

密级状态：绝密() 秘密() 内部() 公开(√)

Rockchip_双屏显示旋转方向调试文档

(技术部)

文件状态： [] 正在修改 [√] 正式发布	当前版本：	V1.01
	作 者：	李斌、董正勇
	完成日期：	2019-2-26
	审 核：	黄德胜、张文平
	完成日期：	2019-2-26

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Electronics Co. , Ltd

(版本所有, 翻版必究)

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.00	李斌/董正勇	2019-2-01	正式发布	
V1.01	董正勇	2019-2-26	增加设置界面操作说明	

目 录

前 言.....	1
1 概述.....	2
1.1 基础介绍.....	3
1.1.1 主副屏相关.....	3
1.1.2 横竖屏相关.....	7
1.2 属性介绍.....	8
2 场景调试说明.....	10
2.1 双屏同显.....	10
2.1.1 主副屏横竖屏属性一致，且宽高比相同.....	10
2.1.2 主副屏横竖屏属性一致，且宽高比不同.....	11
2.1.3 主副屏横竖屏属性不一致.....	13
2.2 双屏异显.....	15
2.3 补丁介绍.....	16

前 言

概述

本文档主要介绍**双屏同显**，**双屏异显**场景中存在的主副屏旋转调试方法及补丁，相关同事可查阅此文档进行调试。

产品版本

芯片名称	Android 版本
RK3399	Android 7.1
RK3328	/
RK3326 / PX30	/
RK3288	Android 7.1
RK312x	/

读者对象

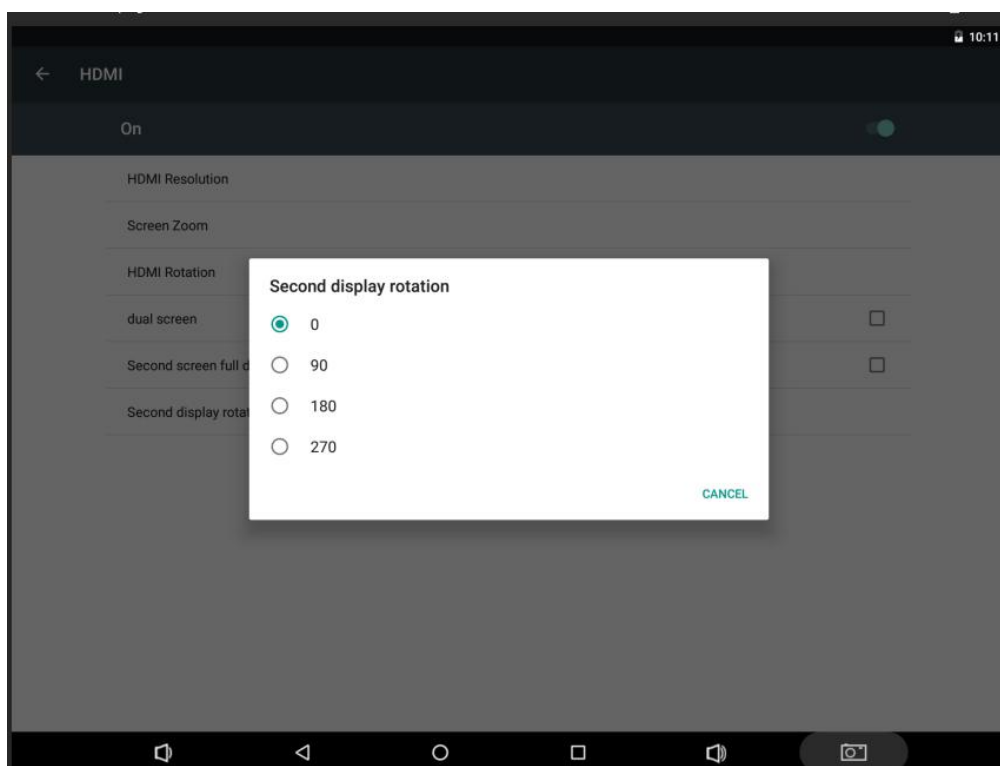
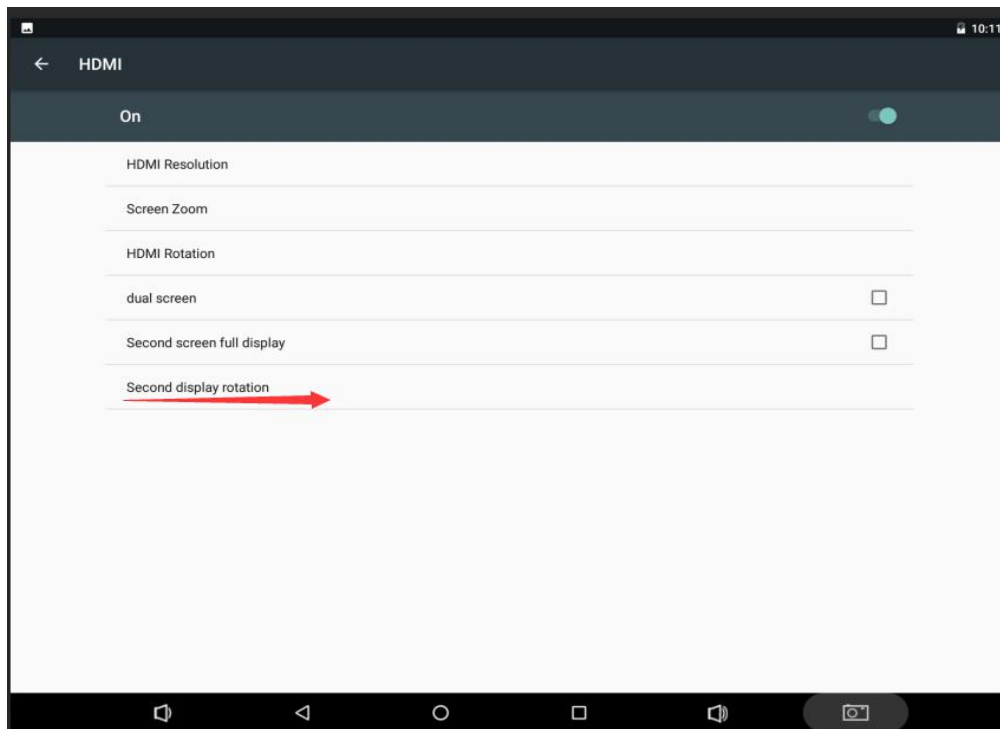
本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

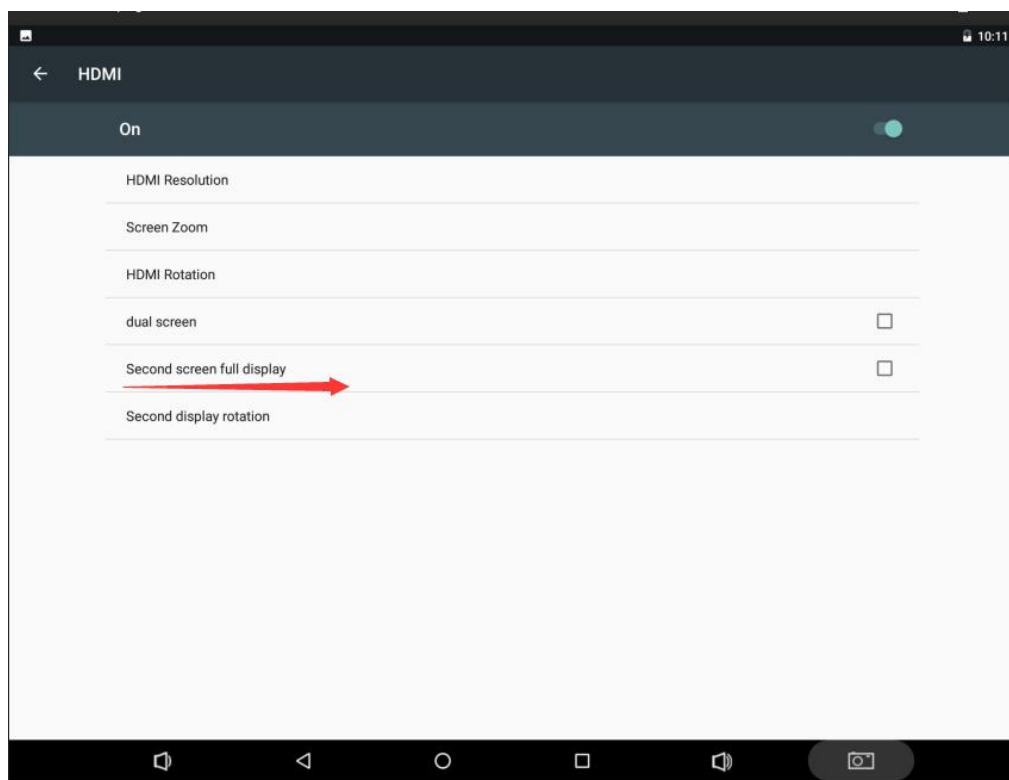
1 概述

由于产品存在屏幕物理尺寸与横竖屏属性差异，还有多屏同显与异显的不同需求，从而导致多屏显示场景的需求复杂，故整理如下调试文档并在设置界面增加对应控制选项，供产品工程师，FAE及用户查阅，方便调试。

设置->显示->HDMI 界面中增加副屏旋转方向控制选项，如下：



设置->显示->HDMI 界面中增加对副屏是否满屏的支持选项，如下：



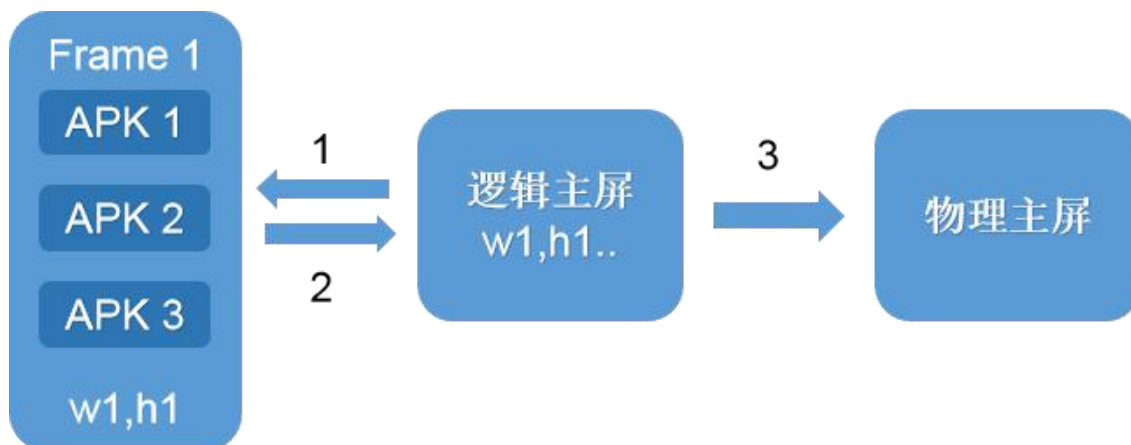
如上界面操作主要通过设置 `persist.sys.rotation.einit`, `persist.sys.rotation.efull` 属性来控制相关屏幕方向，通过界面操作可方便调试工作，最终通过 `getprop` 来获取相关属性的值（具体属性说明可查看属性介绍）。

1.1 基础介绍

1.1.1 主副屏相关

系统逻辑屏分为主屏，副屏与虚拟屏（暂不考虑），对应到实际的物理屏幕，系统中的逻辑屏属性与显示通路会根据显示需求而存在差异，比如双屏同显与双屏异显，在 1.1.1.1 与 1.1.1.2 作简单介绍。

若系统只有一个逻辑屏，即逻辑主屏，则系统的显示通路如下：

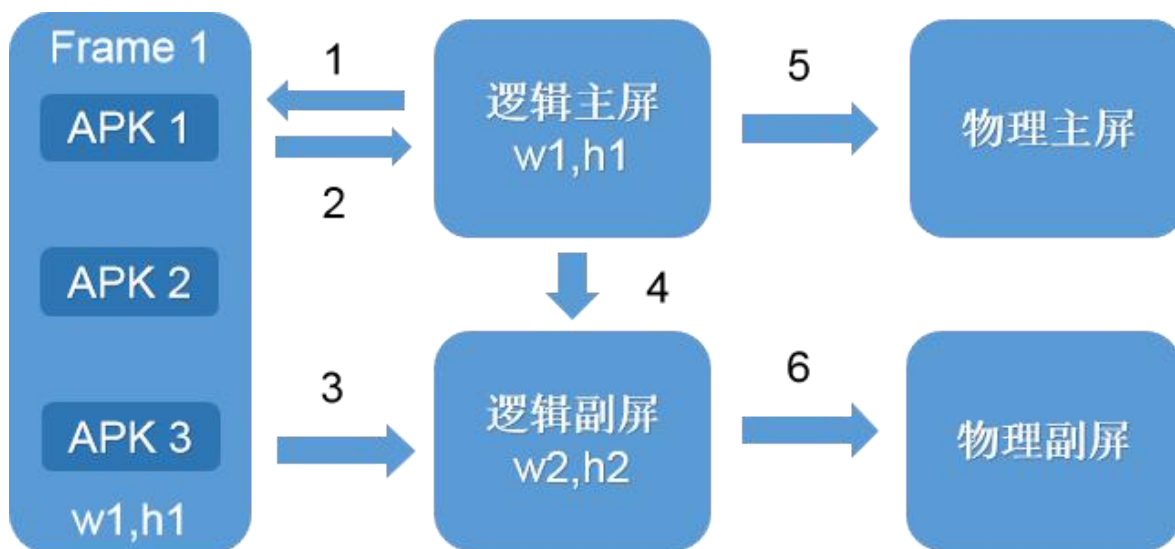


- 1: APK 获取逻辑主屏的属性信息（宽/高/刷新率），作为 APK 的渲染布局；
- 2: 多个 APK 将显示请求提交给逻辑主屏，主屏根据层级结构进行混合；
- 3: 逻辑主屏将混合的结果送显至实际的物理主屏，默认情况下，逻辑主屏的属性信息与物理主屏一致；

备注：由于 APK 通过主屏的属性信息进行渲染绘图的，所以在只有一个逻辑主屏的情况下，显示效果是正常的，不存在拉伸等情况。

1.1.1.1 双屏同显

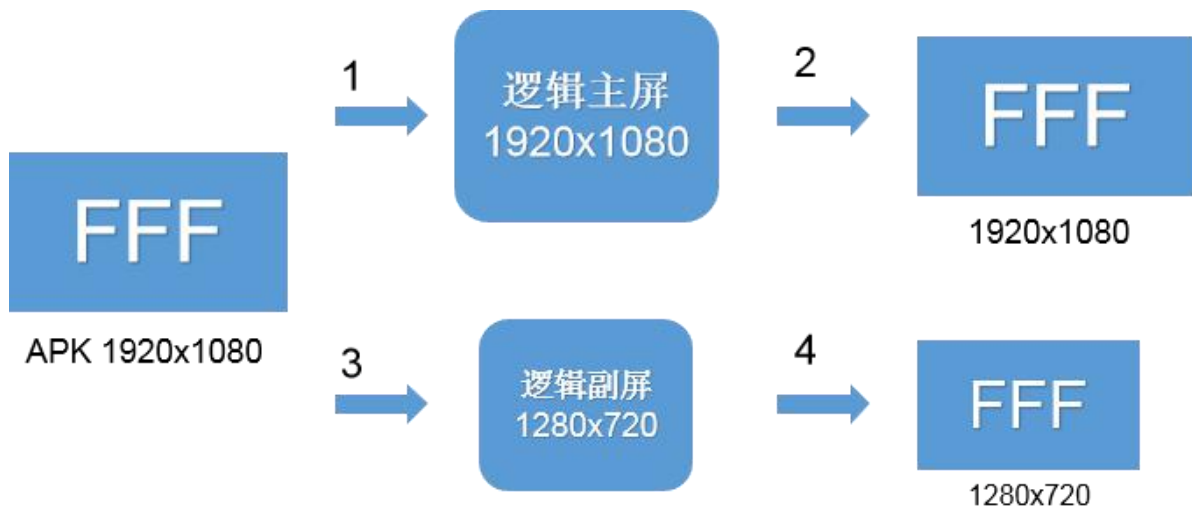
双屏同显，实际上是逻辑副屏将逻辑主屏的显示内容进行送显至逻辑副屏，显示通路如下：



- 1: APK 获取 **逻辑主屏** 的属性信息（宽/高/刷新率），作为 APK 的渲染布局；
- 2: 多个 APK 将显示请求提交给 **逻辑主屏**，主屏根据层级结构进行混合；
- 3: 多个 APK 将显示请求提交给 **逻辑副屏**，副屏根据层级结构进行混合；
- 4: 3 步骤其实可以等效为逻辑主屏将显示内容提交给逻辑副屏；
- 5: **逻辑主屏** 将混合的结果送显至实际的物理主屏，默认情况下，逻辑主屏的属性信息与物理主屏一致；
- 6: **逻辑副屏** 将混合的结果送显至实际的物理副屏；

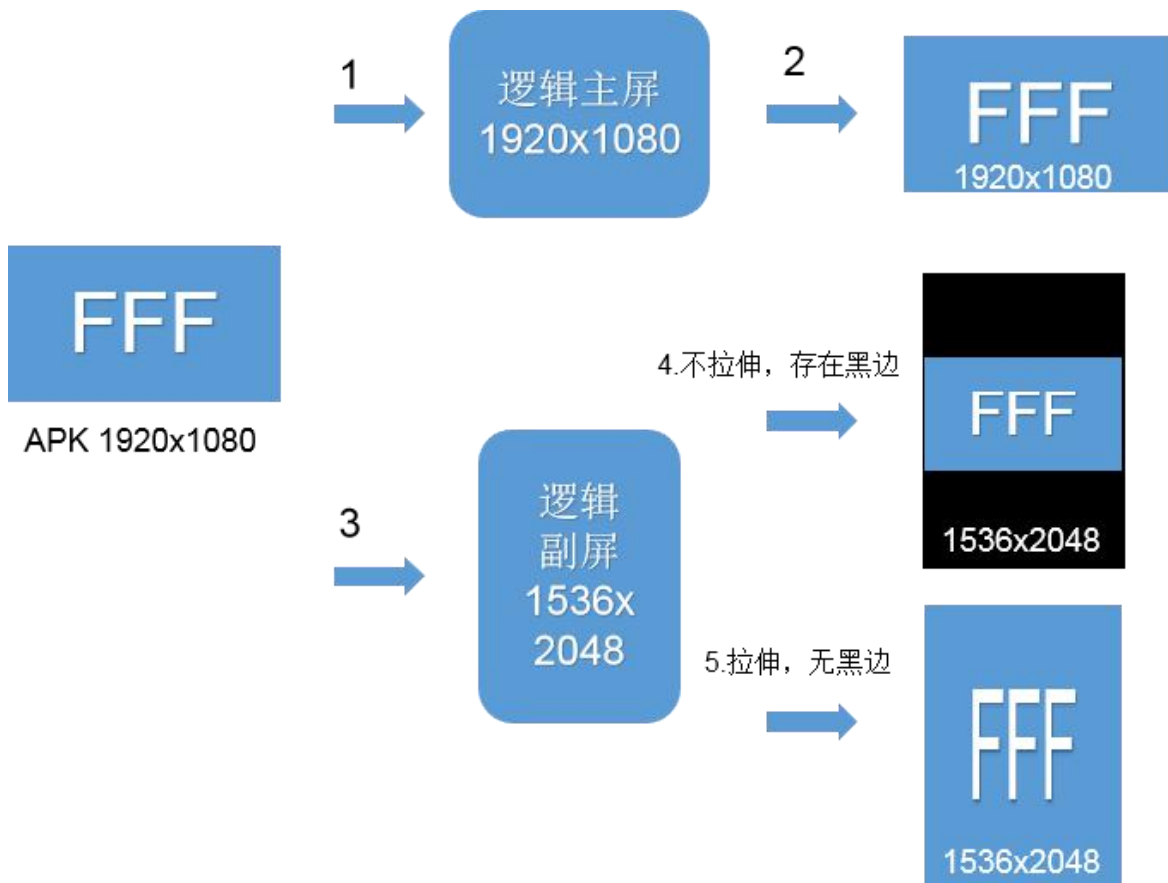
备注：双屏同显逻辑副屏就有可能出现黑边或拉伸的情况，具体的显示情况分为两种，如下：

- 主副屏宽高比相同：即 $w1/h1 = w2/h2$ ；



备注：若 **逻辑主副屏** 宽高比相同，则图像全屏显示，图像不会出现**拉伸**以及**黑边**。

- 主副屏宽高比不同：即 $w1/h1 \neq w2/h2$ ：



1：APK 获取 **逻辑主屏** 的属性信息（宽/高/刷新率），作为 APK 的渲染布局；

2：图像渲染布局与屏幕分辨率相同，则图像正常显示，不存在拉伸及黑边；

3：图像渲染布局为 1920x1080，屏幕分辨率为 1536x2048，宽高比不同，则只有两种显示模式 4 与 5；

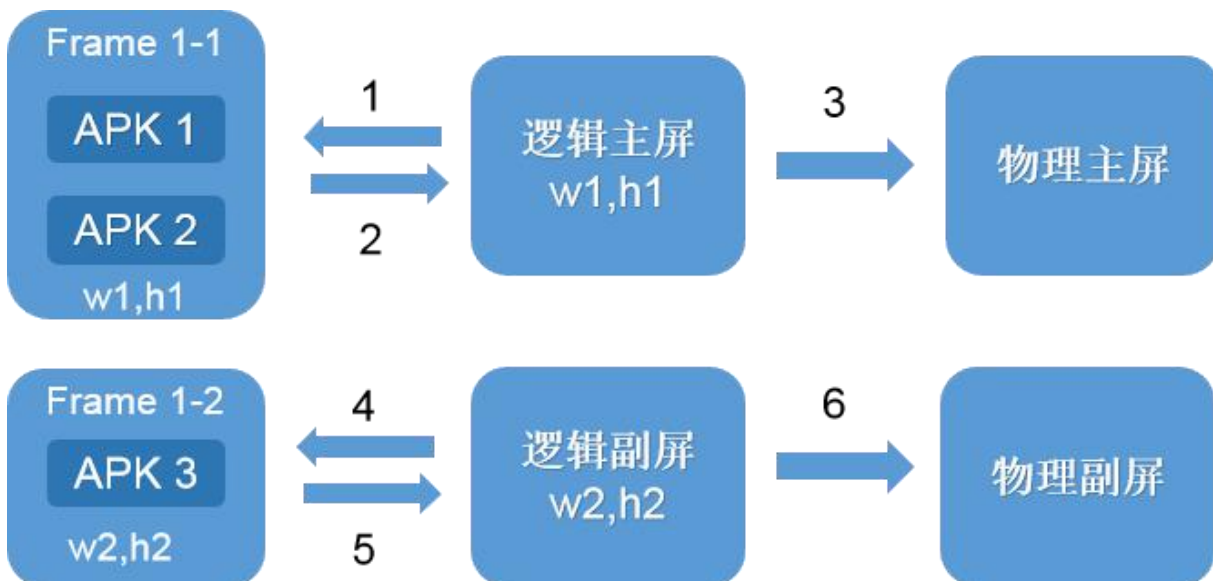
4: 图像居中显示, 图像不拉伸, 但是存在黑边;

5: 图像平铺显示, 图像拉伸, 但可以全屏显示;

备注: 在双屏同显的情况下: 若 **逻辑主副屏** 宽高比不相同, 则只能选择 4 与 5 显示模式的其中一种。

1.1.1.2 双屏异显

双屏异显, 实际上是 **APK** 根据需要显示的逻辑屏属性信息独立绘制, 并独立送显, 区别于双屏同显:



1: APK1, APK2 获取 **逻辑主屏** 的属性信息 (宽/高/刷新率), 作为 APK1,APK2 的渲染布局;

2: APK1, APK2 将显示请求提交给 **逻辑主屏**, 主屏根据层级结构进行混合;

3: **逻辑主屏** 将混合的结果送显至实际的物理主屏, 默认情况下, 逻辑主屏的属性信息与物理主屏一致;

4: APK3 获取 **逻辑副屏** 的属性信息 (宽/高/刷新率), 作为 APK3 的渲染布局;

5: APK3 将显示请求提交给 **逻辑副屏**, 副屏根据层级结构进行混合;

6: **逻辑副屏** 将混合的结果送显至实际的物理副屏, 默认情况下, 逻辑副屏的属性信息与物理副屏一致;

备注: 双屏异显由于 **APK** 送显内容为独立绘制, 故 **APK** 渲染布局始终与需要显示的物理屏幕属性一致, 故不存在拉伸与黑边的情况, 所有屏幕均正常显示, 并且显示的内容取决于各自的 **APK**。

1.1.2 横竖屏相关

区分物理横屏与物理竖屏，以及介绍竖屏横显，横屏竖显等概念：

- 物理横屏：



即实际屏幕宽度大于高度的屏幕，例如 1920x1080, 1280x720 等。

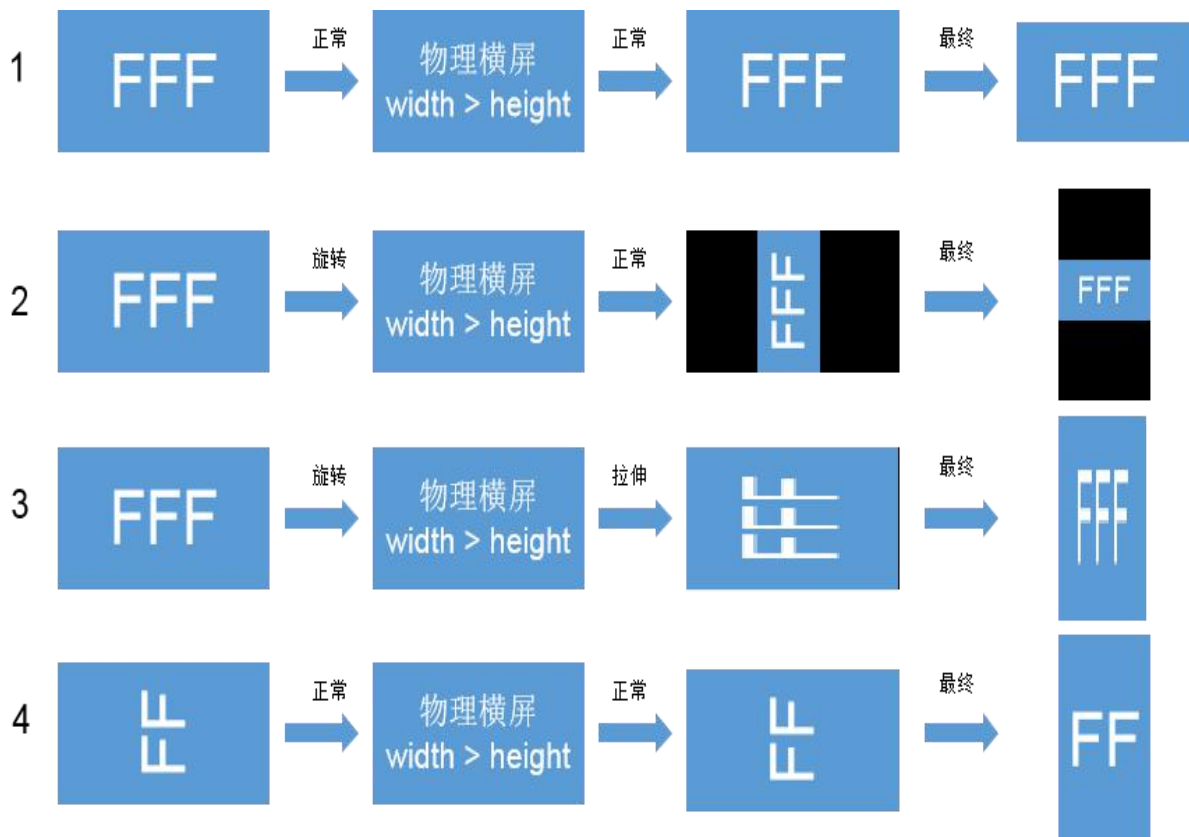
- 物理竖屏：



即实际屏幕高度大于宽度的屏幕，例如 1536x2048, 1080x1920 等。

- 横屏竖显与竖屏横显：

以横屏竖显举例：



- 1: 正常横屏显示;
- 2: 正常横屏+后端旋转显示;
- 3: 正常横屏+后端旋转+后端拉伸显示;
- 4: 正常横屏+APK 端旋转显示;

备注: 2/3/4 均是横屏竖显的 case, 并且均是可以实现的情况。

1.2 属性介绍

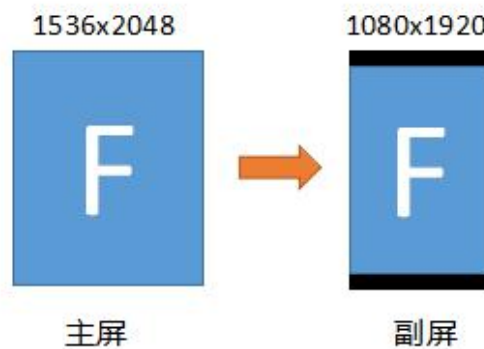
- **ro.sf.hwrotation = 0 \ 90 \ 180 \ 270:**

//为主屏的旋转方向, 默认为 0, 即主屏默认的显示方式, 例 1920x1080 为横屏。

- **persist.sys.rotation.efull = false/ true** //副屏是否全屏显示

false:

副屏图像参考主屏宽高比进行缩放, 若宽高比不一致, 则出现黑边。



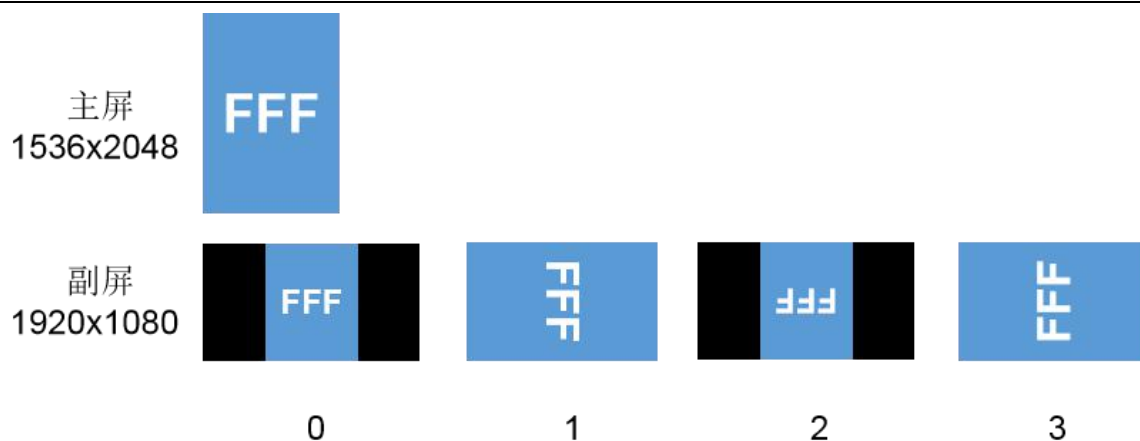
true:

副屏图像强制全屏显示, 凸显出现拉伸。



- **persist.sys.rotation.einit = 0 / 1 / 2 / 3**

//副屏输出旋转方向 (HDMI) 对应 0 / 90 / 180 / 270

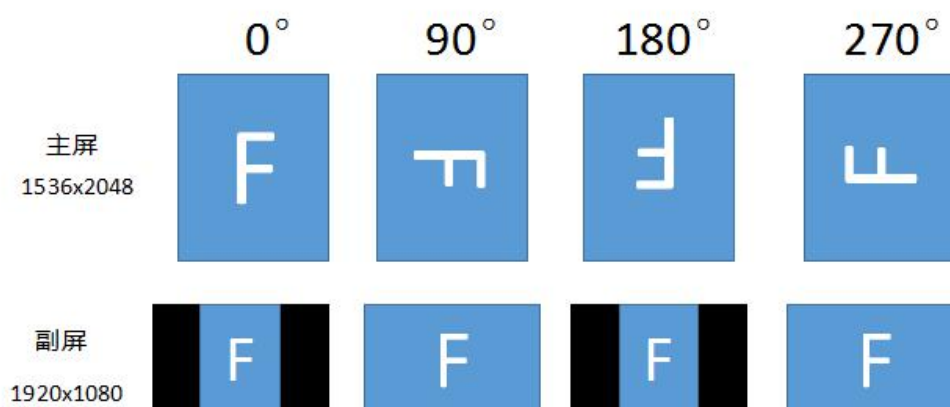


● **ro.sys.rotation.sensor = false / true**

//副屏（HDMI）也随着 g-sensor 旋转（只关注方向，黑边不关注）。

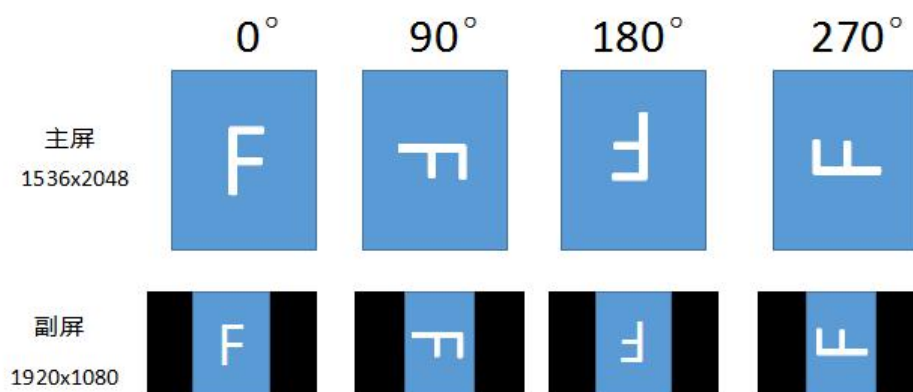
false:

副屏方向不随主屏 g-sensor 旋转，即主屏旋转到任意方向，副屏始终保持一个方向。



true:

副屏方向随着主屏 g-sensor 旋转，并与主屏旋转方向一致。



2 场景调试说明

2.1 双屏同显

2.1.1 主副屏横竖屏属性一致，且宽高比相同

主副屏均为物理横屏或物理竖屏，并且宽高比相同（例如 $1920/1080=1280/720=1.7778$ ），这里以物理横屏为例，进行说明：













主屏为 1920x1080 物理横屏，副屏为 1280x720 物理横屏：

- **ro.sf.hwrotation = 0 \ 90 \ 180 \ 270 :**

persist.sys.rotation.efull = false / true :

备注：主屏方向根据 **ro.sf.hwrotation** 值改变，副屏显示方向始终不变，但显示布局根据主屏而改变，当主屏的显示布局为竖屏时，副屏就居中显示，并存在黑边，若将 **persist.sys.rotation.efull** 设置为 true，则会拉伸图像，铺满全屏。

persist.sys.rotation.einit = 0
ro.sys.rotation.sensor = false
ro.sf.hwrotation:

	0	90	180	270
主屏 1920x1080				
副屏 1280x720 persist.sys.rotation.efull = false				
副屏 1280x720 persist.sys.rotation.efull = true				

- **persist.sys.rotation.einit = 0 / 1 / 2 / 3:**

persist.sys.rotation.efull = false / true:

ro.sf.hwrotation = 0
ro.sys.rotation.sensor = false

主屏
1920x1080



persist.sys.rotation.einit: 0

副屏
1280x720



1



2



3



persist.sys.rotation.efull = false

副屏
1280x720



persist.sys.rotation.efull = true

备注: **persist.sys.rotation.einit** 可调整副屏初始化角度, **persist.sys.rotation.efull**

调整是否强制全屏显示, 若宽高比不同则拉伸至全屏。

● ro.sys.rotation.sensor = false / true:

ro.sf.hwrotation = 0
persist.sys.rotation.einit = 0
ro.sys.rotation.sensor = true

sensor rotation:

0

90

180

270

主屏
1920x1080



副屏
1280x720



persist.sys.rotation.efull = false

副屏
1280x720



persist.sys.rotation.efull = true

备注: **ro.sys.rotation.sensor** 表示副屏是否会随着主屏的 sensor 角度做对应旋转。

2.1.2 主副屏横竖屏属性一致, 且宽高比不同

主副屏均为物理横屏或物理竖屏, 但宽高比不同 (例如 $1920 \times 1080 = 1.7778$, $1360 \times 768 = 1.7708$), 这里以物理横屏为例, 进行说明:

主屏为 1920×1080 物理横屏, 副屏为 1360×768 物理横屏:

● ro.sf.hwrotation = 0 / 90 / 180 / 270:

persist.sys.rotation.efull = false / true:

persist.sys.rotation.einit = 0
ro.sys.rotation.sensor = false
ro.sf.hwrotation: 0 90 180 270

主屏
1920x1080



副屏
1360x768



persist.sys.rotation.efull = false

副屏
1360x768



persist.sys.rotation.efull = true

备注：当主副屏宽高比不同时，**persist.sys.rotation.efull** 为 false，则副屏会出现黑边，所以这种情况，建议将 **persist.sys.rotation.efull** 设置为 true，些许的拉伸不会影响显示效果。

● **persist.sys.rotation.einit = 0 / 1 / 2 / 3:**

persist.sys.rotation.efull = false / true:

ro.sf.hwrotation = 0
ro.sys.rotation.sensor = false

主屏
1920x1080



persist.sys.rotation.einit: 0

1

2

3

副屏
1360x768



persist.sys.rotation.efull = false

副屏
1360x768



persist.sys.rotation.efull = true

备注：当主副屏宽高比不同时，**persist.sys.rotation.efull** 为 false，则副屏会出现黑边。

● **ro.sys.rotation.sensor = false / true:**

persist.sys.rotation.einit = 0
 ro.sys.rotation.sensor = false
 ro.sf.hwrotation: 0 90 180 270

主屏
1920x1080



副屏
1360x768



persist.sys.rotation.einit = false

副屏
1360x768



persist.sys.rotation.einit = true

备注：当主副屏宽高比不同时，**persist.sys.rotation.einit** 为 false，则副屏会出现黑边。

2.1.3 主副屏横竖屏属性不一致

主屏为物理横屏副屏为物理竖屏，或者主屏为物理竖屏副屏为物理横屏，这里以主屏为 1536x2048，副屏为 1920x1080 的分辨率举例说明。

主屏为 1536x2048 物理竖屏，副屏为 1920x1080 物理横屏：

● ro.sf.hwrotation = 0 / 90 / 180 / 270:

persist.sys.rotation.einit = false / true:

persist.sys.rotation.einit = 0
 ro.sys.rotation.sensor = false
 ro.sf.hwrotation: 0 90 180 270

主屏
1536x2048



副屏
1920x1080



persist.sys.rotation.einit = false

副屏
1920x1080



persist.sys.rotation.einit = true

● persist.sys.rotation.einit = 0 / 1 / 2 / 3:

persist.sys.rotation.efull = false / true:`ro.sf.hwrotation = 0``ro.sys.rotation.sensor = false`主屏
1536x2048`persist.sys.rotation.einit: 0`副屏
1920x1080`persist.sys.rotation.efull = false`

1



2



3

副屏
1920x1080`persist.sys.rotation.efull = true`

备注：若主副屏横竖屏属性不一致，则建议将 **persist.sys.rotation.einit** 设置为 1 或 3，并且 **persist.sys.rotation.efull** 设置为 true，能保证副屏能够全屏显示内容，即副屏采用横屏竖显方式。

● ro.sys.rotation.sensor:`ro.sf.hwrotation = 0``persist.sys.rotation.einit = 0``ro.sys.rotation.sensor = true`

sensor rotation:

0

90

180

270

主屏
1536x2048副屏
1920x1080`persist.sys.rotation.efull = false`副屏
1920x1080`persist.sys.rotation.efull = true`

备注：由于主副屏横竖屏属性不一致，故在 **ro.sys.rotation.sensor** 设置为 true 的时候，

副屏在 sensor 各角度均有黑边,故这种情况建议将 **persist.sys.rotation.einit** 设置为 1 或 3, 将副屏显示布局调整为与主屏一致。

- **persist.sys.rotation.einit = 1** 的显示情况如下:

ro.sf.hwrotation = 0

persist.sys.rotation.einit = 1

ro.sys.rotation.sensor = true

sensor rotation:

0

90

180

270

主屏
1536x2048



副屏
1920x1080



persist.sys.rotation.efull = false

副屏
1920x1080



persist.sys.rotation.efull = true

备注: **persist.sys.rotation.einit = 1** 提前将副屏显示内容旋转 90 度, 对比

persist.sys.rotation.einit = 0, sensor 各角度副屏均在原来的基础上旋转了 90 度, 显示效果如上图。

2.2 双屏异显

双屏异显区别于双屏同显, 显示内容为独立绘制, 默认副屏 window 参照的是副屏宽高进行绘制, 所以只要配置正确基本不存在黑边拉伸问题, 这边仅介绍对应属性的设置效果:

- **persist.sys.rotation.einit**: 控制副屏的显示方向 (值分别 0,1,2,3)。

ro.sf.hwrotation = 0

ro.sys.rotation.sensor = false

persist.sys.rotation.efull = false / true

主屏
1536x2048



persist.sys.rotation.einit: 0

1

2

3

副屏
1920x1080



`ro.sys.rotation.sensor`: 异显时不支持副屏随着主屏旋转, 该属性在异显不起作用。

`persist.sys.rotation.efull`: 异显时默认各屏独立参照各自屏幕宽高绘制内容, 不存在拉伸黑边, 无需设置该属性。

`persist.sys.draw.einfo`: 默认异显时各屏上的 `window` 使用所在屏的宽高来绘制, 如果有特殊需求, 需要副屏 `window` 依照主屏宽高来绘制, 可以设置该属性为 `0`, 一般情况下不需要设置。

2.3 补丁介绍

相关代码已提交服务器并对外更新, 客户可以通过服务器同步更新代码, 若需要单独补丁, 请联系 FAE。

补丁目录:

—— patch

└── Android 7.1

└── RK3399

└── frameworks

└── base

└── 0001-DualScreen-1.Add-property-persist.sys.rotation.efull.patch

└── native

└── 0001-DualScreen.patch

└── packages

└── apps

└── Settings

└── 0001-DualScreen-Improve-the-..l-rotation-fullscreen-.patch

└── 0002-DualScreen-improve-be373-remove-resource-zh-rCN.patch

└── readme.txt

具体操作步骤请查看 `readme.txt`。