Rockchip
瑞芯微电子

Rockchip
双屏异显 开发指南

发布版本:1.0.0

日期:2018.01

前言

概述

Rockchip 处理器内双屏异显逻辑模块的功能、常用的接口和内部工作原理，通过实例介绍双屏异显模块的开发过程以及注意事项。

产品版本

芯片名称	SDK 版本
RK3288	Android 7.1

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

修订记录

日期	版本	作者	修改说明
2018.01.24	V1.0.0	LQH, XYP	第一次发布

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

目录

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

1 概述

鉴于双屏异显功能越来越广泛的应用到如车机，POS 机，收银机以及卡拉 OK 等项目上面。因此本文基于 RK 平台，从以下四个方面对双屏异显功能进行说明，主要介绍实现方法、系统 API、实现流程等，方便后续的客户进行开发。

1. RK 平台的双屏显示方案。
2. 谷歌 Presentation 原生双屏异显方案。
3. RK DualScreen 双屏异显显方案。
4. 基于 Presentation 的双屏异显异触方案。

2 RK 平台双屏显示方案

SOC 为双 LCDC 输出，并且显示框架上报给 Android 的显示设备为两个，是实现双屏异显功能的前提。RK3288 SOC 支持输出如 EDP、MIPI、LVDS、HDMI、RGB 等信号，其 7.1 SDK 已经默认支持 EDP+HDMI 的双屏异显功能，如需要做类似 MIPI+LVDS 或者 EDP+LVDS 的双屏功能，请参考《基于 drm 的 android 7.1 显示使用指南》的 2.6 章节进行配置即可，上层框架的异显部分无需进行任何更改。

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

3 RK 平台双屏异显方案

RK 双屏异显方案有两种方式：Android Presentation 和 RK dualscreen。

Android Presentation 是 Google 提供的双屏方案，实现了 View 级别的 VOP 派发，逻辑均在同一个 APP 上进行控制；RK dualscreen 则是实现了 APP 级别的 VOP 派发，异显的两部分分别是不同的 APP。

Presentation 比较适用于对自身需求进行深入定制的方案，RK dualscreen 在满足深入定制方案下，也支持，快速集成多方 APP，进行功能整合。两者各有优缺点，也能够进行互补。

（注：目前 RK dualscreen 方案支持异显异声，相关支持文档在 RKDocs/RKTools manuals/《RK3288 7.1 异显双声功能参考补丁.rar》，客户可根据自身产品需求进行开发）

3.1 Android Presentation 介绍

Presentation 是 Android 针对双屏异显所开发的一个类。它可以做到一个 APK 里面，通过给 Presentation 单独进行 view 的布局，来实现同一个 APK 在主屏和副屏上面显示不同的 view，来达到异显的效果。它的工作原理是通过调用 DisplayManagerService 的 getDisplays 方法来获取第二个显示设备。将第二个显示设备作为参数传给 Presentation，然后在 Presentation 里面实现自己的 UI 内容，最终调用 Presentation 的 show 方法来将 UI 内容显示在第二个显示设备上面。

3.1.1 API 介绍

官方 API 以及示例说明文档：

https://developer.android.com/reference/android/app/Presentation.html

国内文档介绍：

http://www.android-doc.com/reference/android/app/Presentation.html

Presentation 是 google 官方提供的一个特殊的 dialog 类型，用来将特定内容显示到其他非主屏显示器上。它在创建时，需要绑定显示屏，并根据这个目标显示屏以及其大小来配置 context 和 resource。目前系统提供了两种方式来与目标显示屏进行绑定。

1.通过 MediaRouter 接口获取并绑定：

```
    MediaRouter mediaRouter =(MediaRouter)
context.getSystemService(Context.MEDIA_ROUTER_SERVICE);
    MediaRouter.RouteInfo route = mediaRouter.getSelectedRoute();
    if(route !=null){
        Display presentationDisplay = route.getPresentationDisplay();
    if(presentationDisplay !=null){
        Presentation presentation =new MyPresentation(context,
presentationDisplay);
    presentation.show();
        }
    }
```

2.通过 DisplayManager 接口获取并绑定：

```
    DisplayManager displayManager =(DisplayManager)
context.getSystemService(Context.DISPLAY_SERVICE);
    Display[] presentationDisplays =
    displayManager.getDisplays(DisplayManager.DISPLAY_CATEGORY_PRESENTATION
```

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

```
);
    if(presentationDisplays.length >0){
        // If there is more than one suitable presentation display, then we could
consider
        // giving the user a choice.  For this example, we simply choose the first
display
    // which is the one the system recommends as the preferred presentation display.
        Display display = presentationDisplays[0];
        Presentation presentation =new MyPresentation(context,
presentationDisplay);
    presentation.show();
    }
```

MediaRouter 和 DisplayManager 接口不在此处详细介绍。

Presentation 的接口使用简单，功能明确，主要接口如下：

1.构造接口

（1）Presentation(Context outerContext,Display display)

接口说明：实例一个使用默认主题，绑定到目标 display 上的 Presentation 对象。

参数说明：

outerContext：application 的 context 对象；display：要绑定的目标 display，可以通过上述介绍的 MediaRouter 和 DisplayManager 接口获取。

（2）Presentation(Context outerContext, Display display, int theme)

接口说明：实例一个使用特定主题，并绑定到目标 display 上的 Presentation 对象。

参数说明：

outerContext：application 的 context 对象； display：要绑定的目标 display，可通过上述介绍的 MediaRouter 和 DisplayManager 接口获取； theme：窗口使用的主题资源。

2.方法接口

（1）Display getDisplay()

接口说明：获取当前 presentation 显示所在的目标屏。

（2）Resources getResources()

接口说明：

获取用于当前 presentation 的 Resources 对象。

（3）void onDisplayChanged()

接口说明：当 presentation 绑定的 display 发生变化（如大小、方向等）时，系统会回调此接口，并且系统将自动调用 cancel()接口，关闭此 presentation。

（4）void onDisplayRemoved()

接口说明：当 presentation 绑定的 display 被移除时，系统会回调此接口，并在此之后，系统会自动调

用 cancel()接口，关闭此 presentation。

（5）void show()

接口说明：

用于显示此 presentation，如果显示设备无法找到，调用此接口，将抛出 WindowManager.InvalidDisplayException 异常。

（6）void onStart()

接口说明：

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

当此 presentation 启动后，会调用此接口，类似 activity 的 onStart。

（7）void onStop()

接口说明：当此 presentation 正在走 stop 流程时，将会调用到此接口，类似 activity 的 onStop。

3.1.2 Presentation 示例

谷歌官方提供了 ApiDemo，用来展示各个 API 的使用，Presentation 在源码中的范例路径为：

development/samples/ApiDemos/src/com/example/android/apis/app/PresentationActivity.java。

关键代码（红色字体）介绍如下：

1. 获取目标显示设备

```
public void updateContents(){
clear();
String displayCategory = getDisplayCategory();
Display[] displays = mDisplayManager.getDisplays(displayCategory);
addAll(displays);
Log.d(TAG, "There are currently " + displays.length + " displays connected.");
for (Display display : displays) {

Log.d(TAG, "" + display);


}


}

private String getDisplayCategory() {
return mShowAllDisplaysCheckbox.isChecked() ?
null :DisplayManager.DISPLAY_CATEGORY_PRESENTATION;
}
```

通过 DisplayManager 接口获取 PRESENTATION 类型的显示设备，添加到队列当中并绑定到各个 View 当中。

2. 点击 CheckItem，创建 Presentation 并显示

```
public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked){
if (buttonView == mShowAllDisplaysCheckbox) { // Show all displays checkbox
was toggled.
mDisplayListAdapter.updateContents();
} else {
//Display item checkbox was toggled.
final Display display = (Display)buttonView.getTag();
if (isChecked) {
PresentationContents contents = new PresentationContents(getNextPhoto());
showPresentation(display, contents);
} else {
hidePresentation(display);
```

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

```
    }
    }
    }
```

通过 View 的 setTag 绑定对象的方法，将 display 对象绑定到具体 View 当中，并在需要时，通过 getTag 获取对应的 display 对象。

```
    private void showPresentation(Display display, PresentationContents contents) {
    final int displayId = display.getDisplayId();
    If (mActivePresentations.get(displayId) != null) { return;}
    Log.d(TAG, "Showing presentation photo #" + contents.photo + " on display #" +
displayId +".");
    DemoPresentation presentation = new DemoPresentation(this, display, contents);
    presentation.show();
    presentation.setOnDismissListener(mOnDismissListener);
    mActivePresentations.put(displayId, presentation);
    }
```

在范例中自定义了一个 Presentation 类，此处通过调用自定义的 Presentation，并显式调用 show 方法，将此 Presentation 显示到对应的 display 上。

此处的 PresentationContents 只是自定义了一个数据的封装方式，传给 Presentation，实际使用当中，可以根据项目需要，或者通过其他方式给显示的数据即可，不需要也创建这类对象。

3. 自定义 Presentation 类

```
    private final class DemoPresentation extends Presentation {
    final PresentationContents mContents;
    public DemoPresentation(Context context, Display display, PresentationContents
contents) {
    super(context, display);
    mContents = contents;
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    //Be sure to call the super class
    super.onCreate(savedInstanceState);
    }
    // Get the resources for the context of the presentation.
    // Notice that we are getting the resources from the context of the presentation.
    Resources r = getContext().getResources();
    setContentView(R.layout.presentation_content);
    final Display display = getDisplay();
    final int displayId = display.getDisplayId();
    final int photo = mContents.photo;
    // Show a caption to describe what's going on.

    TextView text = (TextView)findViewById(R.id.text);
    text.setText(r.getString(R.string.presentation_photo_text, photo, displayId,
display.getName()));
    // Show a n image for visual interest.
```

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

```
ImageView image = (ImageView)findViewById(R.id.image);
image.setImageDrawable(r.getDrawable(PHOTOS[photo]));
drawable.setShape(GradientDrawable.RECTANGLE);
drawable.setGradientType(GradientDrawable.RADIAL_GRADIENT);
// Set the background to a random gradient.
Point p = new Point();
getDisplay().getSize(p);
drawable.setGradientRadius(Math.max(p.x, p.y) / 2);
drawable.setColors(mContents.colors);
findViewById(android.R.id.content).setBackground(drawable);
}
}
```

如上，可看出 Presentation 跟 Activity 的继承类似，在构造中调用父类构造方法，在 onCreate 中调用 setContentView 方法设置显示的 View 布局。

以上便是 Android Presentation 的大体介绍，总体上来说，使用比较容易，其是 Dialog 类的子类，继承和使用与 activity 类似，比较重要的一个步骤是如何获取要绑定的 display 对象，实现了 View 级别的 vop 派发。

3.2 RK DualScreen 方案

RK DualScreen 与 Android Presentation 的区别，主要在于它实现了应用的派发，允许厂商快速根据现有的 app 功能，进行模块的集成，减少开发周期和研发成本。

3.2.1 RK DualScreen 操作流程

RK3288 Android 7.0 双屏异显功能的实现，主要通过对 framework/base/service 下的 wm 和 am 框架进行修改。以组合按键的方式触发异显的功能，将主屏上显示的应用派发到副屏进行显示，主屏默认显示 Stack 栈中的下一个 Task 的 Activity。再次按下组合按键将触发同显功能，副屏将显示和主屏相同的内容，同时将副屏的应用移到后台进行隐藏。在双屏异显开关开启状态下，可以通过副屏旋转开关（Vice screen rotation switch），来控制副屏的显示方向，LandScape 为横屏显示，Portrait 为竖屏显示。

目前 RK3288 Android 7.1 SDK 异显功能使能开关全部集成在“设置”应用中，可在“设置->显示->HDMI”界面进行操作，具体界面如图所示。

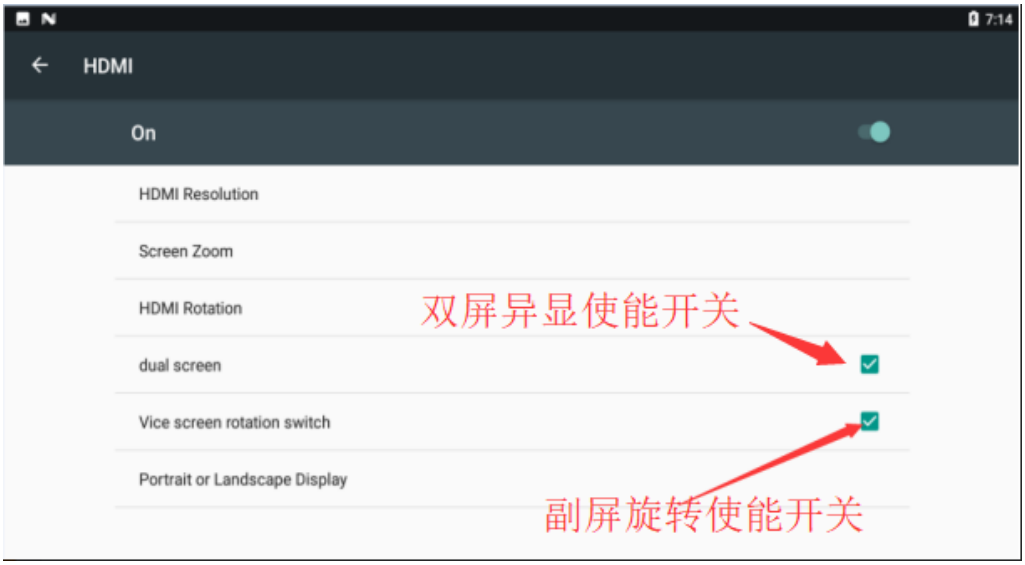

图 3-1 双屏异显设置界面

3.2.2 RK DualScreen API 介绍

RK DualScreen 双屏异显，让主屏和副屏显示不同的 App 应用，需要实现以下 2 点：

1. 在主副屏显示的 App 都必须保持 Resume 状态，并且有一个 Focus 状态的 Window。
2. 使应用 Layer 里的 mLayerStack 成员与 Display 对象 LayerStack 数值相等。两者的值相同，Layer 才会被输出到给该 Display 设备。而副屏的显示方向主要通过 DisplayDeviceInfo 和 DisplayInfo 来进行控制，DisplayDeviceInfo 可以理解为显示器的固有属性，而 DisplayInfo 为 Sensor 传给显示设备的信息。

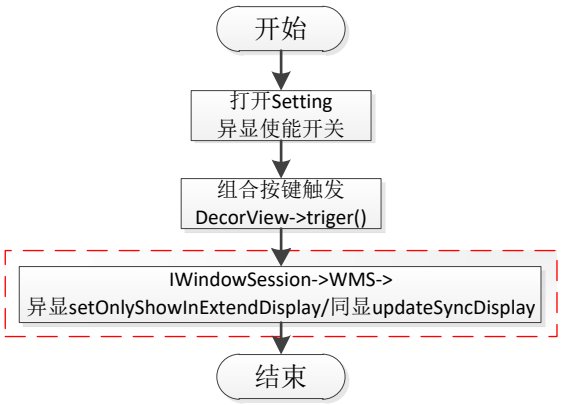

图 3-2 双屏异显触发流程

RK3288 7.1 SDK 双屏异显触发流程，如上图所示。主要分为三大部分：1.异显使能开关开启。2.通过组合按键触发。3.进入异显状态。其中第三部分为异显的核心部分，下面将分别介绍各部分涉及的 API 以及框架修改。

第一部分

主要通过 SettingsProvide 数据库，以及属性对异显功能进行控制。控制代码位于 packages/apps/Settings/src/com/android/settings/HdmiSettings.java，具体属性如下。

1. android.provider.Settings.DUAL_SCREEN_MODE

说明：异显功能全局开关，存放 SettingsProvider 数据库。

属性值：INT 类型，0：关闭异显功能；1：打开异显功能。

2. android.provider.Settings.DUAL_SCREEN_ICON_USED

说明：异显功能状态，存放 SettingsProvider 数据库。

属性值：INT 类型，0：处于同显状态；1：处于异显状态。

3. persist.orientation.vhshow

说明：旋转选项使能控制，系统属性。

属性值：BOOL 类型，true：旋转选项可选；false：旋转选项不可选取。

4. persist.orientation.vhinit

说明：副屏旋转方向，系统属性。

属性值：STRING 类型，0：横屏显示；1：竖屏显示。

第二部分

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

组合按键触发，主要就是通过一定的条件去调用 WSM 里面的 setOnlyShowInExtendDisplay 和 updateSyncDisplay 方法实现异显和同显功能。触发代码主要位于以下文件：

framework/base/core/java/com/android/internal/policy/DecorView.java。

1. public void initDualScreenConfig()：异显触发配置初始化

说明：通过读取 build.prop 文件的参数来设置触发所用的组合键以及间隔时间。

配置参数：sys.dual_screen.keycodes=24,25。24、25 分别是音量+、音量-对应的按键码。sys.dual_screen.interval=2000。触发时间间隔，单位为 ms。

2. public void trigerDualScreen()：触发异显

说明：该方法通过 Session 调用 wms 的 setOnlyShowInExtendDisplay 方法来进入异显流程。

3. public void trigerSyncScreen()：触发同显

说明：该方法通过 Session 调用 wms 的 updateSyncDisplay 方法来进入同显流程。

第三部分

主要是通过调用 WMS 里面的 setOnlyShowInExtendDisplay 和 updateSyncDisplay 方法实现异显和同显功能。其异显流程如图 4-3 所示，同显流程如图 4-4 所示。

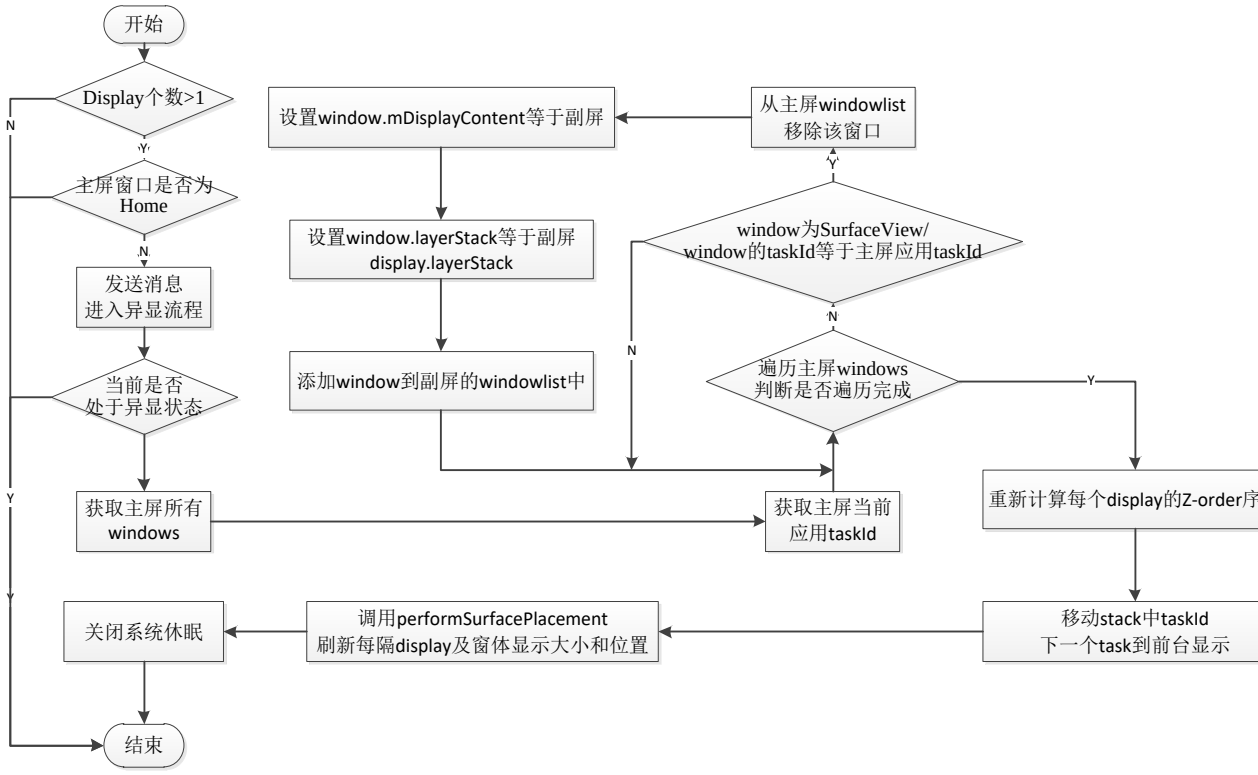

图 3-3 异显流程图

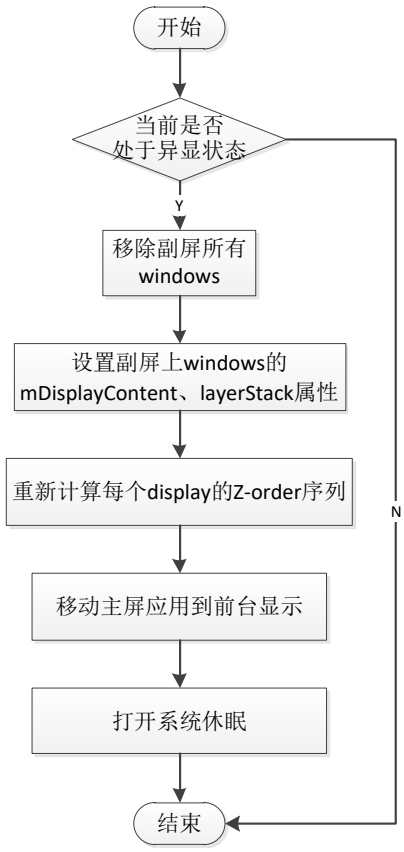

图 3-4 同显流程图

该部分的实现代码主要位于以下文件：

frameworks/base/services/core/java/com/android/server/wm/WindowManagerService.java。具体介绍如下：

1. public void setOnlyShowInExtendDisplay(Session session,IWindow client,int transit)

说明：对异显的先决条件进行判断。如：display 个数、当前 window 是否是 home 等。

2. public void moveTransitionToSecondDisplay(int groupId,int transit)

说明：将主屏上显示的应用派发到副屏进行显示，主屏默认显示 Stack 栈中下一个 Task 的 Activity，且禁止休眠功能。

3. public void updateDisplayShowSynchronization()

说明：退出异显功能，并将主屏应用唤起到前台，打开休眠功能。

4. public String getSecondPackageName()

说明：获取异显下副屏应用包名

5. public void switchFocusWindow(int taskId)

说明：移动 taskId 对应的 Activity 到前台

6. public boolean isHomeWindow(WindowState win)

说明：判断传入的 win 是否是桌面

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

返回值：true：Home 窗口；false：不是 Home 窗口

7. public void moveAppToBack(int taskId)

说明：移动 taskId 对应的 Activity 到后台

8. public int getLaunchTaskId()

说明：获取 Launch 的 TaskId

9. public boolean isShowDualScreen()

说明：通过读取 android.provider.Settings.DUAL_SCREEN_ICON_USED 判断当前是否是异显状态

返回值：True：处于异显状态；false：处于同显状态

10. public boolean isDualConfig()

说明：通过读取 android.provider.Settings.DUAL_SCREEN_MODE 判断当前是否是打开异显功能

true：打开异显功能 false：关闭异显功能

11. public List<Integer> getSecondTaskIds()

说明：获取副屏应用的 TaskId

12. public WindowList getSecondWindowState()

说明：获取副屏所有 WindowList

系统级应用可通过 WMS 服务对以上接口进行调用。系统级应用需要进行以下配置：

AndroidManifest.xml：

在 application 节点，添加 android:sharedUserId="android.uid.system"

Android.mk 文件：

添加 LOCAL_CERTIFICATE := platform

框架中实现双屏同时持有一个 resume 状态的 activity，主要通过修改 ActivityStack.java 的 resumeTopActivityInnerLocked 方法来实现。而副屏的旋转控制，主要通过修改 DisplayManagerService.java 的 performTraversalInTransactionLocked 方法，使用副屏的 DeviceInfo.rotation 属性来旋转副屏。其中的具体流程，不在此处详细说明。

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

4 RK 双屏双触摸方案

RK3288 Android 7.1 SDK 完成对双 Touch Pad 的适配后，默认支持 Presentation 方案下的双屏异显异触功能，目前已在 i2c+usb 和双 usb TP 的方式下经过验证。SDK 中对该部分的修改，主要位于以下目录：

frameworks/native/services/inputflinger，其功能实现的原理如下。

从原生的 Android 系统的代码中可以看到，触摸的事件实体中已经包含了一个叫做 displayId 的成员。这说明双触摸的框架 Android 基本已经做好，如果触摸事件的 displayId 对应的是主屏，那么它就会把该事件送给主屏的 TouchedWindow。同理，如果触摸事件的 displayId 对应的是副屏，那么它就会把该事件送给副屏的 TouchedWindow。所以关键的地方就是这个 displayId 是如何被赋值的，查看 inputflinger 的 Eventhub.cpp 代码，在 openDeviceLocked 中，如果是 USB 或者蓝牙接口的触摸屏，该触摸设备会设置一个 INPUT_DEVICE_CLASS_EXTERNAL 属性，那么 input 框架就是根据这个属性来最终将其的 event 送给副屏的 TouchedWindo。

```
1286:    // Determine whether the device is external or internal.
1287:    if (isExternalDeviceLocked(device)) {
1288:        device->classes |= INPUT_DEVICE_CLASS_EXTERNAL;
1289:    }
```

所以如果使用触摸屏，主屏使用的是 I2C 接口，副屏使用的是 USB 或者蓝牙接口。那么就可以实现双触摸。如果两个都使用的是 I2C，或者都使用的是 USB（蓝牙），只要修改代码保证主屏上的触摸设备不带 INPUT_DEVICE_CLASS_EXTERNAL 属性，副屏上的触摸设备带上这个属性，也可以实现双触摸功能。

```
00513:  struct MotionEntry : EventEntry {
00514:      nsecs_t eventTime;
00515:      int32_t deviceId;
00516:      uint32_t source;
00517:      int32_t action;
00518:      int32_t flags;
00519:      int32_t metaState;
00520:      int32_t buttonState;
00521:      int32_t edgeFlags;
00522:      float xPrecision;
00523:      float yPrecision;
00524:      nsecs_t downTime;
00525:      int32_t displayId;
00526:      uint32_t pointerCount;
00527:      PointerProperties pointerProperties[MAX_POINTERS];
00528:      PointerCoords pointerCoords[MAX_POINTERS];
```

5 Q&A

5.1 异显功能无法启动

请确保在 Settings 中打开相应的使能开关，同时根据其触发方式，进行异显功能的开启。默认触发方式可通过 system/build.proc 进行配置，使用音量+（KeyCode 24）和音量-（KeyCode25）进行触发，按键间隔为 2s。

5.2 查看当前 layer 显示大小、位置与状态。

在双屏异显状态下，进入 shell，执行 dumpsys SurfaceFlinger 可以看到当前系统有两个 display。每个 display 的信息从这段 log 可以清晰的看到，如 Display[0]有 3 个 layer(不包括 HWC_FRAMEBUFFER_TARGET)，而 Display[1]则有 1 个 layer。可以看出此时两个 display 显示的内容并不相同，并且 Display[1]也比 Display[0]少了状态栏与导航栏这两个 layer。很明显，这是在异显状态时抓取的 dumpsys SurfaceFlinger 信息。

```
numHwLayers=4, flags=00000000
   type   |  handle  | hint | flag | tr | blnd |   format    |    source crop (l,t,r,b)     |          frame          | name
----------+----------+------+------+----+------+-------------+------------------------------+-------------------------+-----
      HWC | ac5a0dc0 | 0000 | 0000 | 00 | 0100 | RGBA_8888   |    0.0,    0.0, 1440.0, 2048.0 |   96,    0, 1536, 2048 | com.android.rk/com.android.rk.RockExplorer
      HWC | ac964400 | 0000 | 0000 | 00 | 0105 | RGBA_8888   |    0.0,    0.0,   48.0, 2048.0 | 1488,    0, 1536, 2048 | StatusBar
      HWC | ac964360 | 0000 | 0000 | 00 | 0105 | RGBA_8888   |    0.0,    0.0,   96.0, 2048.0 |    0,    0,   96, 2048 | NavigationBar
FB TARGET | ac963f00 | 0000 | 0000 | 00 | 0105 | RGBA_8888   |    0.0,    0.0, 1536.0, 2048.0 |    0,    0, 1536, 2048 | HWC_FRAMEBUFFER_TARGET
Display[1] configurations (* current):
  * 0: 1280x720, xdpi=213.000000, ydpi=213.000000, refresh=16666666, colorMode=0
numHwLayers=2, flags=00000000
   type   |  handle  | hint | flag | tr | blnd |   format    |    source crop (l,t,r,b)     |          frame          | name
----------+----------+------+------+----+------+-------------+------------------------------+-------------------------+-----
      HWC | ac5a0f00 | 0000 | 0000 | 00 | 0100 | RGBA_8888   |    0.0,    0.0, 2048.0, 1536.0 |    0,    0, 1280,  720 | com.android.settings/com.android.settings.Settings
FB TARGET | ac964860 | 0000 | 0000 | 00 | 0105 | RGBA_8888   |    0.0,    0.0, 1280.0,  720.0 |    0,    0, 1280,  720 | HWC_FRAMEBUFFER_TARGET
DrmCompositor stats:
```

图 5-1 异显 SurfaceFlinger 信息

5.3 查看异显状态下 actvity 和 windows 状态

在异显状态下，每个 Display 上都有一个 resume 状态的 activity 和一个 focus 状态的 window，可以通过 dumpsys activity activities 命令和 dumpsys window windows 命令分别查看 activity 和 window 状态是否正确，结合 logcat 分析状态错误原因。

5.4 副屏旋转功能无效

请确保在 Setings 中打开 Dual Screen 开关以及 Vice Screen Rotation Switch 开关，设置副屏方向，此时副屏旋转方向不会立即生效，需要通过旋转主屏或者触发异显使副屏进行旋转。

5.5 打上异显补丁后，机器部分功能异常

请在测试时，将 Setting 设置中的异显开关关闭，再进行测试。排查是否为异显功能对其他部分功能造成的影响。

5.6 接入两个屏幕，屏幕无法点亮

RK 双屏异显方案，都只针对上层框架进行修改，如果屏幕无法正常点亮使用，请参考文中提

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.

到的《基于 drm 的 android 7.1 显示使用指南》进行配置修改，该部分与双屏异显框架无直接的相关性。

5.7 双屏异触功能无效

目前异触方案只支持 presentation 的异显方案，同时请确认外接的 Touch Pad 功能正常，实现触摸屏与显示屏的正确匹配，如出现匹配偏差等问题，与异触功能无直接关系，请咨询触摸屏硬件厂商进行调试修改。

5.8 打开开发者选项中的触摸指针测试异触功能，触摸指针无法显示在副屏

使用该方法进行异触功能测试，在使用 Presentation 异显后，触摸指针只能显示在主屏上，这是正常现象。触摸指针不支持在 Presentation 异显的 View 视图上进行绘制。可以通过 Presentation 异显视图上的 Button 等控件来进行测试异触的响应功能。

Copyright 2018 @Fuzhou Rockchip Electronics Co., Ltd.