Rockchip Android Automotive开发指南

文件标识：RK-KF-YF-792

发布版本：V1.0.0

日期：2024-06-13

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

Android Automotive 是一个基础 Android 平台，可运行预装的 IVI 系统 Android 应用程序以及可选的第二方和第三方 Android 应用程序。

Android Automotive 为汽车信息娱乐系统和音响主机提供开放性、定制性和可扩展性。开放性通过在免费开源代码库中提供基本的汽车信息娱乐功能来提高效率。定制使实施者能够根据他们认为合适的方式来区分产品。规模化是通过 Android 的通用框架、语言和 API 实现的，所有这些都可以重用全球数十万 Android 开发人员的开发专业知识和已完成的软件。

了解 Android Automotive 与整个 Android 生态系统的关系非常重要：

- Android Automotive 是 Android设备。 Android Automotive 不是 Android 的分叉或并行开发。它与手机、平板电脑等设备上的 Android 具有相同的代码库和相同的存储库。它构建在经过 10 多年开发的强大平台和功能集之上，使其能够利用现有的安全模型、兼容性程序、开发人员工具和基础设施，同时继续高度可定制和可移植，完全免费和开源。
- Android Automotive 扩展了 Android 。在将 Android 构建为功能齐全的信息娱乐平台的过程中，我们添加了对汽车特定要求、功能和技术的支持。 Android Automotive 将成为全栈交钥匙汽车信息娱乐平台，就像 Android 当今的移动平台一样。

十多年来，运营商、OEM 和开发人员一直在使用 Android 来构建精美的设备、应用程序和体验。Android Automotive 现在将 Android 的强大功能带入汽车，汽车制造商可以创建专为数字时代设计的强大信息娱乐系统。

本文主要介绍关于Automotive的相关特性及配置说明。

产品版本

芯片名称	内核版本	**Android**版本
RK3588	Kernel 5.10	Android 12
RK3588/RK3576	Kernel 6.1	Android 14

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

版本号	作者	修改日期	修改说明
V1.0.0	翁韬	2024-06-13	初始版本

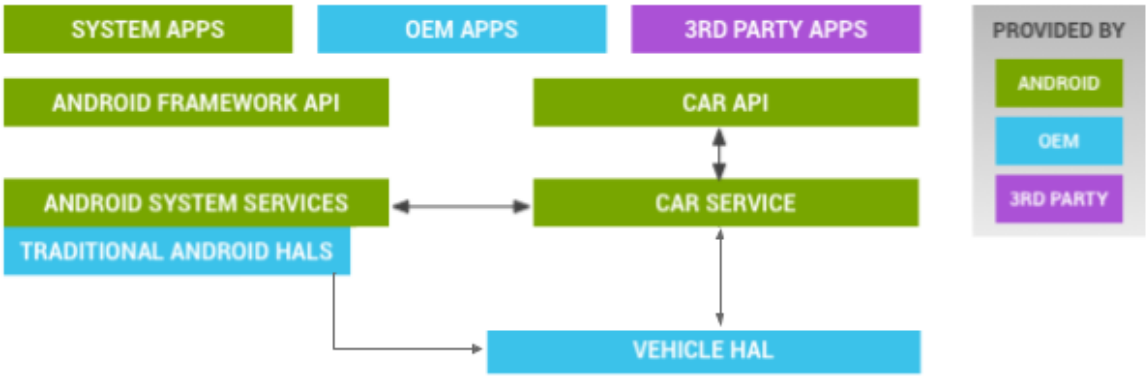
目录

1. 开发指南

1.1 AAOS系统

平板SDK采用的是AOSP，而车载SDK采用AAOS系统，车载SDK适配并完善了Android14 上的AAOS功能，即采用了谷歌车载UI，并支持多用户机制，将驾驶员和乘客进行权限区分，支持多屏幕/触摸/背光/、多音区、仪表盘（有需要补丁的暂时由REDMINE提供）等功能，从而支持单驾驶员和多乘客在使用时相互不影响。AAOS上层接口采用google标准接口，相关功能介绍可以参考谷歌官网：https://source.android.com/docs/devices/automotive

车载 HAL 与 Android Automotive 架构见下图：



- Car API 内有包含 CarSensorManager 在内的 API，如需详细了解受支持的 API，请参阅 packages/services/Car/car-lib。
- CarService 位于 packages/services/Car/
- Car相关APK 位于packages/apps/Car/
- Vehicle HAL 用于定义 OEM 可以实现的车辆属性的接口，位于 hardware/interfaces/automotive/vehicle/

1.2 多用户

AAOS SDK默认配置支持了多用户，包括一个Driver、最多五个Passenger。其中Driver以及Passenger可以支持独立的屏幕、触屏，可以独立操控各自的屏幕来打开不同的apk（也可以打开相同的apk）。

1. 无头用户模式
   Android 系统需要配置无头用户模式，可参考AOSP文档：https://source.android.google.cn/docs/devices/admin/multi-user?hl=zh-cn#android-automotive-multi-user

   配置如下：
   `device/rockchip/common/car/packages_car.mk`

   ```
   PRODUCT_SYSTEM_DEFAULT_PROPERTIES += \
       ro.fw.mu.headless_system_user?=true
   ```

2. 多用户使能

CarService使能多用户，需配置以下参数：

Android 14路径：

`device/rockchip/common/car/overlay/packages/services/Car/service/res/values/config.xml`

Android 12路径：

`device/rockchip/rk3588/rk3588m_car/overlay/packages/services/Car/service/res/values/config.xml`

```
<bool name="enablePassengerSupport">true</bool>
<bool name="enableProfileUserAssignmentForMultiDisplay"
translatable="false">true</bool>
```

Android 14路径：

`device/rockchip/car/overlay/frameworks/base/core/res/res/values/config.xml`

Android 12路径：

`device/rockchip/rk3588/rk3588m_car/overlay/frameworks/base/core/res/res/values/config.xml`

```
<integer name="config_multiuserMaximumUsers">9</integer>
<integer name="config_multiuserMaxRunningUsers">9</integer>
```

3. Occupant配置

配置Occupant与Display的关系。

Android 14路径：

`device/rockchip/common/car/overlay/packages/services/Car/service/res/values/config.xml`

Android 12路径：

`device/rockchip/rk3588/rk3588m_car/overlay/packages/services/Car/service/res/values/config.xml`

```
<string-array translatable="false" name="config_occupant_zones">

 <item>occupantZoneId=0,occupantType=DRIVER,seatRow=1,seatSide=driver</item>
    <!--
<item>occupantZoneId=1,occupantType=DRIVER,seatRow=1,seatSide=driver</item> -
->
  <item>occupantZoneId=1,occupantType=FRONT_PASSENGER,seatRow=1,seatSide=oppo
siteDriver</item>

 <item>occupantZoneId=2,occupantType=REAR_PASSENGER,seatRow=2,seatSide=left</
item>

 <item>occupantZoneId=3,occupantType=REAR_PASSENGER,seatRow=2,seatSide=right<
/item>

 <item>occupantZoneId=4,occupantType=REAR_PASSENGER,seatRow=3,seatSide=left</
item>

 <item>occupantZoneId=5,occupantType=REAR_PASSENGER,seatRow=3,seatSide=right<
/item>
</string-array>
 <string-array translatable="false" name="config_occupant_display_mapping">
```

```
    <item>displayPort=0,displayType=MAIN,occupantZoneId=0</item>
    <item>displayPort=1,displayType=MAIN,occupantZoneId=1</item>
    <!--
<item>displayPort=1,displayType=INSTRUMENT_CLUSTER,occupantZoneId=0</item> --
>
    <item>displayPort=2,displayType=MAIN,occupantZoneId=2</item>
    <item>displayPort=3,displayType=MAIN,occupantZoneId=3</item>
    <item>displayPort=4,displayType=MAIN,occupantZoneId=4</item>
    <item>displayPort=5,displayType=MAIN,occupantZoneId=5</item>
</string-array>
```

- config_occupant_zones：汽车中可用的所有乘员(=驾驶员+乘客)区域。
- occupantZoneId：唯一的无符号整数id代表每个乘客区。每个zone应该有不同的id。
- occupantType：Display的占用者类型。使用CarOccupantZoneManager.OCCUPANT_TYPE_*。如OCCUPANT_TYPE_DRIVER（司机）、OCCUPANT_TYPE_FRONT_PASSENGER（前排乘客）、OCCUPANT_TYPE_REAR_PASSENGER（后排乘客）等。
- seatRow：表示座位所在的行数。
- seatSide：left/center/rgith/表示所在行数的位置。或者可以使用司机/中间/对侧司机来同时处理右舵驾驶和左舵驾驶。如果没有指定车辆是右舵驾驶还是左舵驾驶，则默认假设为左舵驾驶，此时司机侧为左侧。
- config_occupant_display_mapping：指定系统中Display的配置，包括其用途/类型和分配的使用者。如果在此处分配了DEFAULT_DISPLAY，应始终分配给司机区域。
- displayPort：Display的唯一Port ID，可通过`dumpsys display`查看对应的Port id。
- displayUniqueId(Optional)：Display的唯一id，可通过`dumpsys display`查看对应的Unique id。VirtualDisplay的唯一ID将采用‘virtual:<package>:<ID>’的形式。在配置Cluster 时需要使用。
- displayType：Display的显示类型。使用CarOccupantZoneManager.DISPLAY_TYPE_*，如DISPLAY_TYPE_MAIN、DISPLAY_TYPE_INSTRUMENT_CLUSTER等。
- occupantZoneId：指向config_occupant_zones中的occupantZoneId，建立Display和Occupant的匹配关系。

Debug方法：

- `dumpsys carservice`

  可通过OccupantZoneService中的dump方法查看Occupant的配置，包括Occupant和Display的关系，Occupant和User的关系，当前可用的Occupant配置，AudioZone配置等。

- `pm list users`

  查看当前UserInfo的个数及状态。

- `dumpsys user`
  参看当前系统的User信息，如UserName、状态、权限、创建时间、启动时间、限制情况、Display匹配等。

4. HWC的多屏异显

   RK3588拥有4个vp，默认支持4屏显示。RK3576拥有3个vp，默认支持3屏显示。RK3588的六屏显示是通过HWC的多屏拼接模式实现，详细内容可参阅文档
   RKDocs\common\display\Rockchip_Developer_Guide_Android_Car_MultiDisplay_CN.pdf

5. 多屏背光调节

   display_settings.xml对屏幕ID及背光路径进行了定义，默认值与EVB配套，产品上需要进行相应修改。

   Default File: `device/rockchip/common/display_settings.xml`

   RK3588 Android 12: `device/rockchip/rk3588/rk3588m_car/display_settings.xml`

RK3588 Android 14: `device/rockchip/rk3588/rk3588m_u/displays/display_settings.xml`

RK3576 Android 14: `device/rockchip/rk3576/rk3576m_u/displays/display_settings.xml`

以下通过RK3588 Android 14的display_settings.xml进行介绍。

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<display-settings>
    <config identifier="1" />
    <display
        name="port:0"
        backlightPath="/sys/class/backlight/backlight/brightness" />
    <display
        name="port:1"
        shouldShowSystemDecors="true"
        shouldShowIme="true"
        forcedDensity="160"
        backlightPath="/sys/class/backlight/edp2lvds_backlight0/brightness"
/>
    <display
        name="port:2"
        shouldShowSystemDecors="true"
        shouldShowIme="true"
        forcedDensity="160"
        backlightPath="/sys/class/backlight/dsi2lvds_backlight1/brightness"
/>
    <display
        name="port:3"
        shouldShowSystemDecors="true"
        shouldShowIme="true"
        forcedDensity="160"
        backlightPath="/sys/class/backlight/dp2lvds_backlight0/brightness" />
    <display
        name="port:4"
        shouldShowSystemDecors="true"
        shouldShowIme="true"
        forcedDensity="160"
        backlightPath="/sys/class/backlight/edp2lvds_backligh1/brightness" />
    <display
        name="port:5"
        shouldShowSystemDecors="true"
        shouldShowIme="true"
        forcedDensity="160"
        backlightPath="/sys/class/backlight/dp2lvds_backlight1/brightness" />
</display-settings>
```

- config identifier：系统解析此文件时会根据该配置判断name匹配的是Port id（value=1）或Unique id（value=0）。
- name：Display的Port id（port:*）或Unique id（local:*），可通过`dumpsys display`或`dumpsys SurfaceFlinger`获取对应的值。port:&id或local:&id中的id需要和config_occupant_display_mapping中的displayPort或displayUniqueId相匹配，否则会导致CarOccupantZoneService无法找到对应的Display。
- backlightPath：Display对应的背光节点，需要根据实际屏幕和name做好匹配关系，否则会导致背光调节错误。如DSI的背光调节节点为/sys/class/backlight/backlight/brightness，背光调节节点请根据dts进行确认。

背光调节路径需要提供权限给到System用户，可在init.rc中对背光调节节点赋予对应权限，参考如下：

```
on boot
    # backlight
    chown system system /sys/class/backlight/backlight/brightness
    chown system system
/sys/class/backlight/dp2lvds_backlight0/brightness
    chown system system
/sys/class/backlight/dp2lvds_backlight1/brightness
    chown system system
/sys/class/backlight/dsi2lvds_backlight1/brightness
    chown system system
/sys/class/backlight/edp2lvds_backlight0/brightness
    chown system system
/sys/class/backlight/edp2lvds_backlight1/brightness
```

该文件其他参数为Google定义，可查看源码`frameworks/base/services/core/java/com/android/server/wm/DisplayWindowSettingsProvider.java`或参考AOSP相关文档https://source.android.google.cn/docs/core/display/multi_display/recommended-practices?hl=zh-cn#windowing。

背光调节Api：

- 系统级

  以下方法为系统Api，需要系统权限，修改后可同步至ContentProvider，开启或关闭多用户对该Api无差别。

  ```
  DisplayManager mDisplayManager =
  context.getSystemService(DisplayManager.class);
  //参考代码
  packages/apps/Car/Settings/src/com/android/car/settings/qc/BrightnessSlider.java
  //设置亮度
  float linearFloat = BrightnessSynchronizer.brightnessIntToFloat(linear);
  mDisplayManager.setBrightness(displayId, linearFloat);
  //获取亮度
  float linearFloat = mDisplayManager.getBrightness(displayId);
  int linear = BrightnessSynchronizer.brightnessFloatToInt(linearFloat);
  ```

- 应用级

  获取/设置当前Activity亮度，仅当前Activity有效，退出即回退系统原值，开启或关闭多用户对该Api无差别。

  Tips：需要注意的是**Activity**对应的**DisplayId**需要和实际相同，避免使用**Context**，**Context**容易出现**Display**和实际不符。

  ```
  //获取当前Activity亮度(0-1)
  float brightness = getWindow().getAttributes().screenBrightness;
  //设置当前Activity亮度(0-1)
  Window window = getWindow();
  WindowManager.LayoutParams lp = window.getAttributes();
  lp.screenBrightness = (float)brightness;
  window.setAttributes(lp);
  ```

获取/设置当前系统的亮度，**需要系统权限，未使用AAOS多用户的情况下，无法调节非主屏亮度！**

```
//获取当前系统亮度（0-255）
Settings.System.getIntForUser(mContext.getContentResolver(),
Settings.System.SCREEN_BRIGHTNESS, defValue, userId);
//设置当前系统亮度
//设置系统背光模式为手动调节
Settings.System.putIntForUser(mContext.getContentResolver(),
Settings.System.SCREEN_BRIGHTNESS_MODE,
Settings.System.SCREEN_BRIGHTNESS_MODE_MANUAL, userId);
//设置系统背光亮度
Settings.System.putIntForUser(mContext.getContentResolver(),
Settings.System.SCREEN_BRIGHTNESS, birghtessValue, userId);
```

6. 多屏输入

该功能可指定哪些输入设备属于哪些屏幕，此关联由端口号完成，其中端口是指与屏幕连接的物理端口。

该关联在板端的`/vendor/etc/input-port-associations.xml`中配置，以下文件可作参考

RK3588 Android 12: `device/rockchip/rk3588/rk3588m_car/input-port-associations.xml`

RK3588 Android 14: `device/rockchip/rk3588/rk3588m_u/displays/input-port-associations.xml`
RK3576 Android 14: `device/rockchip/rk3576/rk3576m_u/displays/input-port-associations.xml`

以RK3588 Android 14为例：

```
<ports>
    <port display="0" input="himax-2-0048/input0" />  #DSI0 720P
    <port display="1" input="ili210x-5-0041/input0" /> #EDP0 1080P
    <port display="2" input="himax-6-0048/input0" /> #DSI1 720P
    <port display="3" input="himax-4-0048/input0" /> #DP0 720P
    <port display="241" input="ili210x-7-0041/input0" /> #EDP1 1080P
    <port display="243" input="himax-8-0048/input0" /> #DP1 720P
</ports>
```

- display：指定与屏幕连接的端口，可通过`dumpsys display`确定需要分配的端口。
- input：指定与输入设备连接的端口。可通过`dumpsys input`查看设备的`Location`属性。

7. 多音区

SDK默认发布时仅配置四路音区，仅能支持四路已配置音区的Occupant独立播放音视频，如需要更多路Occupant都能够独立播放音视频，需要在以上文件中添加更多路音区的配置。根据实际情况配置音频输出设备，并配置每个音区与Occupant的对应关系。

RK3588 Android 12:

```
device/rockchip/rk3588/rk3588m_car/audio_policy_configuration.xml
device/rockchip/rk3588/rk3588m_car/car_audio_configuration.xml
```

RK3588 Android 14:

```
device/rockchip/rk3588/rk3588m_u/audio/audio_policy_configuration.xml
device/rockchip/rk3588/rk3588m_u/audio/car_audio_configuration.xml
```

RK3576 Android 14:

```
device/rockchip/rk3576/rk3576m_u/audio/audio_policy_configuration.xml
device/rockchip/rk3576/rk3576m_u/audio/car_audio_configuration.xml
```

详细介绍可参考文档Rockchip_RK3588M_Developer_Guide_Android12_Audio_CN.md。

8. 热插拔功能
   Rockchip AAOS可支持Display设备的热插拔功能。

   同时在多用户场景下，Occupant的显示接口（如HDMI、eDP、DP、MIPI等）拔出后再接入同一类型的显示设备，会保持之前的user以及配置，同时恢之前已经打开的应用堆栈，即可恢复拔插之前的状态。

9. 存储

   多用户下的存储挂载情况与非多用户的存储挂载情况相差较大，且由于FUSE的原因，多个目录无法通过shell进行访问。同时AAOS默认开启了FUSE透传模式，挂载目录已变更为`/mnt/pass_through/`，减少FUSE多次在用户空间和内核空间切换的性能开销。

   多用户场景下，每个user的共享存储空间路径通过`Environment.getExternalStorageDirectory()`接口获取路径，获取到的值为`/storage/emulated/{userid}`。注意：这个路径在shell下是不能访问的，shell下可访问的对应路径为：`/mnt/user/userid/emulated/{userid}`。

   如需了解FUSE的相关资料，可以参考AOSP文档https://source.android.google.cn/docs/core/storage?hl=zh-cn以及Kernel文档https://www.kernel.org/doc/html/latest/filesystems/fuse.html。

   Rockchip已完成了多用户下对于U盘等外置存储的支持，对于所有的User来说，都可以通过MediaProvider API对U盘等外置存储进行访问，挂载路径为`/mnt/pass_through/{userId}/{UUID}{userId}`，应用访问路径为`/storage/{UUID}{userId}`。

10. User权限控制
    路径：
    `packages/services/Car/car_product/overlay/frameworks/base/core/res/res/xml/config_user_types.xml`

    此文件可以配置AAOS相关User的权限，如拨打电话、恢复出厂、安装应用等，详细权限可参考AOSP文档：https://source.android.google.cn/docs/devices/admin/multi-user?hl=zh-cn#user_types

11. 预装应用限制

    由于并非所有系统软件包都适用于所有类型的 Android 用户，因此您可以使用许可名单指定应为每种类型的用户预安装哪些系统软件包。避免预安装不必要的系统软件包，这样就可以优化用户创建次数、启动次数和内存用量。

    AAOS的预安装系统软件用户类型配置文件在packages/services/Car/car_product/build/preinstalled-packages-product-car-base.xml，详细介绍请参考AOSP文档：https://source.android.google.cn/docs/core/permissions/preinstalled-packages?hl=zh-cn#base

12. 汽车用户体验限制

    SDK默认汽车用户体验限制：驾驶位在汽车行驶中默认开启显示，在驻车以及空闲状态下默认不限制；其他座位默认都不做限制。
    客户需要根据实际需求做相应调整，默认配置文件为：
    device/rockchip/common/car/overlay/packages/services/Car/service/res/xml/car_ux_restrictions_map.xml

    AAOS文档：https://source.android.google.cn/docs/automotive/driver_distraction/guidelines?hl=zh_cn

13. 关闭多用户

    - 关闭无头用户模式

`device/rockchip/common/car/packages_car.mk`

```
PRODUCT_SYSTEM_DEFAULT_PROPERTIES += \
       ro.fw.mu.headless_system_user?=false
```

- 关闭CarService多用户
  Android 14路径：
  `device/rockchip/common/car/overlay/packages/services/Car/service/res/values/config.xml`

  Android 12路径：
  `device/rockchip/rk3588/rk3588m_car/overlay/packages/services/Car/service/res/values/config.xml`

  ```
  <bool name="enablePassengerSupport">false</bool>
  <bool name="enableProfileUserAssignmentForMultiDisplay"
  translatable="false">false</bool>
  ```

关闭多用户后，多屏幕默认配置成异显，如需修改为同显，请参考以下修改：

- Android 12：

  `packages/apps/Car/Launcher/AndroidManifest.xml`

  ```
  diff --git a/AndroidManifest.xml b/AndroidManifest.xml
  index c7078b89..cf3d8235 100644
  --- a/AndroidManifest.xml
  +++ b/AndroidManifest.xml
  @@ -126,11 +126,11 @@
               android:exported="true"
               android:enabled="true"

  android:configChanges="orientation|screenSize|smallestScreenSize|scr
  eenLayout|colorMode|density">
  -            <intent-filter>
  +        <!--<intent-filter>
                   <action android:name="android.intent.action.MAIN"
  />
                   <category
  android:name="android.intent.category.SECONDARY_HOME" />
                   <category
  android:name="android.intent.category.DEFAULT" />
  -            </intent-filter>
  +        </intent-filter>-->
           </activity>
           <service
  android:name=".homescreen.audio.telecom.InCallServiceImpl"

  android:permission="android.permission.BIND_INCALL_SERVICE"
  ```

  `packages/apps/Car/SystemUI/res/values/config.xml`

```
diff --git a/res/values/config.xml b/res/values/config.xml
index 6b2ffe1f..f8c44cfa 100644
--- a/res/values/config.xml
+++ b/res/values/config.xml
@@ -31,7 +31,7 @@
     <bool name="config_enableLeftSystemBar">false</bool>
     <bool name="config_enableRightSystemBar">false</bool>
     <bool name="config_enableBottomSystemBar">true</bool>
-    <bool name="config_enableBottom2SystemBar">true</bool>
+    <bool name="config_enableBottom2SystemBar">false</bool>

     <!-- Configure the type of each system bar. Each system bar
must have a unique type. -->
     <!--    STATUS_BAR = 0-->
```

- Android 14：

`packages/services/Car/car_product/overlay/frameworks/base/core/res/res/values/config.xml`

```
diff --git
a/car_product/overlay/frameworks/base/core/res/res/values/config.xml
b/car_product/overlay/frameworks/base/core/res/res/values/config.xml
index 59053739cc..c38c29eecb 100644
---
a/car_product/overlay/frameworks/base/core/res/res/values/config.xml
+++
b/car_product/overlay/frameworks/base/core/res/res/values/config.xml
@@ -23,7 +23,7 @@
     <!-- The dreams feature (screensavers) is not supported in
android auto -->
     <bool name="config_dreamsSupported">false</bool>
     <!-- Disable local-display-mirror-content -->
-    <bool name="config_localDisplaysMirrorContent">false</bool>
+    <bool name="config_localDisplaysMirrorContent">true</bool>
     <!-- Enable multi-user. -->
     <bool name="config_enableMultiUserUI">true</bool>
     <!--  Maximum number of supported users -->
```

`device/rockchip/common/car/packages_car.mk`

```
diff --git a/car/packages_car.mk b/car/packages_car.mk
index 6c820a44..afb91b85 100644
--- a/car/packages_car.mk
+++ b/car/packages_car.mk
@@ -164,7 +165,7 @@ PRODUCT_PACKAGES += \
     libcarservicehelperjni \
     com.android.car.procfsinspector \
     com.android.permission \
-    MultiDisplaySecondaryHomeTestLauncher \
+    #MultiDisplaySecondaryHomeTestLauncher \
```

1.3 Launcher实现

AAOS的Launcher与AOSP不同，默认需要多屏异显，配置如下：

Android 14：
`packages/services/Car/car_product/overlay/frameworks/base/core/res/res/values/config.xml`

```
<bool name="config_localDisplaysMirrorContent">false</bool>
```

除了DefaultDisplay外，其余Display的Launcher都是通过SecondaryLauncher实现。在Android 12上，SecondaryLauncher实现在CarLauncher中。在Android 14上，SecondaryLauncher则是通过MultiDisplaySecondaryHomeTestLauncher实现。

两者的差异在于Android 12上的SecondaryLauncher都是User 10的，通过CarOccupantManager获取Display所对应的User，通过UserId处理对应Display的相关操作。在Android 14上，SecondaryLauncher已是CarOccupantManager所对应Display的User，直接获取当前进程的User即可操作。

AAOS Launcher最大的差异在于所有的操作都需要考虑当前的User，目前AOSP及AAOS AP基本已支持传递UserId参数，可通过UserId对操作的用户进行区分。

代码可参考：

Android 14：`packages/services/Car/tests/MultiDisplaySecondaryHomeTestLauncher`
Android 12：`packages/apps/Car/Launcher`

Tips：**需要等待User UnLock后才可以进行对应用户的相关操作，否则会导致数据出现异常。**

1.4 SystemUI

SystemUI在Android 14和Android 12上相差较多，以下分析两者差异。

Android 12：

SystemUI只有一个进程，通过CarSystemBar进行副屏SystemBar的创建和显示，开关选项在`packages/apps/Car/SystemUI/res/values/config.xml`。

```
<bool name="config_enableBottom2SystemBar">true</bool>
```

Android 14：

SystemUI根据User进行创建，CarSystemBar通过OverlayManager实现，开关选项在`device/rockchip/common/car/overlay/packages/services/Car/car_product/rro/CarSystemUIPassengerRRO/res/values/config.xml`。

```
<bool name="config_enableBottomSystemBar">true</bool>
```

Tips: SystemBarConfig中会判断NotificationPanel，如开启了BottomSystemBar，则需使用BottomNotificationPanelViewMediator。代码位于`packages/apps/Car/SystemUI/src/com/android/systemui/car/systembar/SystemBarConfigs.java`。

1.5 休眠唤醒

AAOS中，电源外设等一般由VMCU进行控制，休眠唤醒等功能也与AOSP差异较大，目前可通过`dumpsys car_service suspend/resume`来模拟相关操作。

唤醒源：
当主机处于挂起模式时，必须禁用适当的唤醒源。常见的唤醒源包括心跳、调制解调器、Wi-Fi 和蓝牙。唯一有效的唤醒源必须是来自 VMCU 的中断才能唤醒 SoC。这假设 VMCU 可以侦听调制解调器的远程唤醒事件（例如远程发动机启动）。如果将此功能推送到 应用处理器（AP），则必须添加另一个唤醒源来为调制解调器提供服务。

客户需要根据实际硬件情况，在Kernel或者CarPowerManagerService.java中的doHandleFinish()中去关闭外设（RTC、wifi、以太网、蓝牙等）的二级待机唤醒源， 必要的话需要在handleWaitForVhal()中去恢复这些外设的功能。

AAOS相关文档：https://source.android.google.cn/docs/automotive/power?hl=zh_cn

注意：SDK默认没有配置电源策略power_policy.xml（SDK默认所有电源全开），客户需要根据自己的实际情况配置电源策略，详情参考https://source.android.com/docs/automotive/power/power_policy?hl=zh-cn。

参考图例：https://source.android.com/static/docs/devices/automotive/images/automotive_power_deep_sleep.png?hl=zh-cn

Android 14上可以支持对副屏的单独休眠，代码如下所示

`CarPowerManster.setDisplayPowerState(displayid,false/true)`

如需通过shell模拟，可使用`dumpsys car_service set-display-state displayid  false/true`进行模拟操作。

1.6 Cluster

AAOS的Cluster功能为Driver用户多占用一个屏幕作为Cluster屏使用，需要在config_occupant_display_mapping中进行配置，CarOccupantService启动后会解析出相关的Display。
启动后有两个Cluster的核心应用，分别为ClusterOsDouble和ClusterHomeSample。

- ClusterOsDouble
    - 为显示仪表盘相关数据，接收Vehicle Hal传递的相关指令并做出渲染效果，如档位、转速、车速、水温等等。
    - 开启VirtualDisplay供ClusterHomeSample中的其他效果显示，保证其他应用崩溃异常后不会让仪表数据显示异常。
    - **务必保证此应用的稳定性，此应用不做复杂的内容，仅简单显示车辆信息及提供VirtualDisplay。**
- ClusterHomeSample

  ClusterHomeSample 是ClusterHomeService服务默认指定的cluster home实现UI，其管理的是仪表盘中间的那块虚拟屏，如果要改变成其他apk，可以修改packages/services/Car/service/res/values/config.xml中

  ```
  <!--  The name of Activity who is in charge of ClusterHome. -->
  <string name="config_clusterHomeActivity"
  translatable="false">com.android.car.cluster.home/.ClusterHomeActivit</string
  >
  ```

最终客户应该模仿ClusterHomeSample实现自己的仪表盘引用，如显示导航、音乐、设置、电话等。

此应用主要功能：

- 根据ClusterOsDouble提供的VirtualDisplay，显示相关的功能（显示导航、音乐、设置、电话等）。
- 通过ClusterState展示不同的功能界面，ClusterState由ClusterManager中获取。

如有需要开启AAOS的Cluster服务，可以联系zhijun.xie@rock-chips.com/tao.weng@rock-chips.com获取相关补丁。

1.7 PRODUCT_REMOVE_PACKAGES

该配置为RK为客户开发的一个SDK 宏配置，可以方便客户更快的移除不需要的PRODUCT_PACKAGES，避免花费较多时间在寻找源码上。

Demo:

```
PRODUCT_REMOVE_PACKAGES += \
        Contacts \
        Music \
        android.hardware.lights-service.rockchip
```

1.8 Vehicle Hal

Vehicle Hal为AAOS的车载硬件抽象层（VHAL），可定义原始设备制造商 (OEM) 可以实现的属性，并会包含属性元数据。例如，属性是否为整数以及允许使用哪些更改模式。VHAL 接口以对属性（特定功能的抽象表示）的访问（读取、写入和订阅）为基础。

AAOS文档：https://source.android.google.cn/docs/automotive/vhal?hl=zh_cn

SDK现状：SDK提供了两种VHAL实现，一份为AAOS原生VHAL，代码位于`hardware/interfaces/automotive/vehicle`，另一份为Vehicle-dummy，代码位于`hardware/rockchip/rvcam`，如需打开请参考以下修改：

```
#device/rockchip/common
diff --git a/car/pre_google_car.mk b/car/pre_google_car.mk
index d85f017a..228be91f 100644
--- a/car/pre_google_car.mk
+++ b/car/pre_google_car.mk
@@ -56,7 +56,7 @@ $(call inherit-product,
device/rockchip/common/car/packages_generic_system.mk)
 PRODUCT_PACKAGES += \
             android.hardware.broadcastradio-service.default

-SOONG_CONFIG_rvcam_has_vhal := false
+SOONG_CONFIG_rvcam_has_vhal := true

 ifneq ($(strip $(SOONG_CONFIG_rvcam_has_vhal)), true)
 PRODUCT_PACKAGES += \
```

Android 14 推荐合作伙伴和Soc供应商将HIDL Hal实现替换为AIDL HAL实现，因此Vehicle HAL已经变更为了AIDL HAl，如有需要回退到HIDL HAL，请参考以下补丁：

```
# device/rockchip/common
diff --git a/car/manifest.xml b/car/manifest.xml
index 30a042f4..bd4a56b6 100644
--- a/car/manifest.xml
+++ b/car/manifest.xml
@@ -14,10 +14,19 @@
       limitations under the License.
 -->
 <manifest version="1.0" type="framework">
-    <hal format="aidl">
+    <!-- <hal format="aidl">
         <name>android.hardware.automotive.vehicle</name>
         <version>2</version>
         <fqname>IVehicle/default</fqname>
+    </hal> -->
+    <hal format="hidl">
+        <name>android.hardware.automotive.vehicle</name>
+        <transport>hwbinder</transport>
+        <version>2.0</version>
+        <interface>
+            <name>IVehicle</name>
+            <regex-instance>.*</regex-instance>
+        </interface>
     </hal>
     <hal format="aidl">
         <name>android.hardware.broadcastradio</name>
diff --git a/car/pre_google_car.mk b/car/pre_google_car.mk
index d85f017a..88adf69d 100644
--- a/car/pre_google_car.mk
+++ b/car/pre_google_car.mk
@@ -60,7 +60,8 @@ SOONG_CONFIG_rvcam_has_vhal := false

 ifneq ($(strip $(SOONG_CONFIG_rvcam_has_vhal)), true)
 PRODUCT_PACKAGES += \
-            android.hardware.automotive.vehicle@V1-default-service
+        android.hardware.automotive.vehicle@2.0-default-service
+        #     android.hardware.automotive.vehicle@V1-default-service
 endif

 # Car init.rc
# hardware/interfaces/compatibility_matrices
diff --git a/compatibility_matrices/compatibility_matrix.8.xml
b/compatibility_matrices/compatibility_matrix.8.xml
index 5ea075a377..2f12b12869 100644
--- a/compatibility_matrices/compatibility_matrix.8.xml
+++ b/compatibility_matrices/compatibility_matrix.8.xml
@@ -92,6 +92,14 @@
             <instance>default</instance>
         </interface>
     </hal>
+    <hal format="hidl" optional="true">
+        <name>android.hardware.automotive.vehicle</name>
+        <version>2.0</version>
+        <interface>
```

```
+            <name>IVehicle</name>
+            <regex-instance>.*</regex-instance>
+        </interface>
+    </hal>
     <hal format="aidl" optional="true">
         <name>android.hardware.automotive.vehicle</name>
         <version>1-2</version>
diff --git a/compatibility_matrices/compatibility_matrix.9.xml
b/compatibility_matrices/compatibility_matrix.9.xml
index 188746d79b..15098f814c 100644
--- a/compatibility_matrices/compatibility_matrix.9.xml
+++ b/compatibility_matrices/compatibility_matrix.9.xml
@@ -101,6 +101,14 @@
             <instance>default</instance>
         </interface>
     </hal>
+    <hal format="hidl" optional="true">
+        <name>android.hardware.automotive.vehicle</name>
+        <version>2.0</version>
+        <interface>
+            <name>IVehicle</name>
+            <regex-instance>.*</regex-instance>
+        </interface>
+    </hal>
     <hal format="aidl" optional="true">
         <name>android.hardware.automotive.remoteaccess</name>
         <interface>
```

AAOS VHAL和Vehicle-dummy VHAL在应用层及Hal层的调试方法相同，请参考以下方法：

- 调试VHAL

```
#列出参考 VHAL 支持的调试命令
dumpsys android.hardware.automotive.vehicle.IVehicle/default --help
#通过以下命令读取属性值（如 INFO_VIN）
dumpsys android.hardware.automotive.vehicle.IVehicle/default --get 0x11100100
```

- 通过CarService调试VHAL

```
dumpsys car_service -h
#相关参考方法
#模拟VHAL事件，注入车辆属性进行测试。
dumpsys car_service inject-vhal-event <PROPERTY_ID in Hex or Decimal> [zone]
data(can be comma separated list) [-t delay_time_seconds]
#模拟发送关机：AP_POWER_STATE_REQ::SHUTDOWN_PREPARE
dumpsys car_service inject-vhal-event 289475072 1,0
```

- 通过Kitchen Sink获取/设置Vehicle Hal的属性值。

1.9 Vehicle-dummy

目前SDK里提供了Vehicle-dummy的模拟实现，客户可用于模拟部分Vehicle Hal事件。如实际产品上，需要自行接入MCU和CAN等硬件。

Vehicle-dummy相关实现：

`hardware/rockchip/rvcam/drivers/vehicle-dummy/vehicle_dummy_hw.c`

Vehicle-dummy节点：

```
/sys/devices/platform/vehicle-dummy/ac_on
/sys/devices/platform/vehicle-dummy/auto_on
/sys/devices/platform/vehicle-dummy/defrost_left
/sys/devices/platform/vehicle-dummy/defrost_right
/sys/devices/platform/vehicle-dummy/fan_direction
/sys/devices/platform/vehicle-dummy/fan_speed
/sys/devices/platform/vehicle-dummy/gear
/sys/devices/platform/vehicle-dummy/hvac_on
/sys/devices/platform/vehicle-dummy/power_req
/sys/devices/platform/vehicle-dummy/recirc_on
/sys/devices/platform/vehicle-dummy/seat_temp_left
/sys/devices/platform/vehicle-dummy/seat_temp_right
/sys/devices/platform/vehicle-dummy/temp_left
/sys/devices/platform/vehicle-dummy/temp_right
/sys/devices/platform/vehicle-dummy/turn
```

Demo 1：

```
#模拟右转向
echo 1 > /sys/devices/platform/vehicle-dummy/turn
#模拟左转向
echo 2 > /sys/devices/platform/vehicle-dummy/turn
#关闭转向
echo 0 > /sys/devices/platform/vehicle-dummy/turn
```

Demo 2：

```
#模拟风向，参考VehicleHvacFanDirection
echo 1/2/3/4/6 > /sys/devices/platform/vehicle-dummy/fan_direction
```

Demo 3：

```
#模拟风速变化，参考HVAC_FAN_SPEED
echo 1/2/3/4/5/6 > /sys/devices/platform/vehicle-dummy/fan_speed
```

除以上外，尚有部分模拟实现可以参考Vehicle-dummy。

1.10 AAOS 音频

RockChip AAOS Audio模块 音区配置、音量调节、焦点管理、AudioControl等内容可参考文档：RKDocs\common\Audio\Rockchip_Developer_Guide_AAOS_Audio_CN.pdf。