

---

# 高通多媒体技术期刊 20150318

---



Qualcomm Technologies, Inc.

Confidential and Proprietary – Qualcomm Technologies, Inc.

机密和专有信息——高通技术股份有限公司



# Confidential and Proprietary – Qualcomm Technologies, Inc.

---

## Confidential and Proprietary – Qualcomm Technologies, Inc.

**NO PUBLIC DISCLOSURE PERMITTED:** Please report postings of this document on public servers or web sites to: [DocCtrlAgent@qualcomm.com](mailto:DocCtrlAgent@qualcomm.com). **禁止公开：**如在公共服务器或网站上发现本文档，请报告至：[DocCtrlAgent@qualcomm.com](mailto:DocCtrlAgent@qualcomm.com).

**Restricted Distribution:** Not to be distributed to anyone who is not an employee of either Qualcomm or its affiliated without the express approval of Qualcomm's Configuration Management. **限制分发：**未经高通配置管理部门的明示批准，不得发布给任何非高通或高通附属及关联公司员工的人。 Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc. 未经高通技术股份有限公司明示的书面允许，不得使用、复印、复制、或修改全部或部分文档，不得以任何形式向他人透露其内容。

The user of this documentation acknowledges and agrees that any Chinese text and/or translation herein shall be for reference purposes only and that in the event of any conflict between the English text and/or version and the Chinese text and/or version, the English text and/or version shall be controlling. 本文档的用户知悉并同意中文文本和/或翻译仅供参考之目的，如英文文本和/或版本和中文文本和/或版本之间存在冲突，以英文文本和/或版本为准。 This document contains confidential and proprietary information and must be shredded when discarded. 未经高通明示的书面允许，不得使用、复印、复制全部或部分文档，不得以任何形式向他人透露其内容。本文档含有高通机密和专有信息，丢弃时必须粉碎销毁。 Qualcomm reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed for any damages arising directly or indirectly by their use or application. The information provided in this document is provided on an "as is" basis. 高通保留未经通知即修改本档中提及的产品或信息的权利。本公司对使用或应用本文档所产生的直接或间接损失概不负责。本文档中的信息为基于现状所提供，使用风险由用户自行承担。

Qualcomm is a trademark of QUALCOMM Incorporated, registered in the United States and other countries. All QUALCOMM Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners. Qualcomm是高通公司在美国及其它国家注册的商标。所有高通公司的商标皆获得使用许可。其它产品和品牌名称可能为其各自所有者的商标或注册商标。

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited. 本文档及所含技术资料可能受美国和国际出口、再出口或转移出口法律的 限制。严禁违反或偏离美国和国际的相关法律。

**Qualcomm Technologies, Inc. 5775 Morehouse Drive San Diego, CA 92121 U.S.A.**  
高通技术股份有限公司，美国加利福尼亚州圣地亚哥市莫豪斯路 5775 号，邮编 92121

# Revision History

---

Revision	Date	Description
A	Mar 2015	Initial release

**Note:** There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

# Contents

---

- Camera
  - Video在某个场景出现AE闪烁几帧的现象而在Preview时没有发现
  - Video 时FaceAE引起的偏暗问题
  - Video时 TAF容易出现红框
- Display
  - Libscale.so 简介
  - Libscale.so 的调试
  - PTOR的简介
  - PTOR的调试
- Browser
  - Snapdragon Web Engine (SWE browser)代码同步tips
  - SWE浏览器下个版本计划



---

# Camera

---

# Video在某个场景出现AE闪烁几帧的现象而在Preview时没有发现

在8916/8939 project , 大多数的branch在mm-camera\mm-camera2\media-controller\modules\stats\q3a\aec\aec.h 中找到如下define

```
/* Slow conv parameters */
#define LUMA_TOLERANCE (2)
#define FRAME_SKIP (1)
#define HT_ENABLE (1)
#define HT_LUMA_TOLERANCE (4)
#define HT_THRES (3.0f)
#define HT_MAX (1.0f)
#define HT_GYRO_ENABLE (1)
/* End - Tuning parameters */

/* Start - Advanced tuning parameters */
#define REF_FRAME_RATE (30.0f)
#define STEP_DARK (17.0f)
#define STEP_BRIGHT (12.0f)
#define STEP_REGULAR (4.0f)
#define LUMA_TOL_RATIO_DARK (0.30f)
#define LUMA_TOL_RATIO_BRIGHT (0.20f)
#define RAW_STEP_ADJUST_CAP (2.0f)
#define ADJUST_SKIP_LUMA_TOLERANCE (4)
#define DARK_REGION_NUM_THRES (0.05f)
#define BRIGHT_REGION_NUM_THRES (0.05f)
#define HT_LUMA_THRES_LOW (0.5f)
#define HT_LUMA_THRES_HIGH (0.7f)
#define HT_LUMA_VAL_LOW (1.0f)
#define HT_LUMA_VAL_HIGH (0.6f)
#define HT_GYRO_THRES_LOW (0.5f)
#define HT_GYRO_THRES_HIGH (4.0f)
#define HT_GYRO_VAL_LOW (1.0f)
#define HT_GYRO_VAL_HIGH (0.4f)
```

# Video在某个场景出现AE闪烁几帧的现象而在Preview时没有发现

解决方法：

## 1. 先根据 log 来判断问题是否收敛过快而导致过冲

打开AEC的log：

```
adb shell setprop persist.camera.stats.debug.mask 1
```

从log中搜索关键字“aec\_process\_pack\_output:target\_”

```
E mm-camera-CORE: aec_process_pack_output:target_luma=44 cur_luma=35 stored_digital_gain=1.000000 exp_index=196,  
real_gain=1.003906, linecnt=334 , aec_settled 1
```

可以从这个关键字的Log中看出target luma和cur\_luma的值，当current luma 一直无法收敛到某一固定值时，尝试2 和 3 解法

## 2. 尝试降低收敛速度

在aec.h 中可以找到如下定义

```
#define RAW_STEP_ADJUST_CAP          (2.0f)
```

当定义的值越小，表示收敛的步长越小，但是如果这个值太小，影响收敛速度。

调试准则是，慢慢降低这个值直到闪烁消失

# Video在某个场景出现AE闪烁几帧的现象而在Preview时没有发现

---

## 3. 增大对亮度一致的容忍限度

在aec.h 中可以找到如下定义

```
#define LUMA_TOLERANCE          (2)
```

容忍值在2-8之间，在video的过程中，对每一帧的亮度一致性要求并没有那么严格，可以尝试改大这个值而使闪烁消失

这里需要强调，修改这组值只对video有作用，对拍照没有任何影响，如果需要修改拍照以及preview的值，有类似参数在chromatix header中可修改



# Video 时FaceAE引起的偏暗问题

现象：

前置摄像头拍摄video时，当有人脸出现时，在某些背光角度下，会出现整个画面非常暗，无法在收敛到正常亮度



# Video 时FaceAE引起的偏暗问题

解决：

## 1. 先确定是否有识别到人脸信息

从Log中搜索关键字“In face detection mode”，看是否有检测到face

Line 13164: 07-01 10:58:39.409 285 5204 E mm-camera-CORE: aec\_process\_calc\_roi\_luma\_ctrl: In face detection mode, skip\_needed 0, current speed 0.796875, status time 0.000000, off time 0.000000, on time 0.066667, fd\_status 1, num\_roi 1

## 2. 如果识别到人脸信息，尝试修改tuning文件中 FaceAE的权重

在chromatix header里面的 aec\_face\_weight 的weight值

0.900000f, /\* Weight \*/

调试方法：

先调整到一个比较大的值来确定是否由FaceAE引起，比如调整到0.99

0.99 相当于整帧的AE根据FACE窗口的统计来计算，如果0.99可以使这个现象消失，那么逐步降低这个权重值到，这个现象出现为止。

如果0.99 依然无法消除现象，考虑从bright region的角度来调试。

# Video 时FaceAE引起的偏暗问题

3. 对于背光比较亮的情况，考虑tuning bright region的thresholds和weight  
调试方法：

1. 保持bright region 的threshold，调整weight
2. 保持 weight不变，调整threshold

找出一个平衡值可以消除这种背光人脸无法亮起来的现象

调试参数值：

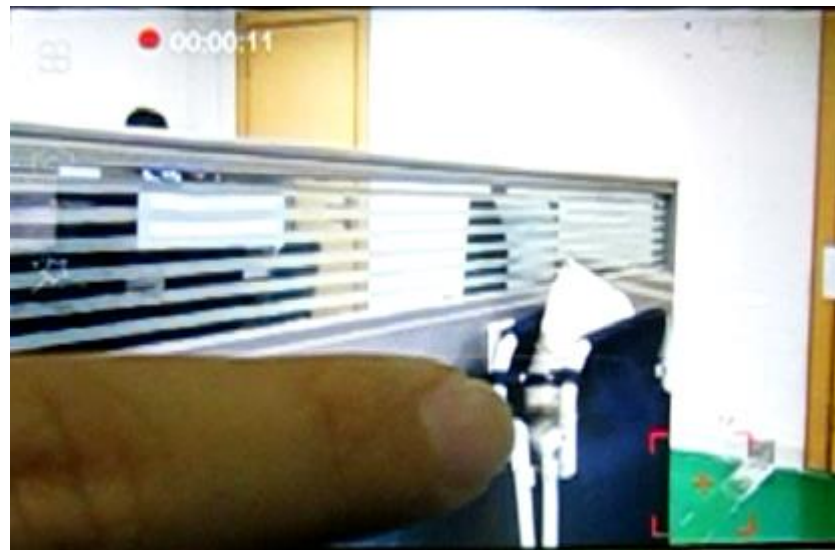
在chromatix 头文件中

270, /\* Bright Weight Lux Index Trigger \*/ --超过这个index，才会触发bright region AE  
0.500000f, /\* Bright Weight Def \* --- weight，权重  
220, /\* Bright Level \*/ --- 亮度大于220，才会是 bright

# Video时 TAF容易出现红框

现象：

在video时，touch AF，会发现比较容易出现对焦failed的红框



# Video时 TAF容易出现红框

解决方法：

## 1. 滤波器已经滤波方式的确认

- 需求首先确定是软件滤波还是硬件滤波

先打开AF log，确认在video mode的时候，是使用软件 stats还是硬件stats，8939可以使用软件的AF滤波器，而8916需要同时考虑软件滤波能否算的过来的问题

open AF log:

```
adb shell setprop persist.camera.stats.debug.mask 4
```

搜索关键字“af\_single\_start\_srch”

```
E mm-camera-CORE: af_single_start_srch: using PAAF, mode 1
```

Enable PAAF: 在ad XXXX\_camcorder.h 中，找到 PAAF\_enable，把0改成1，enable软件滤波方式

- 如果已经确定使用软件滤波，那么还需要进一步确认，滤波器类型，IIR的3阶滤波还是6阶滤波

滤波器设定在XXXX\_camcorder.h中可以查找和修改，关键字“af\_vfe\_sw\_iir\_hpf\_t”

# Video时 TAF容易出现红框

- 如果只用左边3阶滤波，对平坦区的统计，FV值准确度不如6阶。  
但是低阶计算开销较小，如果在8916上使用高阶滤波，同时要兼顾是否FV值经常为0，出现无法实时完成计算的问题

```
{
    1.00000000f, /* a00 */
    -1.0772000f, /* a01 */
    0.61280000f, /* a02 */
    0.00000000f, /* a03 */
    0.00000000f, /* a04 */
    0.00000000f, /* a05 */
},
/* b */
{
    0.19360000f, /* a00 */
    0.00000000f, /* a01 */
    -0.1936000f, /* a02 */
    0.00000000f, /* a03 */
    0.00000000f, /* a04 */
    0.00000000f, /* a05 */
},
```

低阶IIR滤波器

```
{
    1.00000000f, /* a00 */
    -1.3642469f, /* a01 */
    0.75484900f, /* a02 */
    1.00000000f, /* a03 */
    -0.8725360f, /* a04 */
    0.68906000f, /* a05 */
},
/* b */
{
    0.12260000f, /* a00 */
    0.00000000f, /* a01 */
    -0.1226000f, /* a02 */
    0.33660000f, /* a03 */
    0.00000000f, /* a04 */
    -0.3366000f, /* a05 */
},
```

高阶IIR滤波器

# Video时 TAF容易出现红框

2. 在tuning 文件中降低 search size，使search的步长变小，副作用就是会影响对焦时间

调试方法：

1. tuning 文件是adXXXX\_camcorder.h
2. {10, 16, 22, 28, 45,}，找到对应的step size的参数结构体，  
第一个值表示从far end 到 Hyperfocal的步长  
第二个值表示从Hyperfocal 到 region1搜索的步长  
第三个值表示从region1到region2搜索的步长  
第四个值表示从region2到region3搜索的步长  
第五个值表示从region3到 near end搜索的步长  
先从Log中看是哪一段对焦需要优化，再改相应区域的值。

3. region以及far end，near end的定义

这与每个模组，每个马达的特性息息相关，在tuning文件中，可以得到这个设定值，结构体如下图

# Video时 TAF容易出现红框

```
{
    SINGLE_INF_LIMIT_IDX, /* Variable name: CAF_far_end */
    SINGLE_NEAR_LIMIT_IDX, /* Variable name: CAF_near_end */
    SINGLE_INF_LIMIT_IDX, /* Variable name: TAF_far_end */
    SINGLE_NEAR_LIMIT_IDX, /* Variable name: TAF_near_end */
    SINGLE_50CM_IDX, /* Variable name: srch_rgn_1 */
    SINGLE_20CM_IDX, /* Variable name: srch_rgn_2 */
    SINGLE_14CM_IDX, /* Variable name: srch_rgn_3 */
    SINGLE_50CM_IDX, //SINGLE_INF_LIMIT_IDX, /* Variable name: fine_srch_rgn */
    SINGLE_INF_LIMIT_IDX, /* Variable name: far_zone */
    SINGLE_10CM_IDX, /* Variable name: near_zone */
    SINGLE_14CM_IDX, /* Variable name: mid_zone */
    SINGLE_HYP_F_IDX, /* Variable name: far_start_pos */
    SINGLE_NEAR_LIMIT_IDX, /* Variable name: near_start_pos */
    SINGLE_120CM_IDX, /* Variable name: init_pos */
},
```

## 3. 考虑在video时，关掉fine search

### 调试方法：

在tuning文件中找到“Variable name: single\_optic\_t”结构体

在此结构体中可以找出“fine\_srch\_rgn”的变量，将这个值设到到 SINGLE\_INF\_LIMIT\_IDX，可以关掉fine search 功能。

SINGLE\_50CM\_IDX -> SINGLE\_INF\_LIMIT\_IDX, /\* Variable name: fine\_srch\_rgn \*/





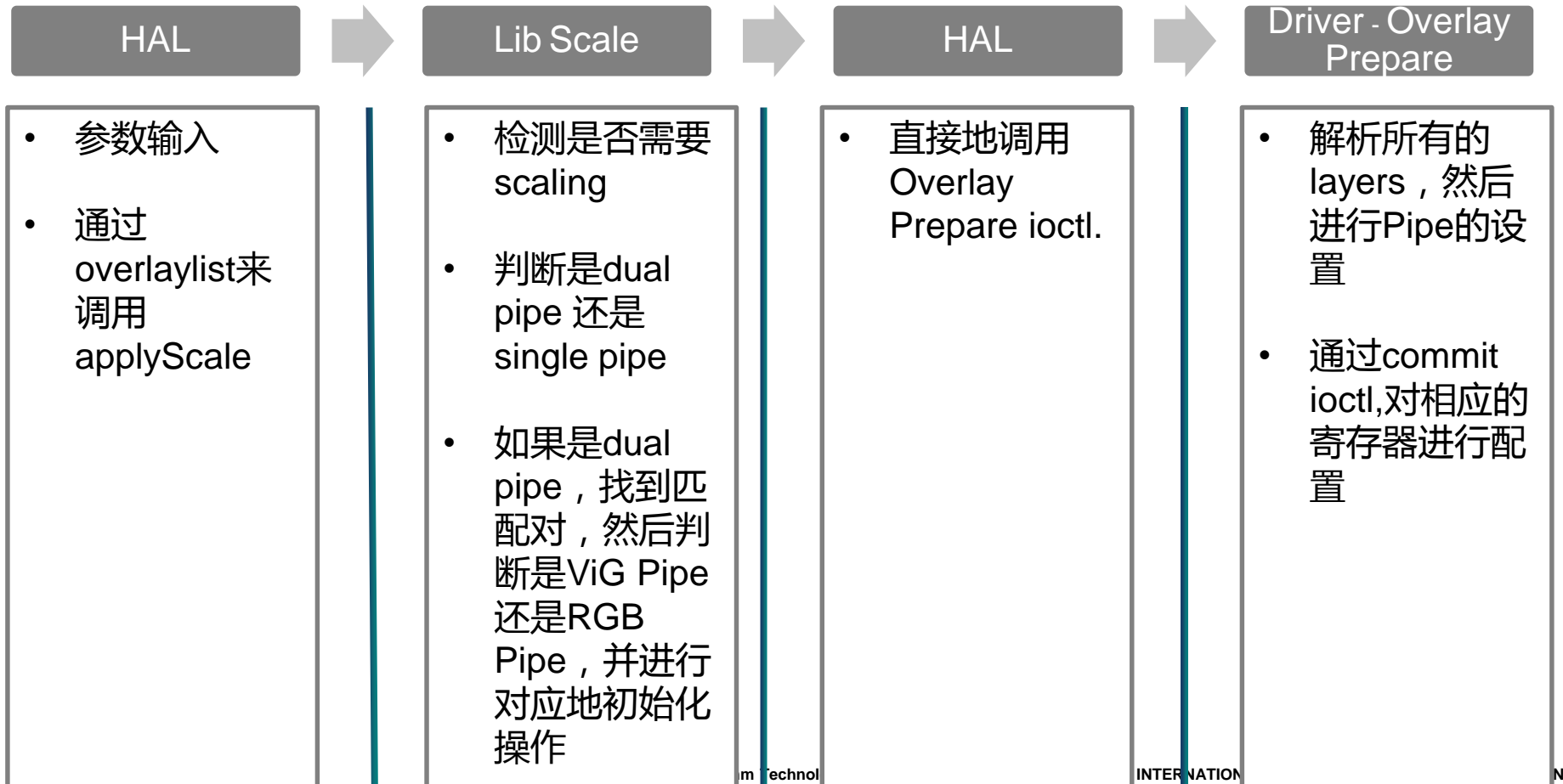
---

# Display

---

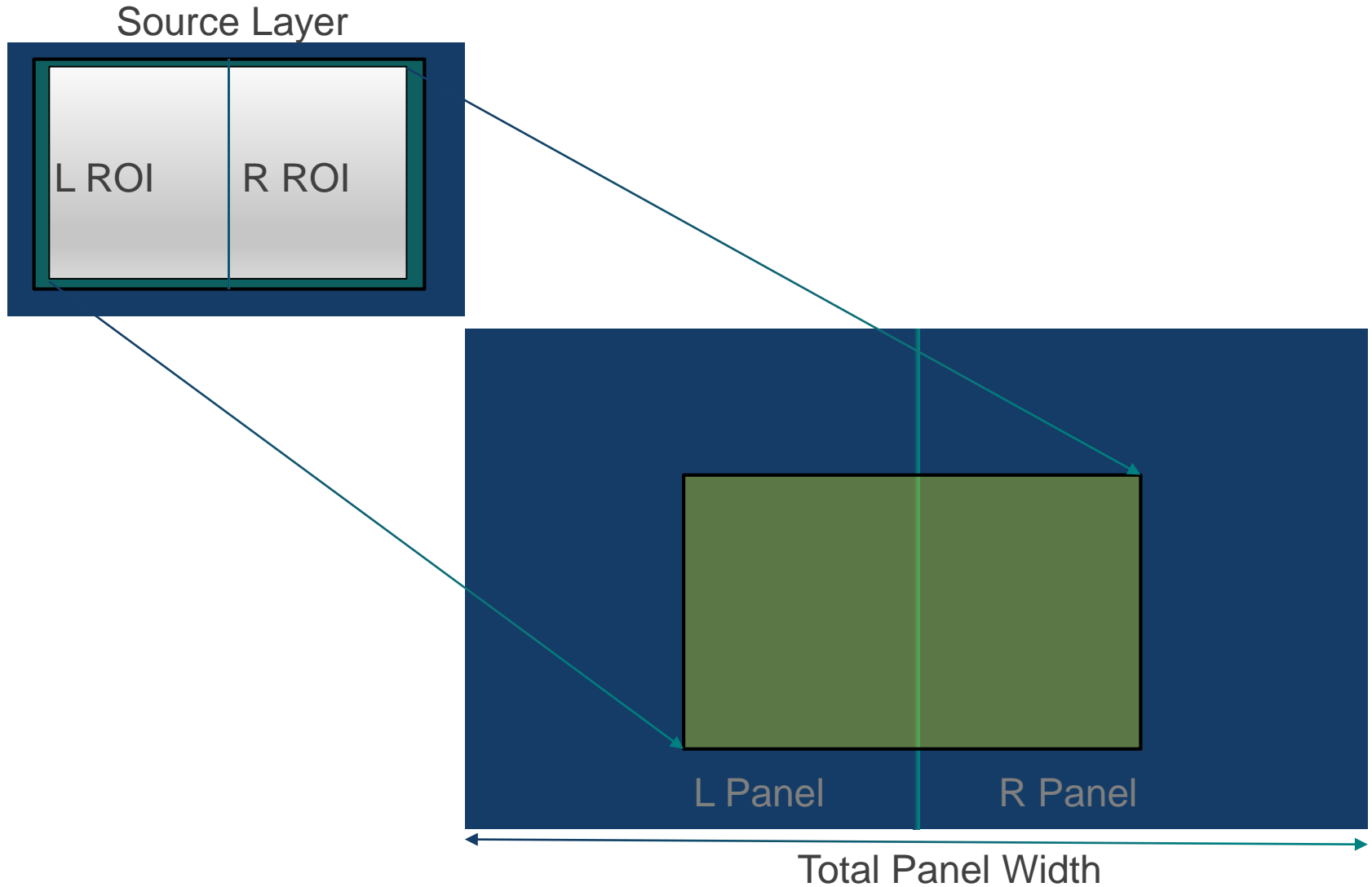
# Libscale.so 简介

- Scaler 模块在SSPP(Source Surface Processor Pipe)中进行实现，其中包括
  - ViG Pipe
  - RGB Pipe
- 流程框图如下：



# Libscale.so 简介- 续一

- Scaler 的示意图



# Libscale.so 的调试

---

- 对于scaler来说，客户只能拿到lib库，故只能通过删除libscale.so来确定问题是否与此库相关。
- 如下面命令：
  - adb root
  - adb shell
  - cd /vendor/lib/
  - rm libscale.so
  - exit
  - adb shell sync
  - adb reboot
- 如果是64位的系统，请删除/vendor/lib64/目录下的libscale.so.

# 如何使能libscale.so的debug 信息

---

- 在build.prop中，添加debug.scale.logs，具体如下
  - adb root
  - adb shell
  - adb pull /system/build.prop .
  - debug.scale.logs=1
  - adb push build.prop /system/
  - adb shell chmod 644 /system/build.prop
  - adb shell sync
  - adb shell reboot
  - adb shell logcat -v threadtime
- 这样，在logcat中，可以看到与libscale相关的log，关键字：MDP\_SCALE。
- 举例如下：对于一个layer来说：

# 如何使能libscale.so的debug 信息- 续一

- 举例来说，对于一个layer的信息

- D/MDP\_SCALE( 330): Original:-
- D/MDP\_SCALE( 330): LEFT:- Src x = 0, y = 0, w = 1131, h = 2416
- D/MDP\_SCALE( 330): Dst x = 139, y = 732, w = 661, h = 1381
- D/MDP\_SCALE( 330): RIGHT:- Src x = 1131, y = 0, w = 469, h = 2416
- D/MDP\_SCALE( 330): Dst x = 0, y = 732, w = 274, h = 1381
- D/MDP\_SCALE( 330): Final:-
- D/MDP\_SCALE( 330): LEFT:- Src x = 0, y = 0, w = 1132, h = 2416
- D/MDP\_SCALE( 330): Dst x = 139, y = 732, w = 661, h = 1381
- D/MDP\_SCALE( 330): RIGHT:- Src x = 1131, y = 0, w = 468, h = 2416
- D/MDP\_SCALE( 330): Dst x = 0, y = 732, w = 274, h = 1381
- D/MDP\_SCALE( 330): DECI:- h\_deci=0 vert\_deci=0
- D/MDP\_SCALE( 330): Input Format = 13
- D/MDP\_SCALE( 330): RGB scalar LEFT:-
- D/MDP\_SCALE( 330): phase\_step\_x = 36c265, phase\_step\_y = 37fb8d
- D/MDP\_SCALE( 330): init\_phase\_x = 0, init\_phase\_y = 0
- D/MDP\_SCALE( 330): roi\_w = 1132, flags = 60200
- D/MDP\_SCALE( 330): RGB scalar RIGHT:-
- D/MDP\_SCALE( 330): phase\_step\_x = 36c265, phase\_step\_y = 37fb8d
- D/MDP\_SCALE( 330): init\_phase\_x = ffe3eec9, init\_phase\_y = 0
- D/MDP\_SCALE( 330): roi\_w=468, flags=60300

# 如何使能libscale.so的debug 信息- 续二

```
D/MDP_SCALE( 330): Pixels plane = 0  flags = 60200  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 0, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixels plane = 1  flags = 60200  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 0, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixels plane = 2  flags = 60200  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 0, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixels plane = 3  flags = 60200  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 0, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 0  flags = 60200
D/MDP_SCALE( 330):      left ftch = 0, rpt = 0
D/MDP_SCALE( 330):      right fetch = 1, rpt = 0
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 1  flags = 60200
D/MDP_SCALE( 330):      left ftch = 0, rpt = 0
D/MDP_SCALE( 330):      right fetch = 1, rpt = 0
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 2  flags = 60200
D/MDP_SCALE( 330):      left ftch = 0, rpt = 0
D/MDP_SCALE( 330):      right fetch = 1, rpt = 0
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 3  flags = 60200
D/MDP_SCALE( 330):      left ftch = 0, rpt = 0
D/MDP_SCALE( 330):      right fetch = 1, rpt = 0
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixels plane = 0  flags = 60300  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 1, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixels plane = 1  flags = 60300  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 1, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixels plane = 2  flags = 60300  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 1, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixels plane = 3  flags = 60300  noscale x,y = (0, 0)
D/MDP_SCALE( 330):      left extn pixels = 1, right extn pixels = 1
D/MDP_SCALE( 330):      top extn pixels = 0, botm extn pixels = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 0 flags = 60300
D/MDP_SCALE( 330):      left ftch = 1, rpt = 0
D/MDP_SCALE( 330):      right fetch = 0, rpt = 1
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 1 flags = 60300
D/MDP_SCALE( 330):      left ftch = 1, rpt = 0
D/MDP_SCALE( 330):      right fetch = 0, rpt = 1
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 2 flags = 60300
D/MDP_SCALE( 330):      left ftch = 1, rpt = 0
D/MDP_SCALE( 330):      right fetch = 0, rpt = 1
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
D/MDP_SCALE( 330): Pixel Fetchplane = 3 flags = 60300
D/MDP_SCALE( 330):      left ftch = 1, rpt = 0
D/MDP_SCALE( 330):      right fetch = 0, rpt = 1
D/MDP_SCALE( 330):      top fetch = 0, rpt = 0
D/MDP_SCALE( 330):      botm fetch = 0 rpt = 1
```

# Libscale 照成Split screen 问题

- 在使用Dual DSI panel时，有时候会有split screen的问题，部分log如下：

**D/MDP\_SCALE( 234): Single Overlay Scale detected**

**D/MDP\_SCALE( 234): Single pipe scaling is not handled**

E/qdoverlay( 234): Bad ov dump: mdp\_overlay z=2 fg=0 alpha=24 mask=-1 flags=0x860100 id=4

E/qdoverlay( 234): src msmfb\_img w=1664 h=2400 format=13 MDP\_RGBA\_8888

E/qdoverlay( 234): src\_rect mdp\_rect x=1 y=84 w=1438 h=2315

E/qdoverlay( 234): dst\_rect mdp\_rect x=275 y=460 w=426 h=482

D/MDP\_SCALE( 234): Single Overlay Scale detected

D/MDP\_SCALE( 234): Single pipe scaling is not handled

- 从上面的信息来看，libscale中有问题，此时，请提case到高通来帮忙解决。



# PTOR的简介

- PTOR(Peripheral Tiny Overlap Removal)功能：
  - 对于4 Layers的场景，比如，HomeScreen，包括Status bar, Launcher, Wallpaper and Nav Bar，由于MDP带宽限制的原因，此时会使用GPU合成，从而影响系统的性能。
  - 为了克服这种情况，对于有重叠的小区域，使用copybit，从而可以优化性能。
- PTOR使用的条件：
  - 请查看函数fullMDPCompWithPTOR
- PTOR struct：
  - struct PtorInfo {
  - int count;
  - int layerIndex[MAX\_PTOR\_LAYERS];
  - hwc\_rect\_t displayFrame[MAX\_PTOR\_LAYERS];
  - **bool isActive() { return (count>0); }**
  - int getPTORArrayIndex(int index) {
  - int idx = -1;
  - for(int i = 0; i < count; i++) {
  - if(index == layerIndex[i])
  - idx = i;
  - }
  - return idx;
  - }
  - };

# PTOR的代码流程

- 在函数MDPComp::init中，读取persist.hwc.ptor.enable的值，如果为true 或者“1”，创建一个copybit对象，部分代码如下：
  - ```
bool defaultPTOR = false;
```
  - ```
//Enable PTOR when "persist.hwc.ptor.enable" is not defined for
```
  - ```
//8x16 and 8x39 targets by default
```
  - ```
if((property_get("persist.hwc.ptor.enable", property, NULL) <= 0) &&
```
  - ```
    (qdutils::MDPVersion::getInstance().is8x16() ||
```
  - ```
     qdutils::MDPVersion::getInstance().is8x39())) {
```
  - ```
    defaultPTOR = true;
```
  - ```
}
```
  - ```
if (defaultPTOR || (!strncasecmp(property, "true", PROPERTY_VALUE_MAX)) ||
```
  - ```
    (!strcmp(property, "1", PROPERTY_VALUE_MAX))) {
```
  - ```
    ctx->mCopyBit[HWC_DISPLAY_PRIMARY] = new CopyBit(ctx,
```
  - ```
        HWC_DISPLAY_PRIMARY);
```
  - ```
}
```
- 在函数hwc\_sync中，如果PTORinfo 是active 状态，获取一个render buffer为copybit，部分代码如下：
  - ```
if ((fd >= 0) && !dpy && ctx->mPtorInfo.isActive()) {
```
  - ```
    // Acquire c2d fence of Overlap render buffer
```
  - ```
    if(LIKELY(!swapzero) )
```
  - ```
        acquireFd[count++] = fd;
```
  - ```
}
```

# PTOR的调试

---

- 在build.prop中，具体如下
  - adb root
  - adb shell
  - adb pull /system/build.prop .
  - persist.hwc.ptor.enable=false or true // false 是disable，true 是enable
  - adb push build.prop /system/
  - adb shell chmod 644 /system/build.prop
  - adb shell sync
  - adb shell reboot
- 或者通过adb command对其进行操作：
  - adb shell setprop persist.hwc.ptor.enable false
  - adb shell stop
  - adb shell start //注意一定要执行stop/start



---

# Browser

---

# Snapdragon Web Engine (SWE browser)代码同步tips

---

- SWE浏览器代码同步需要在codeaurora网站完成，具体指南在网页

<https://www.codeaurora.org/xwiki/bin/Chromium+for+Snapdragon/Build>中

- 因为codeaurora.org网站的访问速度在一些地方可能较慢，高通在北京建立了映像服务器以加速中国用户的访问，请参见<https://www.codeaurora.org/xwiki/bin/Support/CodeAuroraMirrors>并使用此映像来加速SWE代码的同步

- 具体设置如下：

git config --global [url.git://cn.codeaurora.org.insteadof](https://www.codeaurora.org/xwiki/bin/Support/CodeAuroraMirrors)  
[git://codeaurora.org](https://www.codeaurora.org/xwiki/bin/Support/CodeAuroraMirrors)

## SWE浏览器下个版本计划

---

- SWE浏览器将在下个版本中基于ChromeShell开发
- 与目前基于修改过的 Android\_WebView的SWE相比，ChromeShell将提供更好的多进程支持并充分使用Chromium中已支持特性和集成的代码
- OEMs在SWE浏览器UI上的现有定制可能会受影响，请提前准备。

---

## Questions?

<https://support.cdmatech.com>

