在对应 BoardConfig.mk 脚本中配置 eMMC 所使用 system、userdata、cache、sdcard 分区的大小。

```
BOARD_SYSTEMIMAGE_PARTITION_SIZE := 524288000
BOARD_USERDATAIMAGE_PARTITION_SIZE := 1073741824
BOARD_CACHEIMAGE_PARTITION_SIZE := 104857600
BOARD_CACHEIMAGE_FILE_SYSTEM_TYPE := ext4
BOARD_SDCARDIMAGE_PARTITION_SIZE := 1572864000
```

说明

这个配置是必须的，原因是制作 ext4 镜像的工具（Android 原生自带）要求输入 ext4 分区大小。单位是 byte，例如 system 分区是 500MB，那么配置如下：

```
BOARD_SYSTEMIMAGE_PARTITION_SIZE：=500*1024*1024 = 524288000
```

步骤 2 配置 bootargs。

```
bootcmd=mmc read 0 0x1FFFFC0 0x4B000 0x5000; bootm 0x1FFFFC0
bootargs=mem=1G console=ttyAMA0,115200
blkdevparts=mmcblk0:1M(fastboot),1M(bootargs),10M(recovery),2M(deviceinfo
),8M(baseparam),8M(pqparam),20M(logo),20M(logobak),40M(fastplay),40M(fast
playbak),40M(kernel),20M(misc),500M(system),1024M(userdata),100M(cache),-
(sdcard)
```

- blkdevparts=mmcblk0 是关键字，必不可少。
- bootcmd=mmc read 0 0x1FFFFC0 0x 4B000 0x 5000; bootm 0x1FFFFC0

  这是 kernel 加载命令，其中：
  - 0x1FFFFC0 是 kernel 内存加载地址，不用更改；
  - 0x4B000 0x5000 分别乘以 0x200 才是 kernle 的镜像的起始地址和读取长度：例如：
    - 0x4B000 表示：0x4B000*0x200 =157286400 /1024/1024 MB = 150MB；
    - 0x5000 表示：0x5000*0x200 = 10485760 = 10MB。

  例如：kernel 位置是在 80MB，读取 10MB 长度（读取长度 10MB 足够，因为 kernel 大小基本在 10MB 之内），那么 kernel 开始地址：80*1024*1024/512 = 0x28000，长度：10*1024*1024/512 = 0x5000，最终 bootcmd 写为：

  ```
  bootcmd=mmc read 0 0x1FFFFC0 0x28000 0x5000; bootm 0x1FFFFC0
  ```

步骤 3 在 Project 根目录执行 make bigfish 或者 make ext4fs，在/out 的芯片目录的 eMMC 文件夹下生成 ext4 镜像。

----结束

4 外设类

4.1 前面板

4.1.1 前面板键值映射应该怎么修改？

问题描述

如果想把某一个按键改成其他功能，比如：将原 KEY_SETUP 按键改成 POWER 按键，应该怎么修改？

解决方法

修改文件 device/hisilicon/bigfish/system/frontpanel/key_pars/frontPanel.xml

```
<frontPanel_xml>
    <hisi-key>
        <key value="0x3"  name="KEY_UP"        />     <!--key up-->
        <key value="0x4"  name="KEY_DOWN"      />    <!--key down-->
        <key value="0x2"  name="KEY_LEFT"      />     <!--key left -->
        <key value="0x5"  name="KEY_RIGHT"     />     <!--key right-->
        <key value="0x1"  name="KEY_ENTER"     />     <!--key ok -->
        <key value="0x6"  name="KEY_BACK"      />     <!--Back -->
        <key value="0x0"  name="KEY_SETUP"     />     <!--Setup-->
        <key value="0x8"  name="KEY_HOME"      />     <!--Home-->
    </hisi-key>
</frontPanel_xml>
```

其中：

- 红色数值为底层 Linux 驱动获取的键值。
- 深蓝色键值为我们希望映射成的键值。

可以根据需求随意修改。如果想定义其他键值，键值可参考 device/hisilicon/bigfish/system/frontpanel/key_pars/linux_key.h 中 Linux_KeyCode_Ary 结构体。

比如上面将原 KEY_SETUP 按键改成 POWER 按键，修改后：

```
<frontPanel_xml>
    <hisi-key>
        <key value="0x3"  name="KEY_UP"        />     <!--key up-->
        <key value="0x4"  name="KEY_DOWN"      />    <!--key down-->
        <key value="0x2"  name="KEY_LEFT"      />     <!--key left -->
        <key value="0x5"  name="KEY_RIGHT"     />     <!--key right-->
        <key value="0x1"  name="KEY_ENTER"     />     <!--key ok -->
        <key value="0x6"  name="KEY_BACK"      />     <!--Back -->
        <key value="0x0"  name="KEY_POWER"     />     <!--Power-->
        <key value="0x8"  name="KEY_HOME"      />     <!--Home-->
    </hisi-key>
</frontPanel_xml>
```

4.1.2 如果前面板硬件修改了，相应软件应该怎么修改？

问题描述

如果客户使用其他厂家芯片的前面板，或者是直接弄几个 gpio 作为按键，那么相应的软件应该怎么修改？

解决方法

此类问题涉及到定制修改，就不一一根据场景描述了，用户可以在 device/hisilicon/bigfish/system/frontpanel/driver_interface/目录下添加自己的芯片驱动。重新实现 key_get（），key_init（）函数就好了。

4.1.3 前面板使用其他海思芯片，我们该怎么修改？

问题描述

现支持 5 款海思前面板芯片，分别是：

- 74HC164
- PT6961
- CT1642
- PT6964
- FD650

默认支持 CT1642，如果需要使用其他前面板芯片，需要怎么修改相应软件？

解决方法

在 device/hisilicon/bigfish/system/frontpanel/driver_interface/ keyled.h 中，有个宏定义，如下：

```
/*****************************************************
```

```
 *
 *   KEYLED_TYPE_DEF = 0:  74HC164
 *   KEYLED_TYPE_DEF = 1:  PT6961
 *   KEYLED_TYPE_DEF = 2:  CT1642
 *   KEYLED_TYPE_DEF = 3:  PT6964
 *   KEYLED_TYPE_DEF = 4:  FD650
 *
 ************************************************************/
#define  KEYLED_TYPE_DEF   2
```

当需要换成其他前面板芯片时，只需要重新定义 KEYLED_TYPE_DEF 的值就好了。

比如，需要使用 74HC164 芯片，那么可以将：

```
#define  KEYLED_TYPE_DEF   2
```

修改为：

```
#define  KEYLED_TYPE_DEF   0
```

5 浏览器类

5.1 界面显示

5.1.1 如何解决网页文字乱码问题？

问题描述

在浏览个别网页时，可能出现网页文字乱码，要如何解决？

问题分析

Android 平台支持多种网页编码方式，如 UTF8、GBK、Latin 等。如果网页源文件编码与网页指定的网页编码类型匹配且为正确类型，一般不会出问题。

当遇到一些网页写的不够规范时，可能会出现网页的文件编码类型与网页代码中指定的网页内容编码类型不匹配或不正确。这时浏览器按网页指定的编码或默认编码来显示文字就会出现乱码问题。

解决办法

浏览器设置里面有一个选项，用于手动设置网页的编码类型。当浏览器用于特定的语言地区时，可以手动将此编码类型更改为相应的文字编码类型，以提升征对特定语言文字的兼容性。

比如，当浏览器专门用于中文（简体）地区时，可以设置该选项为 GBK。

1. 对于开发者，可以在指定的配置文件中修改这个选项，方法如下：

步骤 1 进入浏览器源码目录

```
packages/apps/Browser
```

步骤 2 打开文件“res/values/strings.xml”。

步骤 3 找到指定的选项“pref_default_text_encoding_default”。

图5-1 找到指定的选项

```
<string name="pref_default_text_encoding_default" translatable="false">Latin-1</string>
```

步骤 4 修改对应的选项。

图5-2 修改对应的选项

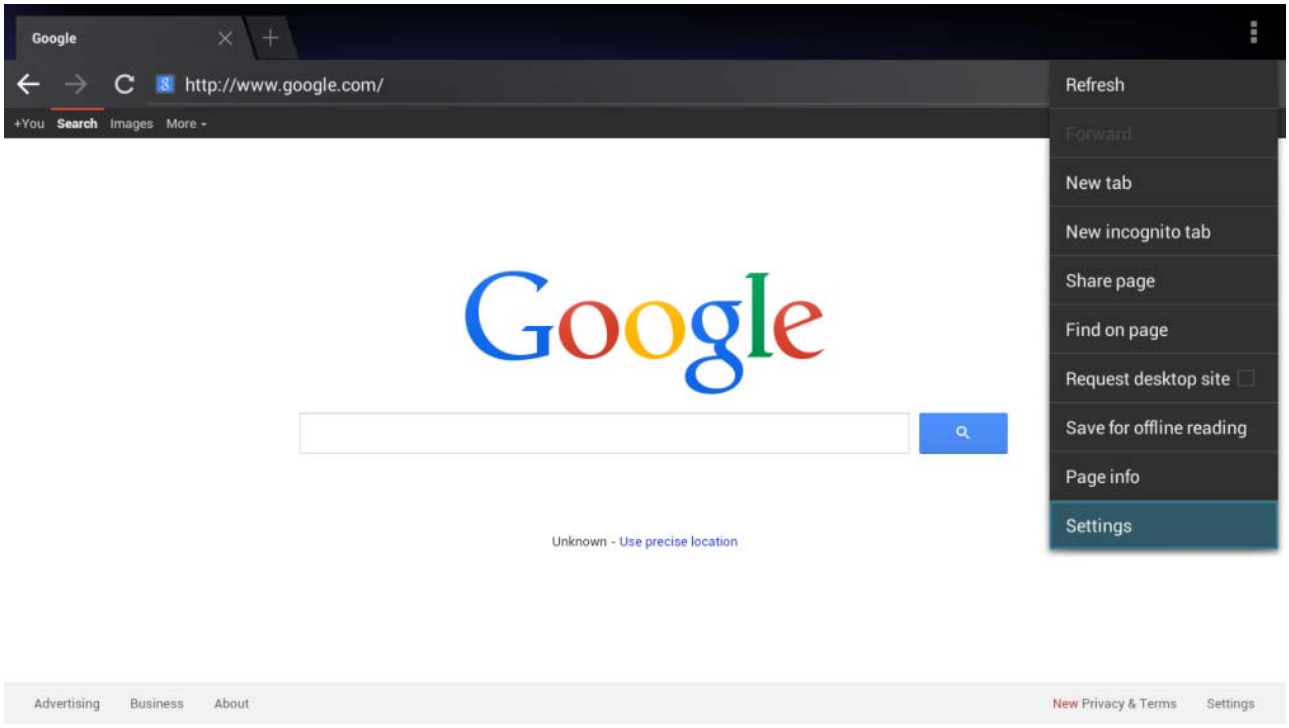
```
<string name="pref_default_text_encoding_default" translatable="false">Chinese (GBK)</string>
```

----结束

2. 对于使用者，可以在指定的浏览器设置界面修改这个选项，方法如下：

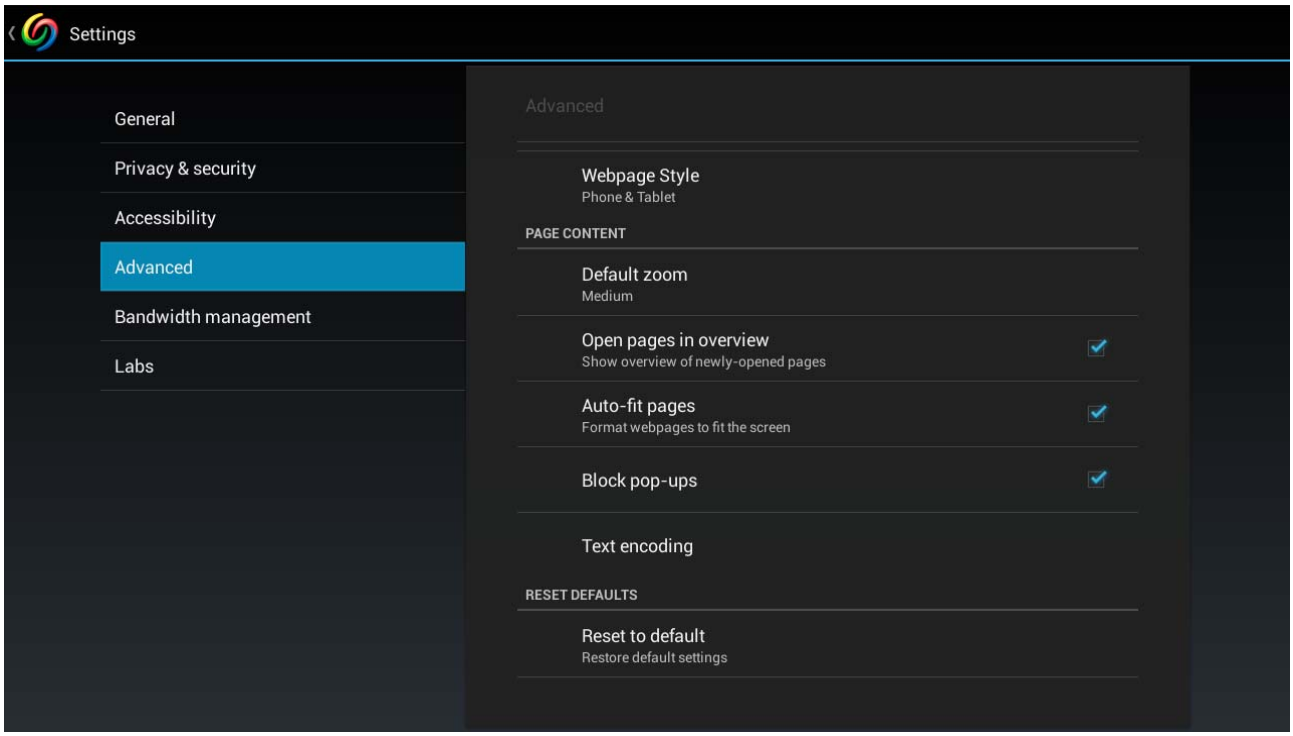
步骤 5 打开浏览器“Settings”。

图5-3 打开浏览器“Settings”



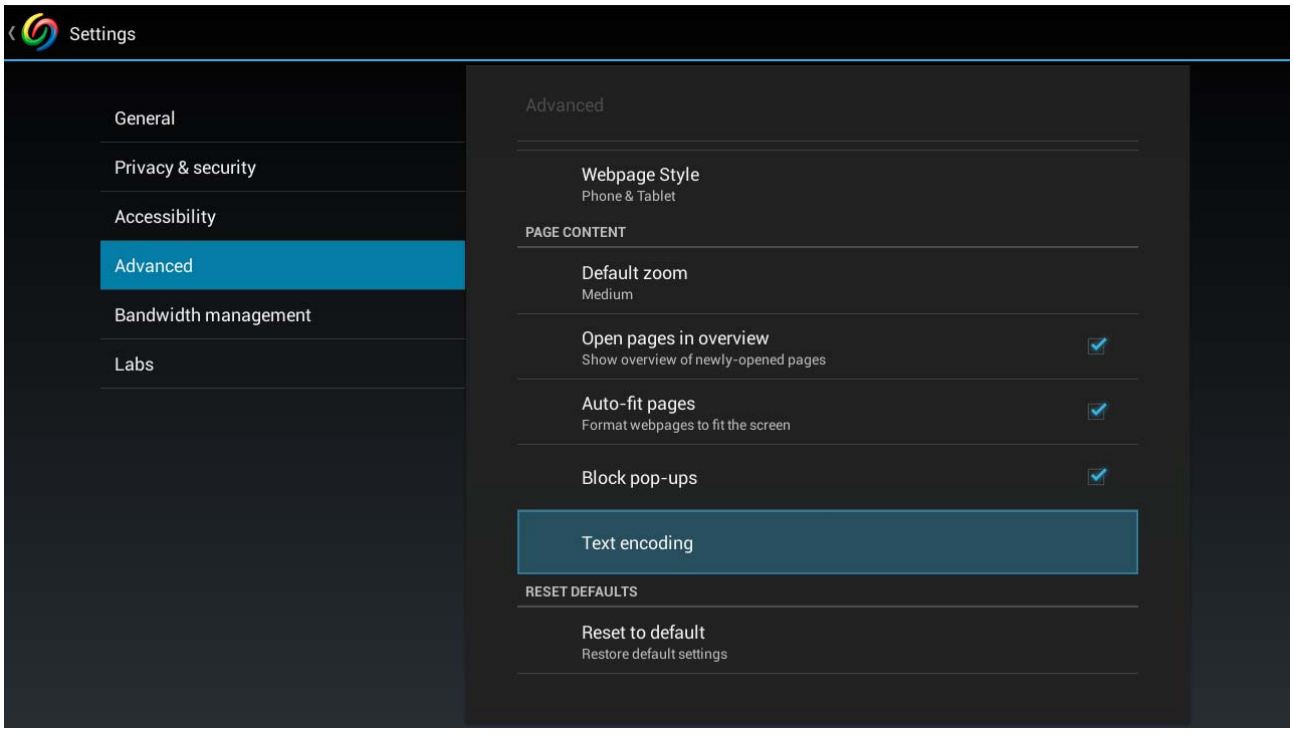
步骤 6 进入“Settings — Advanced”界面。

图5-4 进入“Settings — Advanced”界面



步骤 7 选择“Text coding”选项。

图5-5 选择“Text coding”选项



步骤 8　设置指定的编码格式。

图5-6 设置指定的编码格式

----结束

5.1.2 如何输入网址

说明
该问题只针对 Android 4.4 版本之后的版本。

问题描述

浏览器为全屏显示的网页，且默认状态下没有显示地址栏，要如何输入网址？

问题分析

为了在 TV 上使用遥控器获得更好的操作体验，我们对界面作了优化，即网页全屏显示，默认不会显示地址栏。用户在刚开始使用时，会感觉不适应。

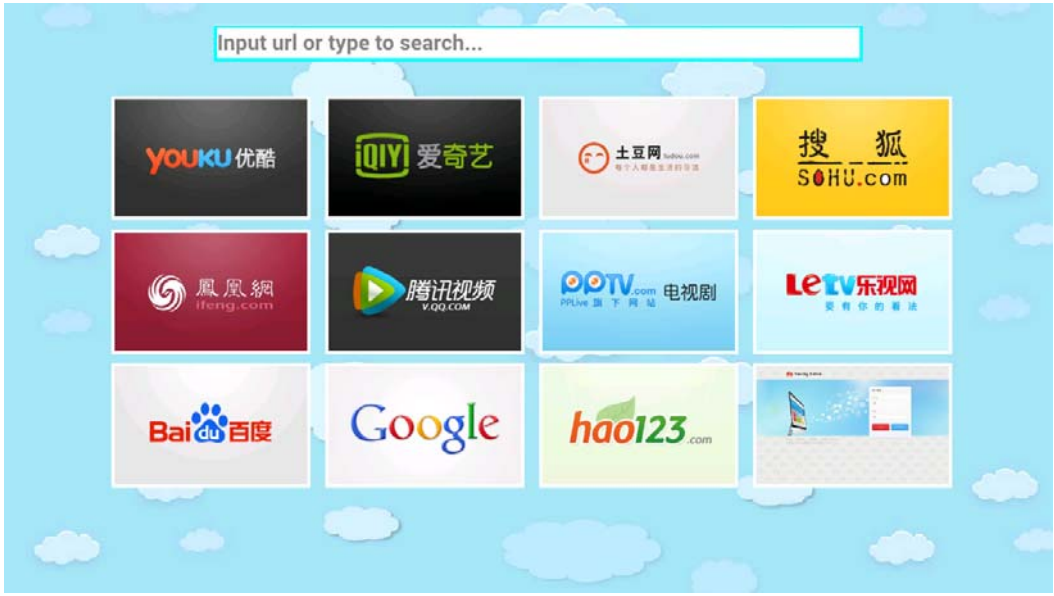
解决办法

浏览器提供一个简单易用的控制面板界面用于为用户提供常用的浏览器操作，包括输入 URL 地址、主页跳转、设置入口、添加书签、前进、后退、查看历史记录。有两种使用方法。

方法 1 如下：

步骤 1　默认进入浏览器时，显示为主页，如图 5-7 所示。

图5-7 浏览器首页



步骤 2　移动光标到如下区域，如图 5-8 所示。

图5-8 移动光标到输入框

步骤 3　按 OK 按键，调出控制面板，如图 5-9 所示。

图5-9 首页调出控制面板

步骤 4 移动光标到地址栏区域，按 OK 按键调出输入法，如图 5-10 所示。

图5-10 调出输入法

步骤 5 输入完成后，点击 GO，进入指定网页，如图 5-11 所示。

图5-11 输入完成

----结束

方法 2 如下：

当浏览任意界面时，如图 5-12 所示。

图5-12 浏览任意界面

步骤 6 按菜单键(也称 MENU 键)，调出控制面板，如图 5-13 所示。

图5-13 任意界面调出控制面板

步骤 7 移动光标到地址栏区域，按 OK 按键调出输入法，如图 5-14 所示。

图5-14 调出输入法

步骤 8 输入完成后，点击 GO，进入指定网页，如图 5-15 所示。

图5-15 输入完成

----结束

5.1.3 如何修改首页导航网站的说明

说明
该问题只针对 Android 4.4 及 Andorid4.4 版本之后的版本。

问题描述

浏览器首页有 12 个导航链接，要如何修改这些导航网站？

问题分析

目前只可以在代码中更改相应的资源文件，来替换这些导航网站。

解决办法

修改方法如下：

步骤 1 进入浏览器代码目录

`packages/apps/Browser`

步骤 2 找到需要更改的代码和资源（以更改“腾讯视频”链接为例）

`assets/HiBrow/LocalNavigation/navigation.html`

图5-16 修改前的代码

```
47
48          <div id="content_02">
49              <div class="link_01">
50                  <a href="http://v.ifeng.com/" tabindex="5"><img src="img/ifeng.png"></a>
51              </div>
52
53              <div class="link_02">
54                  <a href="http://v.qq.com/" tabindex="6"><img src="img/qq.png"></a>
55              </div>
```

`assets/HiBrow/LocalNavigation/img/qq.png`

步骤 3 更改相应的代码和资源（将“腾讯视频”改为 facebook）

`assets/HiBrow/LocalNavigation/navigation.html`

图5-17 修改后的代码

```
47
48          <div id="content_02">
49              <div class="link_01">
50                  <a href="http://v.ifeng.com/" tabindex="5"><img src="img/ifeng.png"></a>
51              </div>
52
53              <div class="link_02">
54                  <a href="https://www.facebook.com/" tabindex="6"><img src="img/facebook.png"></a>
55              </div>
```

增加文件：

`assets/HiBrow/LocalNavigation/img/facebook.png`

删除文件：

`assets/HiBrow/LocalNavigation/img/qq.png`

步骤 4 重新编译生成应用程序

----结束

5.1.4 如何让浏览器 HTML5 视频播放支持小窗口播放模式？

说明
该问题只针对 Android 4.4 版本。

问题描述

浏览器 HTML5 视频播放默认是进入全屏播放模式，如何让浏览器 HTML5 视频播放默认是小窗口模式？

问题分析

默认情况下，点击播放，浏览器 HTML5 视频播放进入全屏播放模式，原因是电视上视频播放大多需要全屏播放，屏蔽了小窗口播放模式，如果需要将 HTML5 视频播放默认设置为小窗口模式，需要通过修改配置属性并重新编译。

解决办法

修改方法如下：

步骤 1 进入目录。

```
/device/hisilicon/bigfish/build
```

步骤 2 打开属性配置文件 device.mk，修改属性值。

默认值为 true 表示默认全屏播放，修改为 false 表示默认小窗口播放。

```
hibrowser.default.fullscreen=false
```

步骤 3 重新编译生成镜像。

----结束

5.2 多媒体

5.2.1 如何打开网页上的 Flash 资源？

说明
该问题针对 Android 4.4 之前版本，Android 4.4 及 Andorid4.4 之后的版本不支持 Flash。

问题描述

有些网站打开后提示没有安装 Flash，有些内容无法显示。如何解决？

问题分析

默认情况下，浏览器优先获取网站的 HTML5 媒体资源。但有些网站没有 HTML5 资源，仍然发送了 Flash 资源，导致浏览器无法显示网页中的一些多媒体内容，如 Flash 广告，Flash 游戏，Flash 视频等。

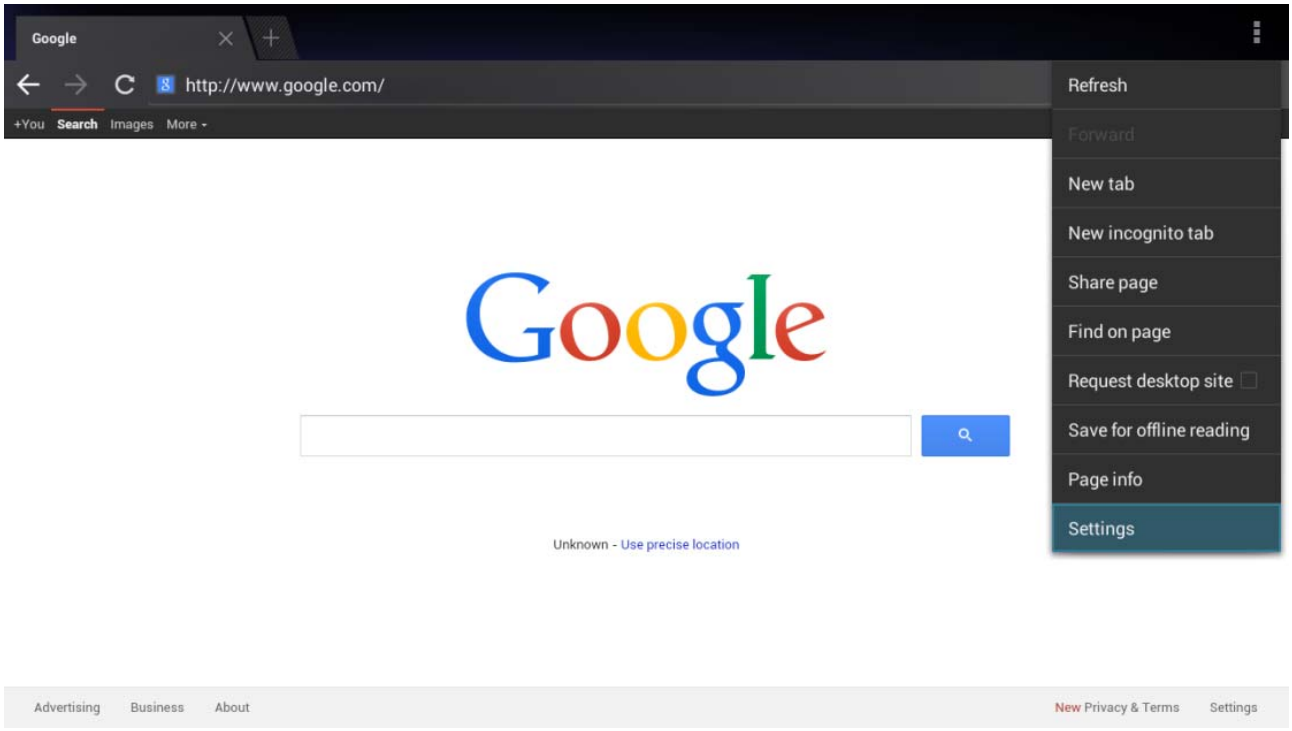
解决办法

浏览器设置里面有一项专门用于设置浏览器的模式，默认使用“Phone&Tablet”模式。如果要显示网页中的 Flash 视频、游戏等，就需要切换到“PC”模式。

操作方法如下：

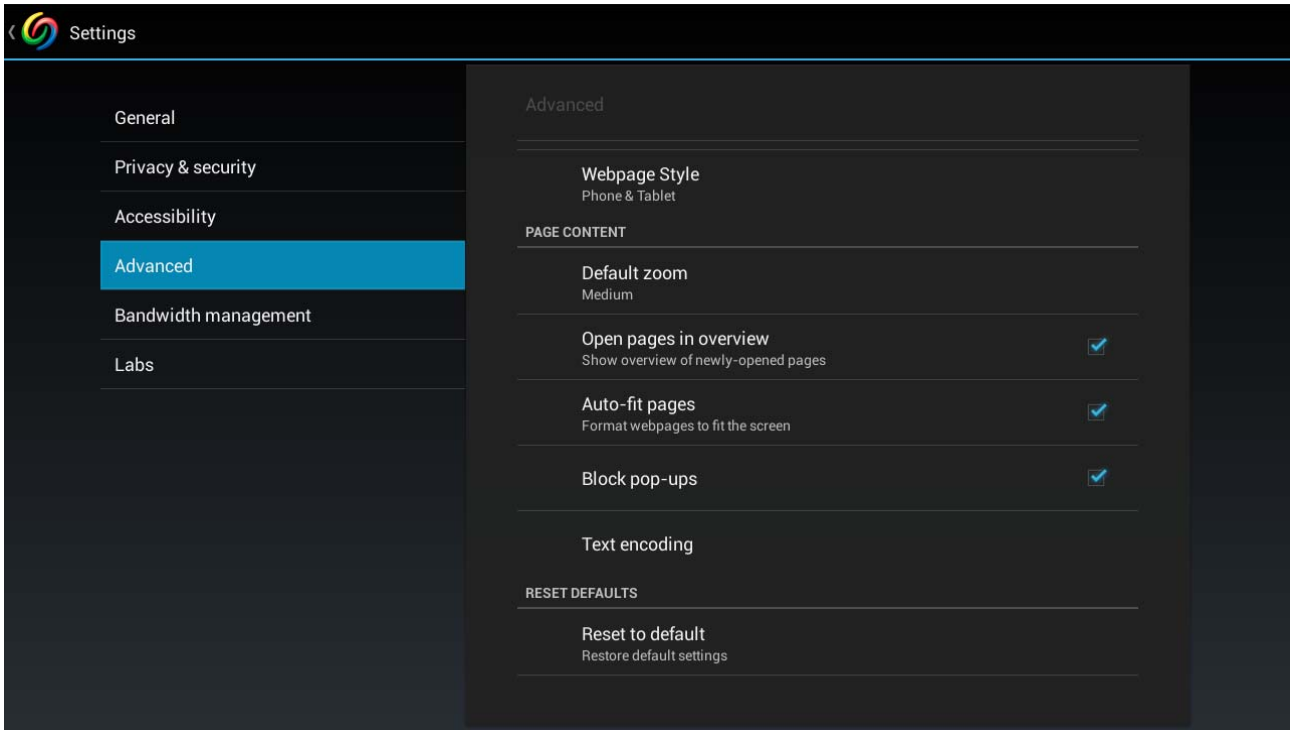
步骤 1 打开浏览器“Settings”。

图5-18 打开浏览器“Settings”



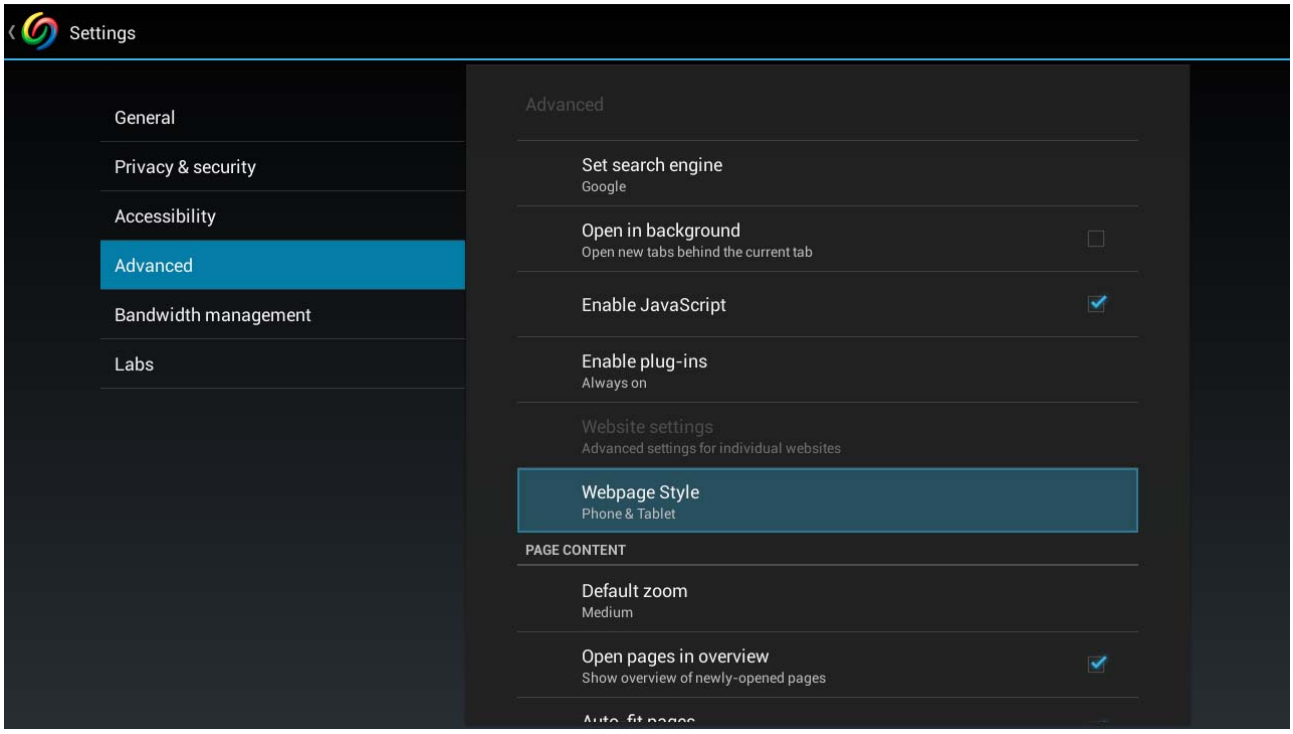
步骤 2 进入“Settings — Advanced”界面。

图5-19 进入“Settings — Advanced”界面



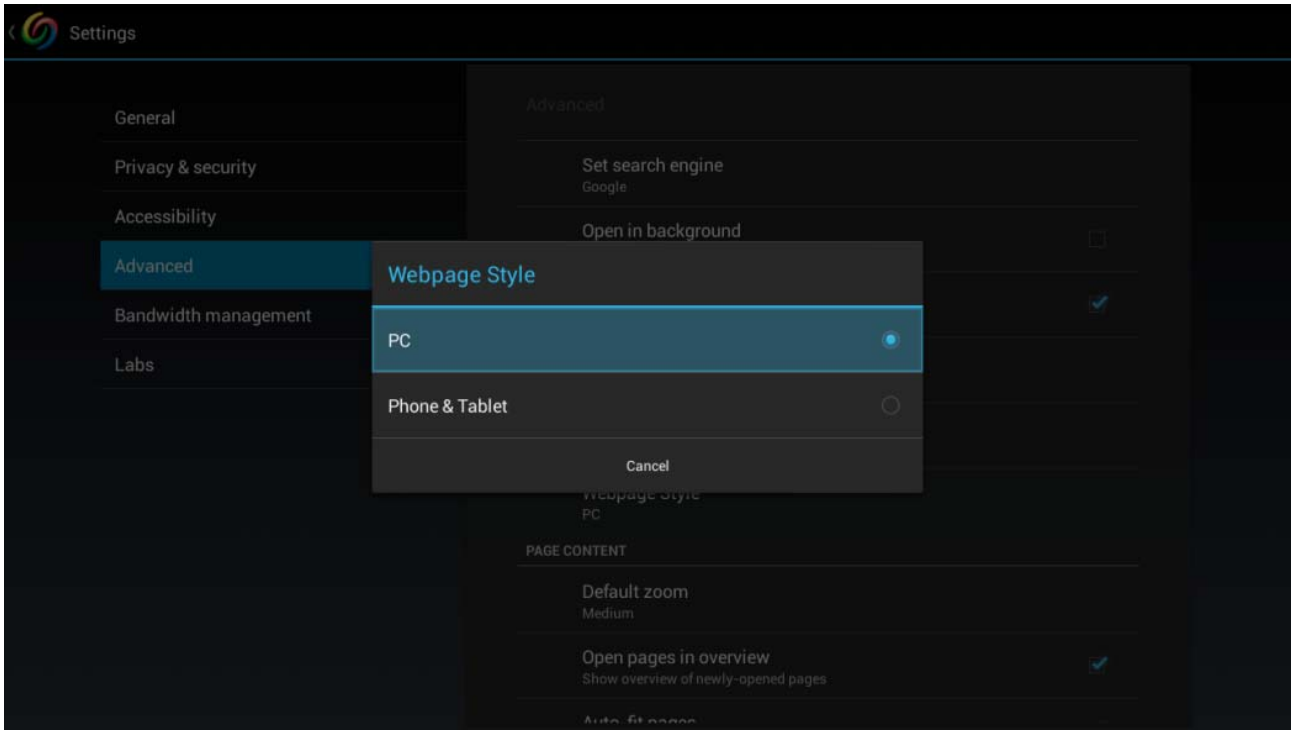
步骤 3　选择“Webpage Style”选项。

图5-20 选择“Webpage Style”选项



步骤 4　设置浏览器为“PC”模式。

图5-21 设置浏览器为“PC”模式



步骤 5 再次打开页面，或刷新网页。

----结束

6 媒体处理类

6.1 音频

6.1.1 如何开发音量控制功能？

问题描述

Android 下的音量调节实现接口与 SDK UNF 接口之间是共存的，在开发 DVB、IPTV、媒体播放等一些应用时不清楚应该使用哪些接口？

问题分析

造成上述问题的原因是因为 Android 原生代码提供了接口，同时 UNF 通路也可以调整，调用不当则会产生问题。

解决办法

我们在 Alsa HAL 层调用了 HIAO 驱动调节音量的接口，在调用 Android 原生接口同时可以调节到 HIAO 的音量，从而解决问题。该方案在典型场景下的使用说明如下：

调音入口只限定 Setting 页面和遥控器，这两者调的都是全局音量；apk 直接调 UNF 接口不能调全局音量，只能调整单路 track 音量。

- DVB、IPTV、HiPlayer 等应用调音通过全局调音生效，应用不需要响应遥控器按键事件，android 框架会自动处理。
- DVB 需要支持单频道音量，主要是当前频道音量实时保存。采用的方案：DVB 应监听 VOLUME_CHANGED_ACTION 事件，遥控器调音后会广播该事件，DVB 收到后保存当前音量及当前频道。DVB 切台到新频道时首先读取之前的保存值，然后调用 AudioManager 调音接口设置该频道音量初始值。
- 音量值的保存。
  - 全局音量：在 AudioService 调节完音量后会调用 persistvolume 把音量值写入 setting provider。在 AudioService 启动时，会从 setting provider 读取保存的值来初始化 mStreamStates 变量。
  - 单路音量：应用程序自己保存。

6.1.2 HDMI 透传输出策略

问题描述

HDMI 怎样才能按照透传模式输出？

问题描述

Setting 设置选项比较多，不清楚具体怎么用透传模式输出。

解决办法

为了可以得到更好的声音效果，我们提供了 HDMI、SPDIF 的源码输出功能。

在 Setting 应用 Sound UI 布局中提供了控制 HDMI 输出的关闭/解码/透传/次时代音频等几个开关，通过几个开关的使用可以满足各种场景的需求。

Sound 选项说明：

- HDMI 关闭：关闭的时候 HDMI 不再输出。
- HDMI 自动：音频 AO 会获取 EDID，根据 EDID 信息自动匹配输出模式。
  如果输出终端支持透传则透传输出，否则解码输出。
- HDMI 解码：对于没有外接解码设备的情况下可以选择这种模式，系统会将所有的音频解码成 PCM 后输出
- HDMI 透传：选择此项后，系统会始终将得到的 Dolby Dts Truehd 音频数据以源码的形式输出，但是对于一些普通格式的 mp3/aac 等则仍然解码输出。
- 蓝光次世代降规格，默认处于 Auto 状态，可以理解为 disable，对 HDMI 与 SPDIF 的输出不产生效果。选中 RAW5.1/7.1 选项时，若 HDMI 没有关闭则会强制使 HDMI 按照此规格输出。若播放码流不支持次世代音频设置规格，则输出当前支持的最高规格；对于 truehd 复核流的场景，如果强制指定 7.1 输出时则会选择 truehd 解码库处理，如果是强制 5.1 或者解码输出则会选择 Dolby 解码库。

6.2 视频

6.2.1 应用程序通过 UNF 接口进行视频播放的视频透出

问题描述

客户通过自己编写的应用程序，通过 JNI（Java Native Interface）方式调用 UNF 接口或者通过 MediaPlayer 标准接口方式实现多媒体播放时，出现以下情况：

- 有音频无视频现象；
- 在屏幕边缘看到有一条视频在播放，大部分视频画面被遮挡。

问题分析

通过 UNF 接口播放多媒体文件时，视频显示在视频层。默认的，视频层位于图形层的后方，需要将应用程序对应在图形层上的区域设置为透明，将视频层的视频透出来，用户才可以看到所播放的视频内容。

出现上述问题现象，可以通过隐藏图形层来确认视频是否是由于图形层的遮挡而不可见：

隐藏图形层，将视频层暴露在最上面，在串口输入以下命令：

```
echo hide > /proc/msp/hifb0
```

恢复图形层的显示，在串口输入以下命令：

```
echo show > /proc/msp/hifb0
```

如果隐藏图形层后，视频显示正常，则说明上述问题是由于视频层被图形曾遮挡引起。需要将视频所对应的区域，在图形层上设置为透明，来实现视频透出。

解决办法

应用程序通过创建一个 SurfaceView，用以在图形层上对应所播放的视频位置，并将该 SurfaceView 设置为透明，实现视频的透出。在海思平台上，需要将所对应的 SurfaceView 设置透明标志来实现所对应区域的透明处理。具体方法为：

```
msurfaceholder.setType(SurfaceHolder.SURFACE_TYPE_HISI_TRANSPARENT);
```

```
//msurfaceholder 为 SurfaceView 所对应的 SurfaceHolder
```

如果使用上述方法在某些异常情况下还是显示不出视频，请使用以下方式在有问题场景时抓取日志，由研发进行更深入分析。

- 抓取 android 屏幕截图，使用 PC 上的 ddms 进行抓图，或者在串口执行：
  `screencap -p /data/screen.png`
- 抓取 SurfaceFlinger 的状态日志，在串口执行：
  `dumpsys SurfaceFlinger`

将以上两项抓到的截图和 SurfaceFlinger 状态日志发送给海思 FAE 和研发分析。

6.2.2 多参考帧视频支持

问题描述

发布包默认配置编译出的版本多参考帧视频无法播放。

问题分析

当前使用的 SDK 版本支持的参考帧数量小于问题视频的参考帧数量。

解决办法

可按如下步骤进行解决：

步骤 1 首先确认当前版本使用的 SDK 配置文件，即 XXX_cfg.mak 文件。

以 Hi3798MV100 芯片为例，进入代码中对应的产品目录， Hi3798MV100 芯片的产品目录为 device/hisilicon/ Hi3798MV100。

说明
其他芯片的产品目录都可以在 device/hisilicon/目录下找到。

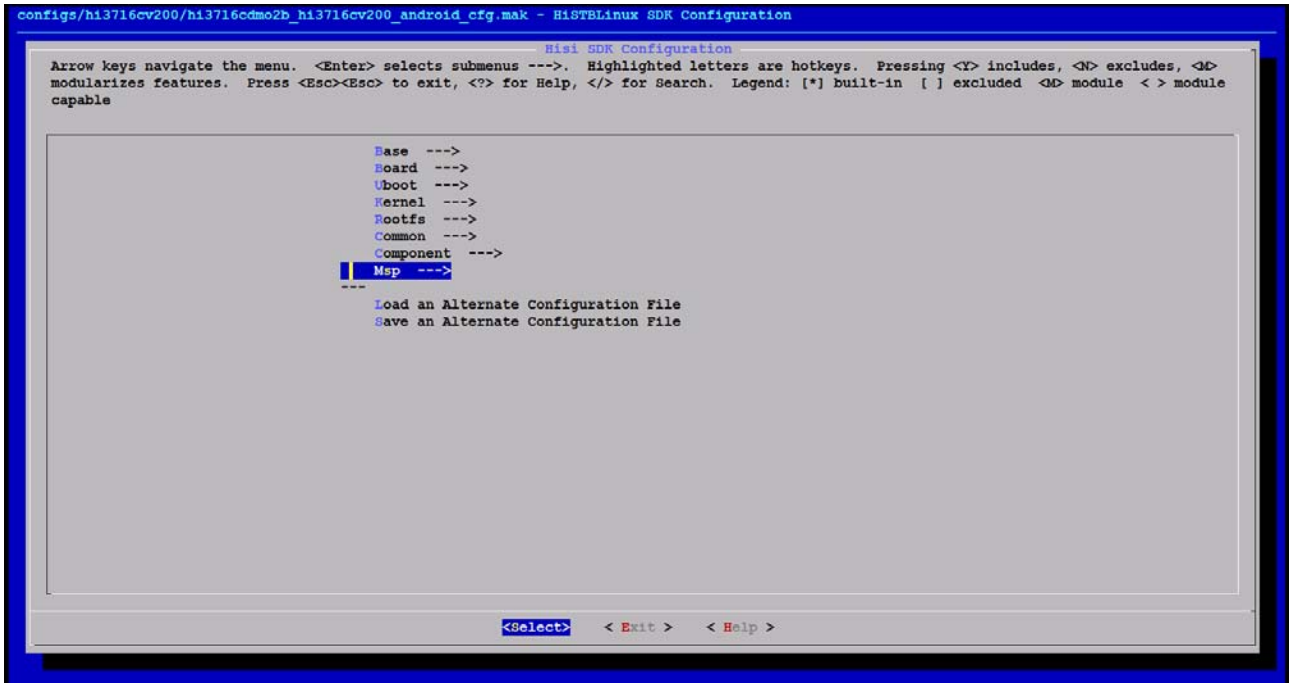
在此目录下的 BoardConfig.mk 文件中查找环境变量 HISI_SDK_ANDROID_CFG，此变量的值即为该产品使用的 SDK 配置文件。 Hi3798MV100 芯片的 SDK 配置文件为：hi3798mdmo1b_hi3798mv100_android_cfg.mak

步骤 2 进入 device/hisilicon/bigfish/sdk 目录，执行：

```
make menuconfig
SDK_CFGFILE=configs/hi3798mv100/hi3798mdmo1b_hi3798mv100_android_cfg.mak
```

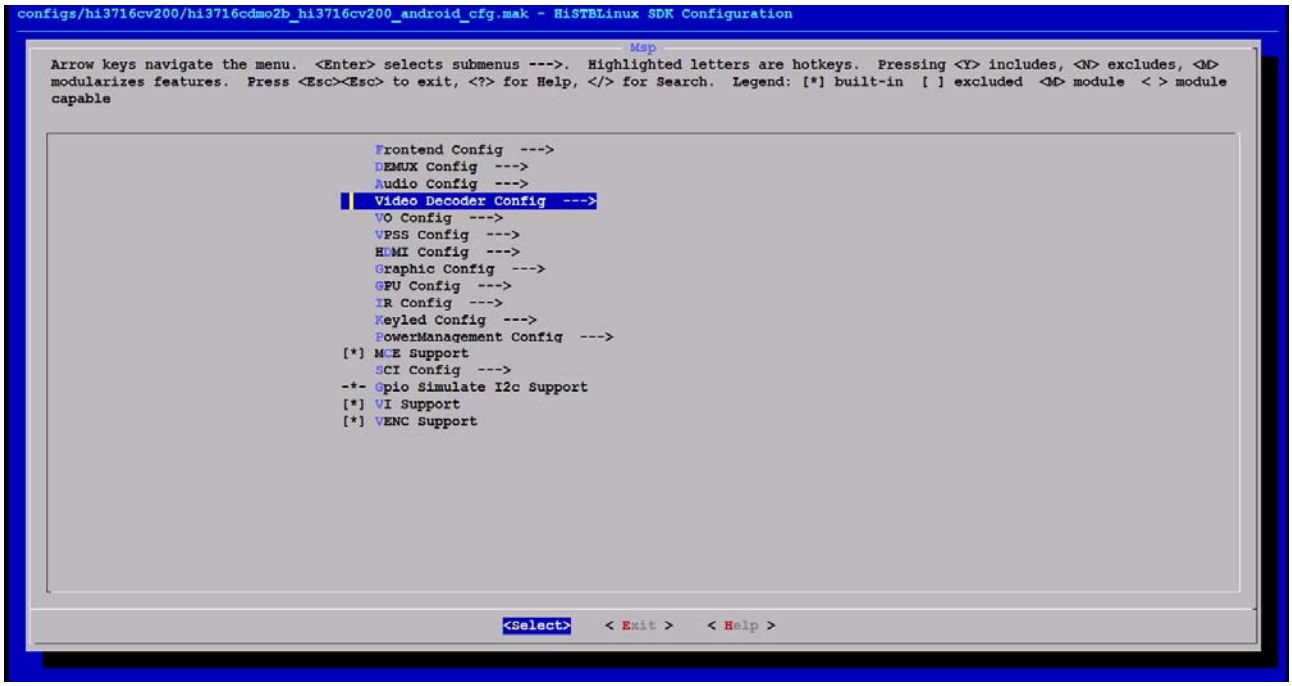
步骤 3 执行完上述命令后，进入类似如下页面：选择“Msp”。

图6-1 选择“Msp”



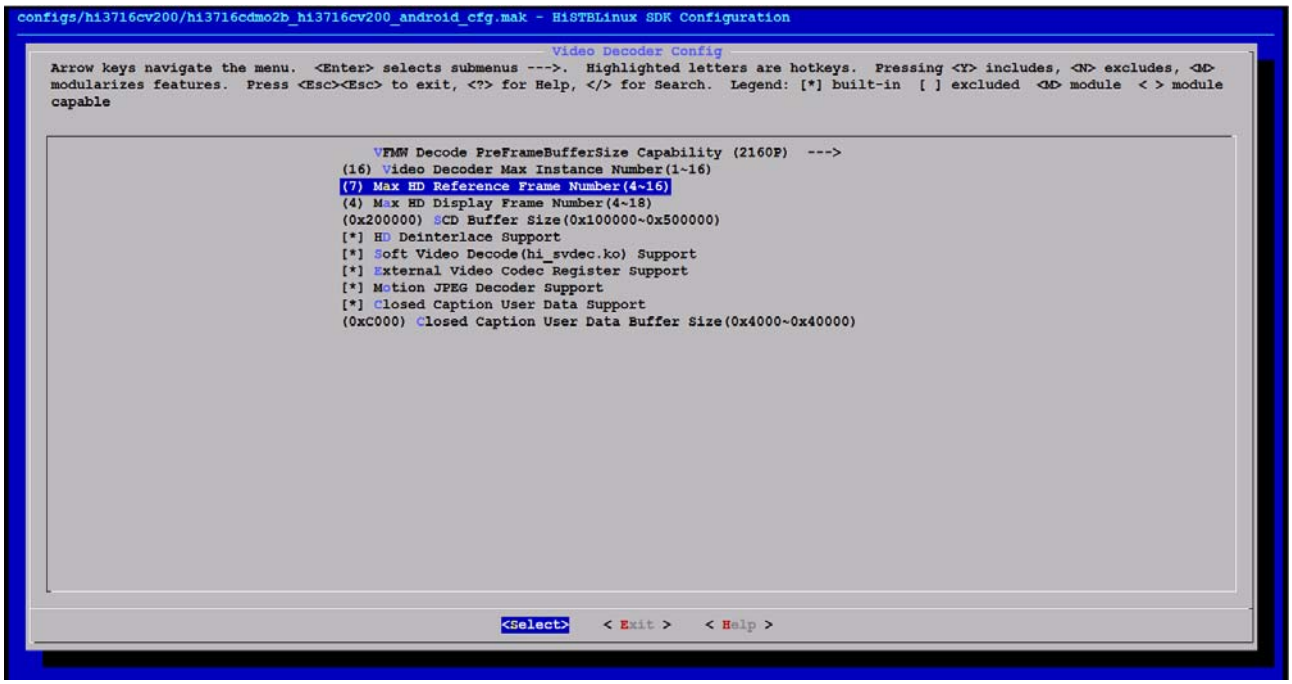
步骤 4 选择“Video Decoder Config”。

图6-2 选择“Video Decoder Config”



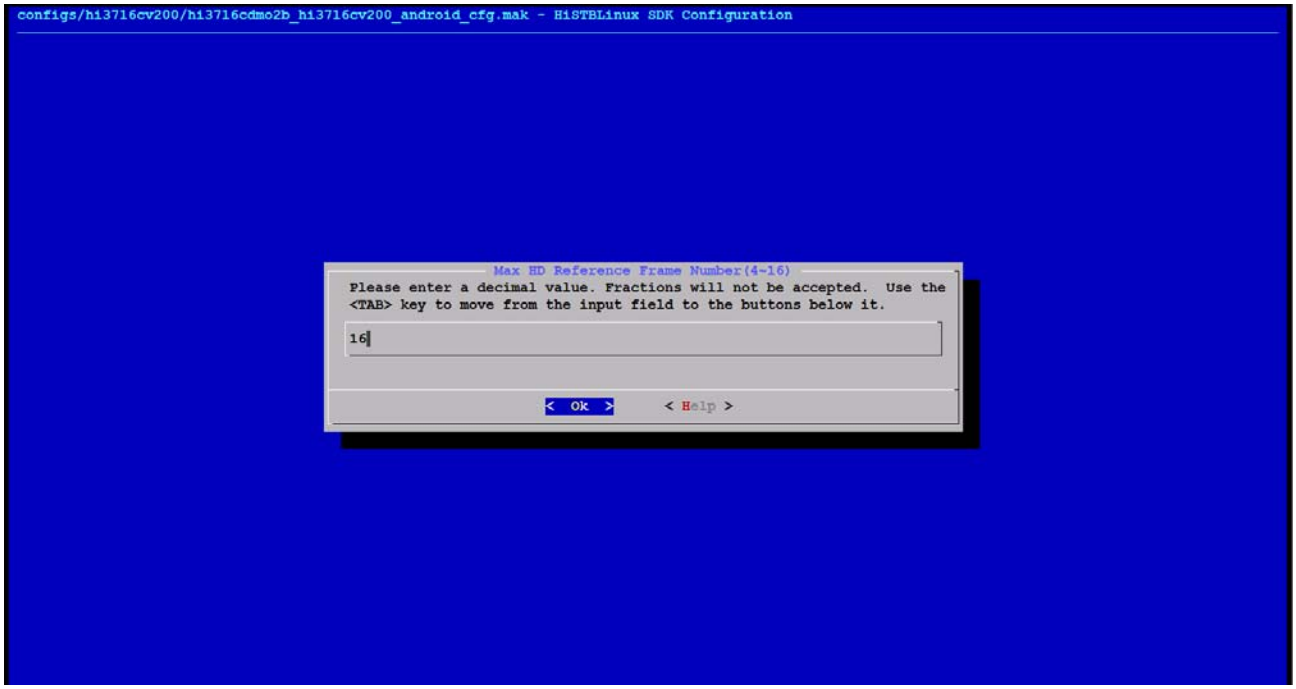
步骤 5 选择“Max HD Reference Frame Number(4~16)”。

图6-3 选择“Max HD Reference Frame Number(4~16)”



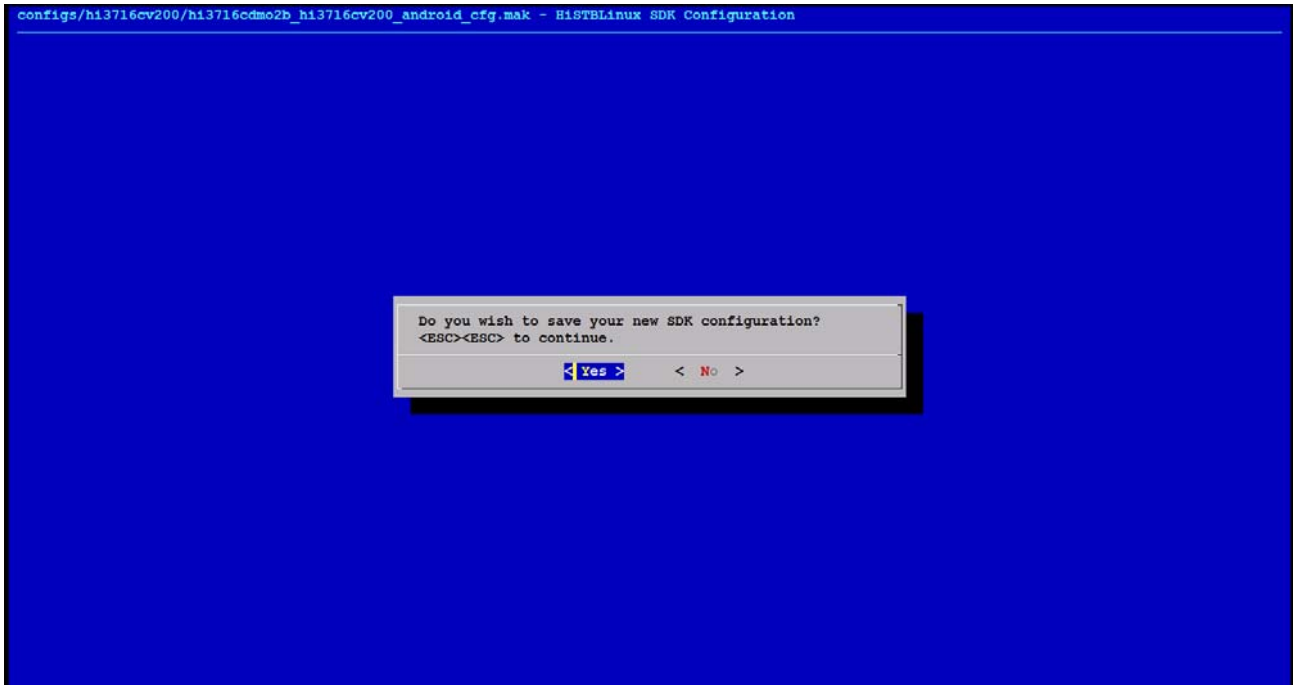
修改默认值为所需要的值即可，本例中直接修改为最大值“16”。

图6-4 修改默认值



步骤 6 保存，退出。

图6-5 保存并退出



至此，SDK 配置文件已修改完成，只需要重新编译内核即可支持多参考帧视频播放。重新编译内核方法参考代码根目录下 install_notes 中的相关说明。

----结束

6.2.3 如何让 HDMI 与 CVBS 同时输出

注意

- 目前 HDMI 与 CVBS 的互斥策略仅用于如下芯片：Hi3798MV100。
- Dolby 认证时，HDMI 与 CVBS 默认是同时输出的，可以不用参照本节的方式修改 persist.sys.cvbs.and.hdmi。

问题描述

Hi3798MV100 解决方案在默认条件下 HDMI 与 CVBS 是互斥的，即 HDMI 与 CVBS 不会同时输出：插入 HDMI 时，CVBS 不输出；拔掉 HDMI 后，CVBS 输出。该互斥策略的引入，是出于减少功耗的考虑。因为同时输出相比互斥输出，功耗要高 150mW 左右。

如果需要标清和高清同时显示，那么需要使 HDMI 与 CVBS 能够同时输出。那么，如何让 HDMI 与 CVBS 同时输出呢？

解决办法

为了能控制 HDMI 与 CVBS 是否同时输出，Android 中增加了一个系统 prop 属性 persist.sys.cvbs.and.hdmi。在编译 Android 前，支持使能或禁用 HDMI 与 CVBS 的互斥策略。

即修改 device/hisilicon/Hi3798MV100/customer.mk 文件中“persist.sys.cvbs.and.hdmi”的值。

- HDMI 与 CVBS 同时输出，配置如下：
  `persist.sys.cvbs.and.hdmi=true`
- HDMI 与 CVBS 互斥（默认情况），配置如下：
  `persist.sys.cvbs.and.hdmi=false`

6.2.4 如何让视频层和图形层的显示区域保持一致

说明

Android4.2 版本视频层和图形层的显示区域默认是保持一致的，Android4.4 版本视频层和图形层的显示区域默认是不一致的，该问题只针对 Android4.4 版本的。

问题描述

Android4.4 在设置页面和设置屏幕输出区域，OSD 对应有变化，但是使用本地播放播放视频时依然是全屏输出。

解决办法

当前 Android4.4 版本，通过 Settings 进入设置 Display，可以调整图形在电视上的显示区域。

默认情况下，该设置只对图形层显示起作用，视频播放不受这个设置选项的控制，默认都是以全屏进行视频播放。这样做主要是考虑保证视频的满屏播放。

如果客户想要视频层和图形层的显示区域一致，需要修改 SDK 配置文件，可以按照如下修改：

步骤 1 确认并选择配置文件。

产品使用的 SDK 配置文件的名称在各个产品 BoardConfig.mk 的 HISI_SDK_ANDROID_CFG 变量中定义，比如：

在 device/hisilicon/Hi3798MV100/BoardConfig.mk 中有：

```
HISI_SDK_ANDROID_CFG := hi3798mdmo1a_hi3798mv100_android_cfg.mak
```

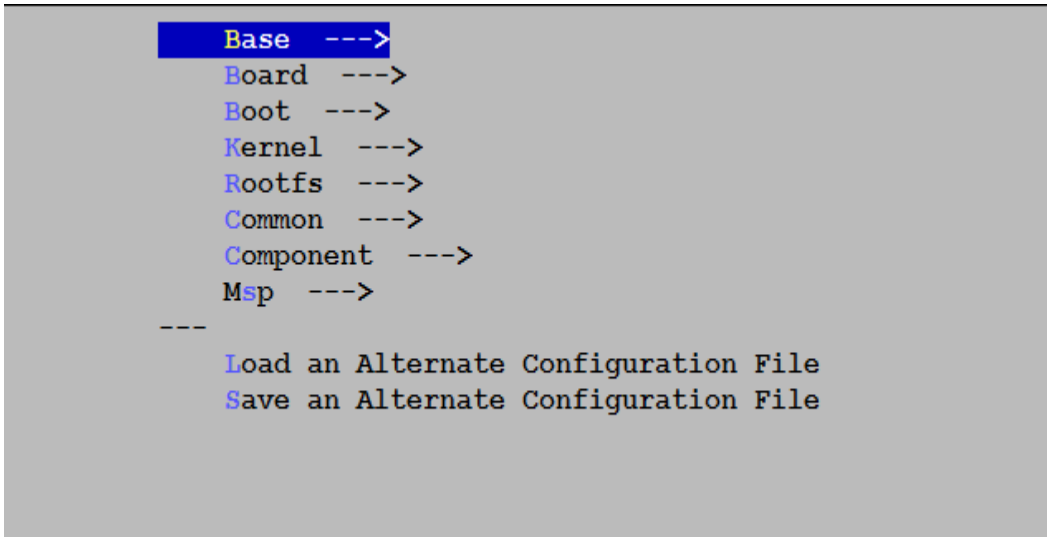
步骤 2 配置系统参数。

1. 确认好了配置文件后，进入目录 device/hisilicon/bigfish/sdk 执行命令"make menuconfig SDK_CFGFILE=configs/要修改的配置文件名"，会弹出修改配置的界面，如：

```
make menuconfig
SDK_CFGFILE=configs/hi3798mv100/hi3798mdmo1b_hi3798mv100_android_cfg.
mak
```

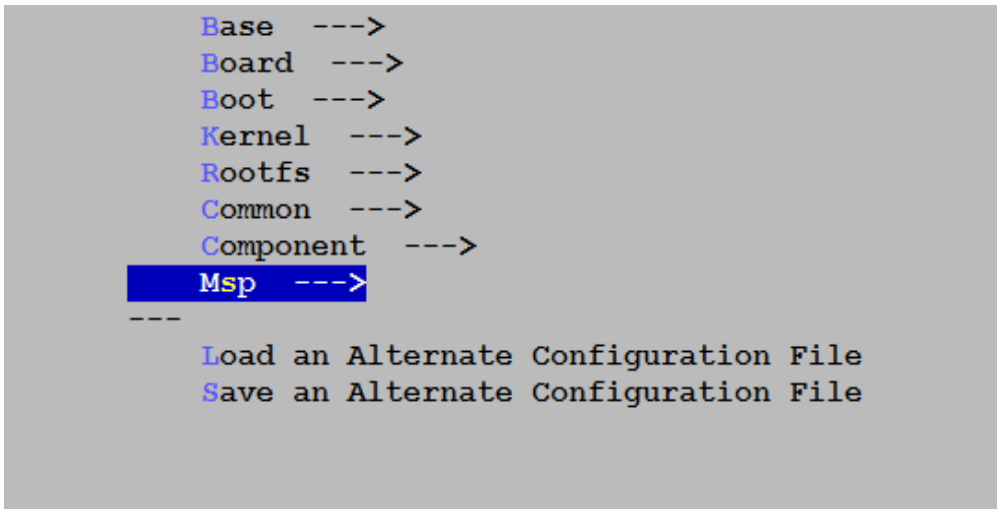
配置界面如图 6-6 所示。

图6-6 进入 SDK 配置界面



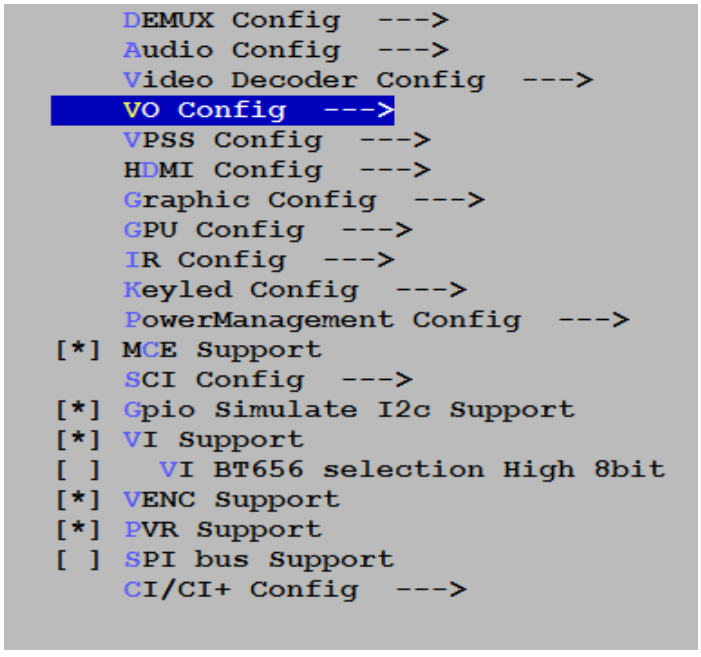
2. 选择 msp 项，键盘按下 Enter 键进入。

图6-7 选择 Msp 进入



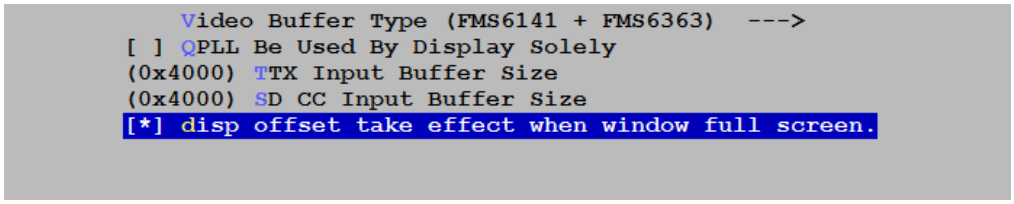
3. 选择 VO Config 项，键盘按下 Enter 键，进入该选项。

图6-8 选择 VO Config 进入



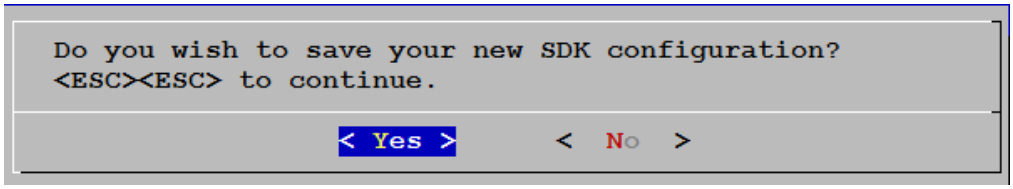
4. 选中“disp offset take effect when window full screen”，点击“Y”，选上这个选项。

图6-9 配置选项 disp offset take effect when window full screen



5. 配置完成后，点 ESC 退出该配置界面，弹出是否要保存配置，选择 YES。

图6-10 保存配置选项



步骤 3 重新编译烧写镜像。

完成配置后，回到源码的根目录下，清除以前编译过的镜像文件（若该份代码以前编译过的话），输入：

```
make clean
```

清除完成后，重新编译全部代码，顺序输入：

```
source build/envsetup.sh
```

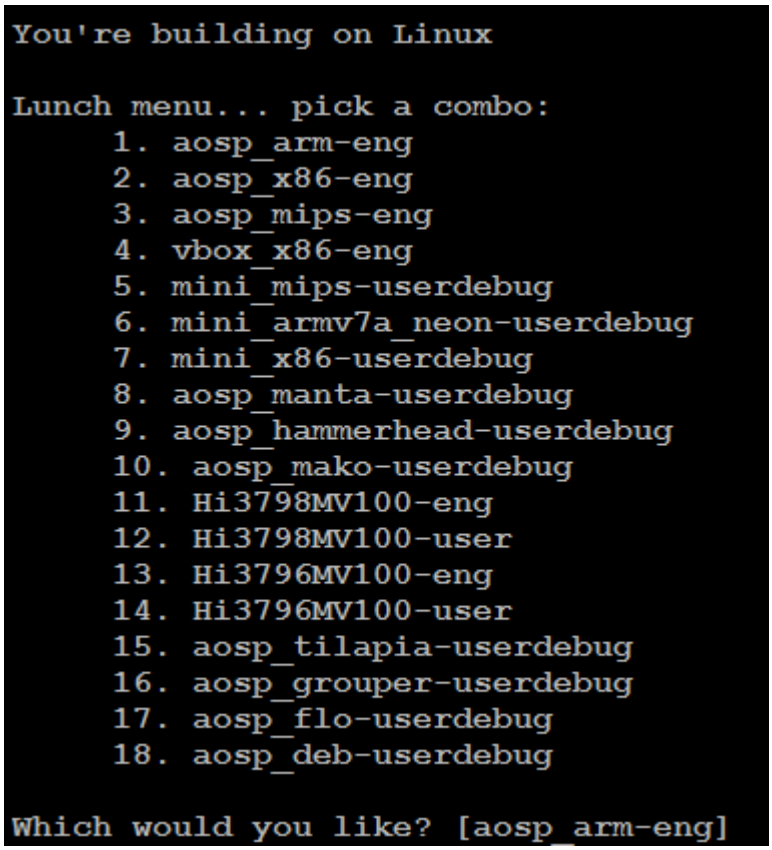
```
lunch Hi3798MV100-eng
```

```
make bigfish -j32
```

注意：针对上述第二个命令 `lunch Hi3798MV100-eng` 做以下说明：

`Hi3798MV100` 这个是芯片名称，后面跟的是将要编译的版本（此处为 eng 版本）。若是想编译其他的芯片（比如：Hi3796MV100）或者其他版本（比如：user 版本）的话，可以在该步骤直接输入：`lunch` 命令，系统会有如下打印。

图6-11 lunch 选项列表



在图 6-11 中选择想要编译的版本，键入他图中的序号（比如要编译 Hi3798MV100-eng，键入 11），然后回车即可。

等待编译完成后，烧录各个分区镜像文件即可。

就可支持在 Settings 中通过设置 Display 显示区域来调整视频播放显示的区域，即图形窗口和视频窗口是一致的。

----结束

7 应用类

7.1 恢复系统的状态栏显示？

问题描述

系统的默认状态栏怎么不见了？如何恢复状态栏的显示？

问题分析

很多客户在开发自己的产品时，由于系统的原生状态栏不符合其产品风格，客户一般都会重新定制开发新的状态栏，因此需要将原生的状态栏隐藏，不进行显示。为方便客户修改，默认将系统的原生状态栏进行了隐藏操作。

解决办法

如果要显示系统原生状态栏，请修改 frameworks/base/core/res/res/values/dimens.xml。

将以下代码恢复成 Android 原始代码：

```
<!-- Height of the status bar
       <dimen name="status_bar_height">25dip</dimen>
       -->
       <dimen name="status_bar_height">0dip</dimen>
   <!-- Height of the bottom navigation / system bar.
   <dimen name="navigation_bar_height">48dp</dimen>
       -->
       <dimen name="navigation_bar_height">0dp</dimen>
   <!-- Height of the bottom navigation bar in portrait; often the same
as @dimen/navigation_bar_height
   <dimen name="navigation_bar_height_landscape">48dp</dimen>
       -->
       <dimen
name="navigation_bar_height_landscape">@dimen/navigation_bar_height</dime
n>
```

恢复成如下 Android 原始代码：

```
<dimen name="status_bar_height">25dip</dimen>
<dimen name="navigation_bar_height">48dp</dimen>
<dimen name="navigation_bar_height_landscape">48dp</dimen>。
```

7.2 如何打开媒体扫描

问题描述

系统版本里有媒体扫描吗？

如何打开媒体扫描对 U 盘、硬盘进行扫描呢？

问题分析

基于在系统版本中的媒体扫描不响应 U 盘、硬盘插拔响应，默认只扫描/mnt/sdcard/和/system/media/目录。

解决办法

如果要打开媒体扫描，请修改：

```
/device/hisilicon/Hi3798MV100/device.mk
```

将以下代码:

```
# MediaScanner
PRODUCT_PROPERTY_OVERRIDES += \
    ro.mediaScanner.enable=false
```

修改为:

```
# MediaScanner
PRODUCT_PROPERTY_OVERRIDES += \
    ro.mediaScanner.enable=true
```

7.3 为何关闭 Android 原生动画效果

问题描述

Android 界面切换的时侯是没有动画效果的，那么为何将动画效果关闭？

问题分析

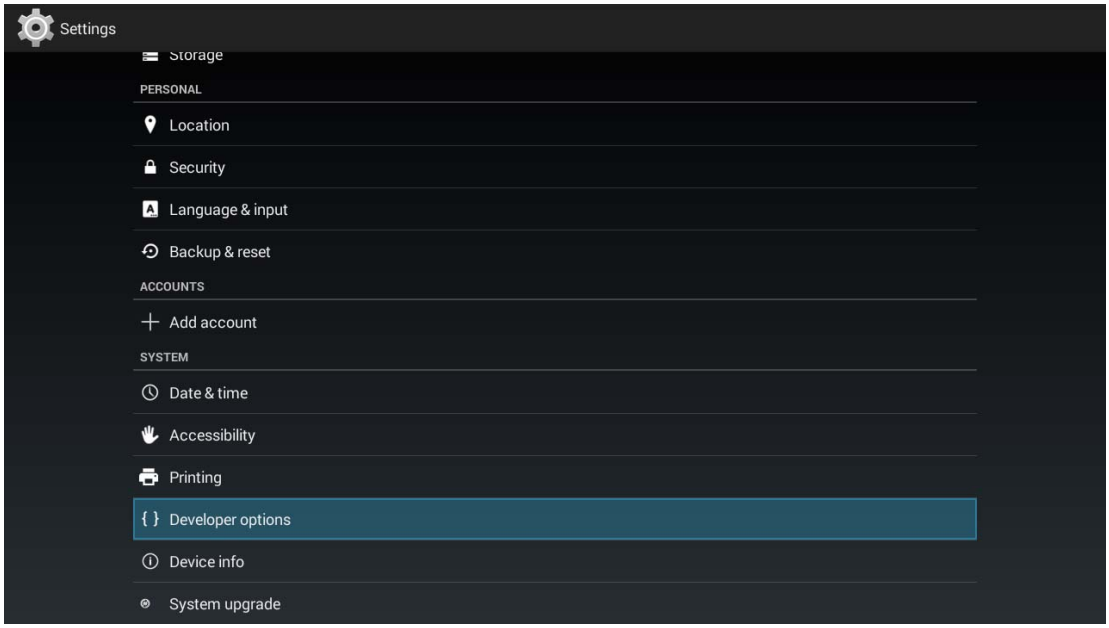
打开动画效果，则 Android 界面在切换的时侯，界面消失的过程可以看见，类似残留消失，体验性比较差。

解决办法

如何打开动画效果？具体操作步骤如下：

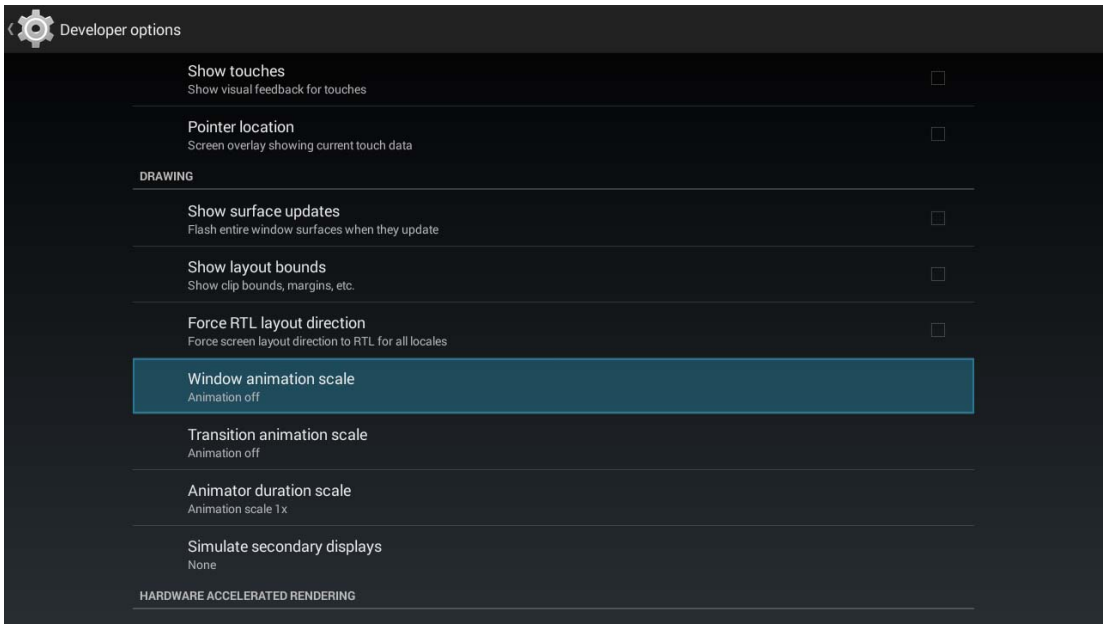
步骤 1　打开 settings 应用，点击 Developer options，如图 7-1 所示。

图7-1 打开 Developer options



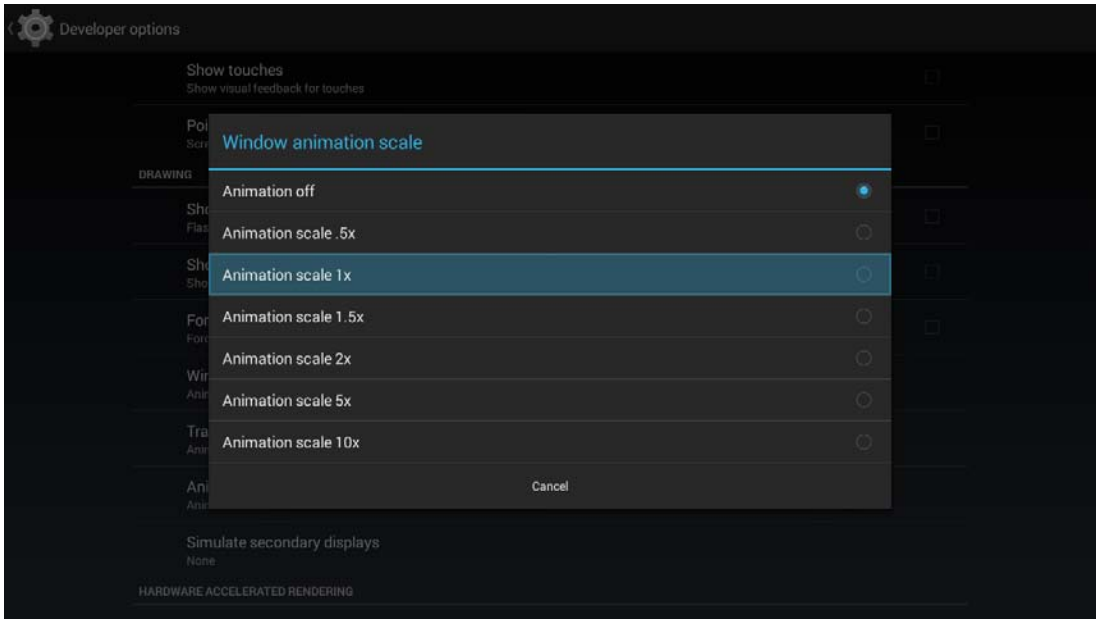
步骤 2　点击 Window animation scale，如图 7-2 所示。

图7-2 打开 Window animation scale



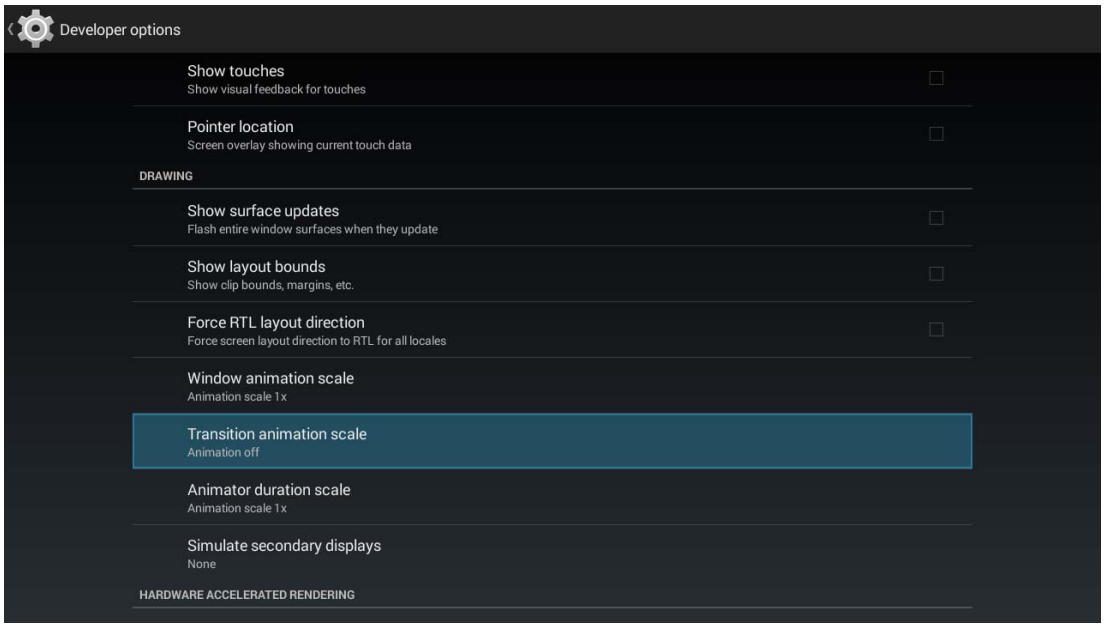
步骤 3　根据自己的需求选择窗口动画缩放的倍数（off 是关闭动画），如图 7-3 所示。

图7-3 设置窗口动画



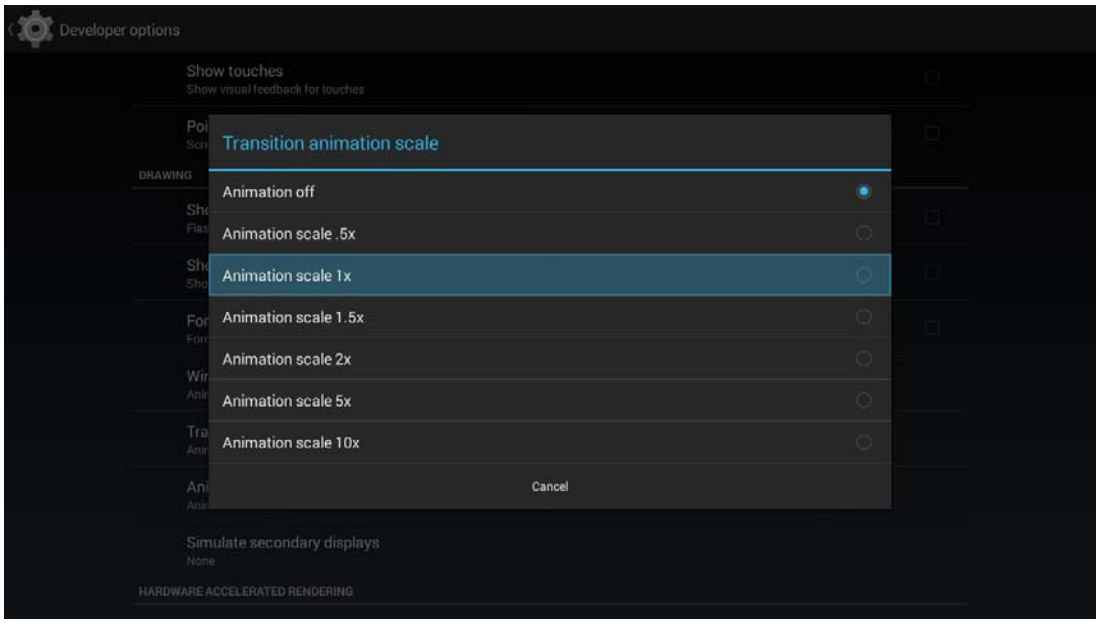
步骤 4　点击 Transition animation scale，如图 7-4 所示。

图7-4 打开 Transition animation scale



步骤 5　根据自己的需求选择过渡动画缩放的倍数（off 是关闭动画），如图 7-5 所示。

图7-5 设置过渡动画



步骤 6 至此，界面切换的动画效果设置完毕。

----结束

7.4 如何修改 DLNA 和 MultiScreen 绑定网卡的优先级

问题描述

在 Android 下 Ethernet 和 WiFi 是可以共存的，DLNA 和 MultiScreen 只会工作在一个指定的网卡上。目前在 Ethernet 和 WiFi 共存的时候，优先工作在 Ethernet 网卡上。如何修改指定 DLNA 和 MultiScreen 优先工作在 WiFi 上？

解决办法

DLNA 和 MultiScreen 需要修改如下文件：

- DMR

  device/hisilicon/bigfish/hidolphin/component/hidlna/android/packages/apps/HiMediaRender/src/com/hisilicon/dlna/dmr/UpnpBootBroadcastServiceDMR.java
- DMS

  device/hisilicon/bigfish/hidolphin/component/hidlna/android/packages/apps/HiMediaShare/src/com/hisilicon/dlna/dms/MediaService.java
- MultiScreen

  device/hisilicon/bigfish/hidolphin/component/himultiscreen/android/packages/apps/MultiScreenServer/src/com/hisilicon/multiscreen/server/MultiScreenService.java

将文件中的变量：

```
private final static String DEFAULT_USE_ADAPTER_WIFI_ETHERNET =
"eth";//"wlan";
```

默认为“eth”，标示 Ethernet 和 WiFi 共存的时候优先工作在 Ethernet 网卡上。

改为“wlan”即优先工作在 WiFi 上。

8 系统类

8.1 支持 512MB 内存设备

8.1.1 如何编译可用于 512MB 内存设备的版本

问题描述

设备只有 512MB 内存，如何编译一个可以在这种设备上运行的 Android 版本？

解决办法

请参考 install_notes_cn.txt 文件的 2.15 章节“为低 RAM（512MB）设备编译”。

说明
当前只有 Android4.4 及以上版本才支持 512MB 内存设备。

8.2 1080P UI

8.2.1 为何使用 1080P UI？

问题描述

Android 盒子设备采用 720P 分辨率下，应用界面图片效果不够清晰美观，厂商要求引入更高分辨率 UI。

问题分析

主要原因如下：

- android 4.4 CTS 认证要求。

  对于 OTT 电视属于可变像素设备，认证 CTS 可选择 720p（213DPI）和 1080p（320DPI）分辨率，排除了 160DPI；另外 720p（213DPI）设置对盒子应用页面布局很难适配，故 OTT 盒子 CTS 认证一般都用 1080p（320DPI）分辨率。

说明
附文档 android4.4-cdd.pdf 原文截图描述如图 8-1 所示。

图8-1 分辨率要求图

For example, a tablet that is 7" diagonal size with a 1024x600 pixel resolution is considered a fixed-pixel large mdpi display implementation. If it contains a video output port that displays at 720p or 1080p the device implementation MUST scale the output so that applications are only executed in a large mdpi window, regardless of whether the fixed-pixel display or video output port is in use.

Variable-Pixel Device Implementations

Variable-pixel device implementations MUST support one or both of 1280x720, or 1920x1080 (that is, 720p or 1080p). Device implementations with variable-pixel displays MUST NOT support any other screen configuration or mode. Device implementations with variable-pixel screens MAY change screen configuration or mode at runtime or boot-time. For example, a user of a set-top box may replace a 720p display with a 1080p display, and the device implementation may adjust accordingly.

Additionally, variable-pixel device implementations MUST report the following configuration buckets for these pixel dimensions:

- 1280x720 (also known as 720p): 'large' screen size, 'tvdpi' (213 dpi) density
- 1920x1080 (also known as 1080p): 'large' screen size, 'xhdpi' (320 dpi) density
- 3840x2160 (also known as 4K): 'large' screen size, 'xxxhdpi' (640 dpi) density

For clarity, device implementations with variable pixel dimensions are restricted to 720p, 1080p, or 4K in Android 4.4, and MUST be configured to report screen size and density buckets as noted above.

- 由于 android 当前大部分第三方应用开发仅考虑手机和 pad 等小屏幕设备，未考虑往后兼容 1080p（320DPI）电视设备及后续的 4K 电视，所以 1080p(320DPI)下系统测试应用兼容性差，240DPI 下兼容大部分基于手机及 pad 开发应用。
- 系统尚未做到同时使用不同 DPI 值。

解决办法

Hisi 平台切换 1080P UI 做法如下：

步骤 1 设置 base 分区虚拟屏幕大小为 1080P UI（宽：1920；高：1080）。

步骤 2 设置系统默认 240DPI，ro.sf.lcd_density 默认 240DPI。

步骤 3 应用需支持 320DPI 和 240DPI 应用布局和资源。

----结束

8.2.2 如何启用 1080P UI?

问题描述

Android 盒子切换到 1080P UI，需要考虑如下因素：

- 系统原生属性需要支持
- 系统内存需要增大
- 应用修改 320DPI 和 240DPI 两个布局支持

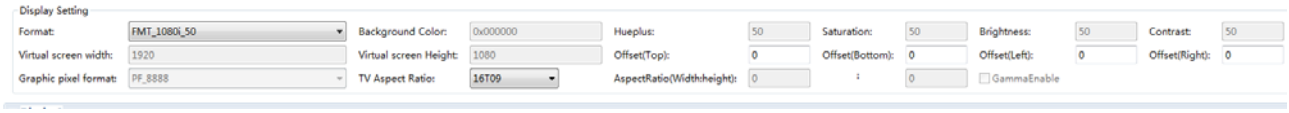
在这种情况下，如何启用 1080P UI 呢？

解决方法

切换 1080P UI 具体步骤如下：

步骤 1 使用 hitool.exe 工具制作 1080p 参数分辨率的 baseparam.img 分区。

图8-2 Baes 分区范例图



步骤 2 设置系统默认 DPI 值为 240DPI 或者 320DPI，以 Hi3798C 平台为例

修改 device\hisilicon\Hi3798CV100\device.mk，添加如下代码。

```
#setup 1080p UI default
PRODUCT_PROPERTY_OVERRIDES += \
    ro.sf.lcd_density=240
```

步骤 3 检查内存现状，1080P UI 内存参考值至少需要 1GB 内存。

步骤 4 所有系统应用和自研应用布局及资源图片修改。

----结束

8.2.3 如何使用动态 UI 切换功能

问题描述

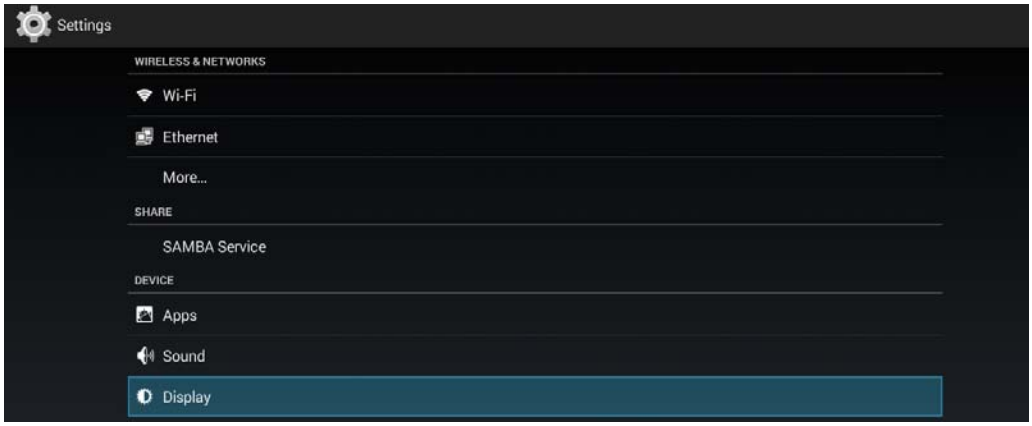
查看使用的平台是否支持动态切换 UI 功能，及如何使能平台支持动态切换 UI 功能。

解决方法

添加或检查动态 UI 切换功能步骤如下：

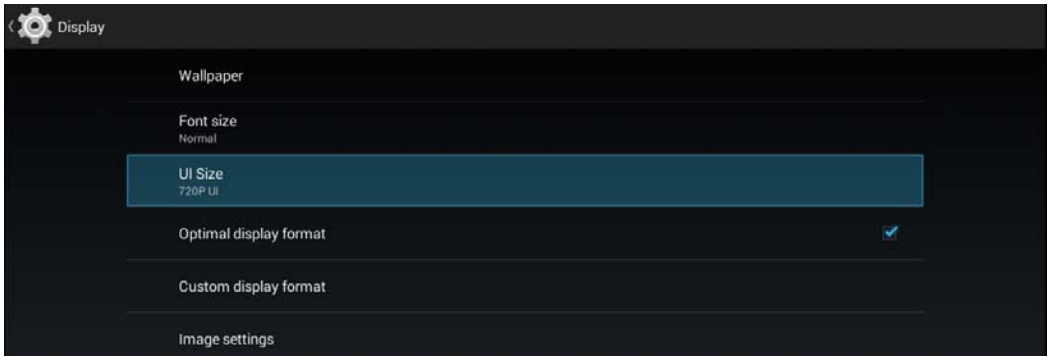
步骤 1 看 settings 应用显示选项：

图8-3 打开 settings 应用



步骤 2 进入 Display 栏。

图8-4 进入 Display 栏



如果有 UI Size 选项栏，切换功能就已具有。

步骤 3 检查功能控件是否已经打开，控件路径为：

packages\apps\Settings\res\xml\ display_settings.xml，名为 android:key="virtual_screen"；

查看 settings 应用是否添加支持动态切换 DPI 的平台参数，如果没有添加，参照如下方法 Hi3798C 平台添加红色部分代码：

修改为：

```
Setting/src/com/android/settings/DisplaySettings.java
public void onCreate(Bundle savedInstanceState) {
          ---
static final String CHIP_98C =
"type=HI_CHIP_TYPE_HI3798C;version=HI_CHIP_VERSION_V100";
if(!(CHIP_98C.equals(HiSdkinvoke.getChipVersion()))
&& !(CHIP_98M.equals(HiSdkinvoke.getChipVersion()))){
getPreferenceScreen().removePreference(mVirtualScreenPref);
}
```

```
---
}
```

步骤 4 重新编译替换原来 Settings 应用。

----结束

8.3 竖屏显示

注意

目前竖屏显示特性只在如下版本中支持：

Android 4.4.2 双核版本 HiSTBAndroidV500R001C01CP0003 及后续版本；

Android 4.2.2 双核版本 HiSTBAndroidV500R001C00CP0010 及后续版本；

Android 4.4.2 四核版本 HiSTBAndroidV600R001C00SPC060 及后续版本。

8.3.1 如何使用竖屏显示特性

问题描述

竖屏（portrait）显示特性，使 Android UI 和视频播放支持竖屏显示。如何使用竖屏显示特性？

解决办法

在使用竖屏显示特性时，需要注意如下 2 点。

- 在编译 Android 前需要进行配置，以使能竖屏显示特性。

  增加了一个系统 prop 属性 persist.sys.screenorientation，在编译时，支持配置横屏/竖屏显示；而在系统运行过程中，不支持动态切换显示方向。

  Android 4.2.2 与 Android 4.4.2 中，进行编译配置时，需要修改的文件有所差异：

  - 对于 Android 4.2.2，在编译 Android 前，修改 device/hisilicon/{CHIPNAME}/device.mk 中“persist.sys.screenorientation”的值。

    其中，{CHIPNAME}为 chip 的名字，例如：

    Hi3716CV200 芯片需要修改 device/hisilicon/Hi3716CV200/device.mk。

  - 对于 Android 4.4.2，在编译 Android 前，修改 device/hisilicon/Hi3798MV100/device.mk 中“persist.sys.screenorientation”的值。

    persist.sys.screenorientation 的修改方法如下：

    ➢ 使能竖屏显示

persist.sys.screenorientation=portrait

➢ 禁用竖屏显示（使能横屏显示）

persist.sys.screenorientation=landscape

- 开机 logo、fastplay、Android bootanimation 的竖屏显示，需要客户的支持。

  请在制作时，将 logo.img、fastplay.img、bootanimation.zip 做成竖屏。

  - 竖屏 logo.img 的制作方式。

    第一步，制作适应竖屏显示的图片。适应竖屏显示的图片相对于横屏的图片，画面顺时针旋转了 90°。图片格式可以选择 jpeg、png、bmp 或 gif。

    第二步，使用 HiTool 工具，打开 HiFastplay，选择“Logo 设置”，添加第一步制作的图片，保存镜像。从而生成竖屏的 logo.img。

  - 竖屏 fastplay.img 的制作方式。

    第一步，制作适应竖屏显示的 ts 码流。适应竖屏显示的 ts 码流相对于横屏的 ts 码流，画面顺时针旋转了 90°。ts 码流中 video 格式可以选择 MPEG2、MPEG4 或 H264；audio 格式可以选择 MP3 或 MP2。码流大小最大支持 50MB。

    第二步，使用 HiTool 工具，打开 HiFastplay，选择“本地播放设置”，添加第一步制作的 ts 码流，根据 ts 码流的信息设置“视频 PID”、“音频 PID”、“视频类型”和“音频类型”，保存镜像。从而生成竖屏的 fastplay.img。

  - 竖屏 bootanimation.zip 的制作方式。

    竖屏 bootanimation.zip 中所有的图片相对于横屏 bootanimation.zip 的图片，画面顺时针旋转了 90°。需要分别制作适应竖屏显示的图片，然后再按照 bootanimation.zip 的标准制作方式打包生成竖屏 bootanimation.zip。

8.4 支持开机优化

8.4.1 如何使能开机优化特性

问题描述

开机优化特性，支持安卓开机时间减少 3 秒。如何使能开机优化特性？

解决方法

在使用开机优化特性时，需要注意以下两点：

- 请在编译 Android 版本之前进行配置，以使能开机优化特性。
- 增加了 2 个系统属性 persist.sys.zygote.optimize 和 persist.sys.boot.optimize。在编译 Android 前请修改/device/hisilicon/{CHIPNAME}/customer.mk 中如下值：
  - persist.sys.zygote.optimize
  - persist.sys.boot.optimize

  true 为打开，false 为关闭。默认值为 true。

说明

{CHIPNAME}为 chip 的名字，例如：Hi3798MV100 芯片需要修改 device/hisilicon/Hi3798MV100/customer.mk。

在烧写完版本后，也可以在串口中进行配置，使用如下命令打开快速开机特性：

```
setprop persist.sys.zygote.optimize true 和 setprop persist.sys.boot.optimize true
```

使用如下命令关闭快速开机特性：

```
setprop persist.sys.zygote.optimize false 和 setprop persist.sys.boot.optimize false
```

使用如下命令获取这 2 个系统属性的值：

```
getprop persist.sys.zygote.optimize
getprop persist.sys.boot.optimize
```

其中：

- true 为打开；
- false 为关闭。