

密级状态： 绝密() 秘密() 内部资料() 公开(√)

RK SVEP MEMC User Guide

(技术部，图形计算平台中心)

文件状态： <input type="checkbox"/> 草稿 <input type="checkbox"/> 正在修改 <input checked="" type="checkbox"/> 正式发布	当前版本：	1.4.3
	作 者：	GPU Team
	完成日期：	2023-08-29
	审 核：	熊伟
	审核日期：	2023-08-29

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2023 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

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

本文主要介绍内容如下：

- SVEP (SVEP, Super Vision Enhancement Process, 超级视觉增强处理) 算法模块中的MEMC (MEMC, Motion Estimation and Motion Compensation, 运动估计与运动补偿) 的技术说明；
- Android 系统显示框架集成SVEP-MEMC技术对外接口说明。

适用平台

芯片平台	系统平台	系统平台版本
RK3588	Android	12 and above

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

MEMC版本	HWC版本	作者	审核	更新时间	修订说明
1.4.3	1.5.143	GPU Team	熊伟	2023/08/29	【优化】优化算法鉴权逻辑; 【优化】解决若干稳定性问题;
1.4.0	1.5.119	GPU Team	熊伟	2023/07/28	【新增】集成实现MEMC功能; 【新增】实现MEMC控制接口属性;

目录

一、概述

1.1 系统集成架构

1.2 数据流处理说明

二、MEMC 系统功能接口说明

2.1 版本号查询接口

2.2 模式使能接口

2.3 左右对比模式

2.5 VendorStorage 授权接口

 2.5.1 VendorStorage 分区介绍

 2.5.2 VendorStorage Write 实例代码

 2.5.3 MEMC 授权流程

2.6 OSD 字幕接口

 2.6.1 OSD修改方法介绍

 2.6.2 字幕限制信息

 2.6.3 关闭OSD字幕

 2.6.4 OSD-Oneline 模式

2.7 MEMC 使能流程

 2.7.1 通用图层匹配逻辑

 2.7.2 黑名单处理流程说明

 2.7.3 白名单处理流程说明

 2.7.3 配置文件说明

三、FAQ

3.1 简单介绍MEMC具体做了哪些处理?

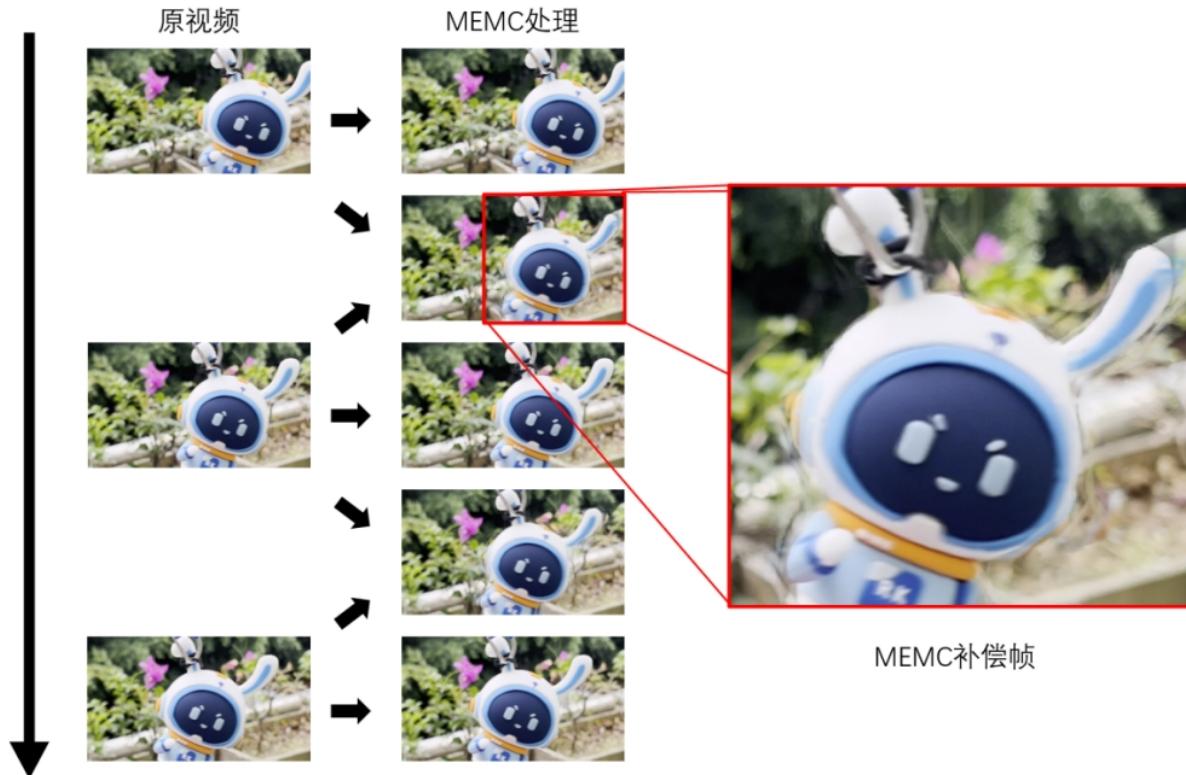
3.2 如何确认MEMC授权成功?

3.3 如何判断是否正确集成并开启MEMC功能?

3.4 Android SDK如何使能MEMC功能?

一、概述

MEMC (MEMC, Motion Estimation and Motion Compensation, 运动估计与运动补偿) , 算法利用深度神经网络从前前后两帧原始帧计算出中间的预测帧, 提高视频帧率以获得较原始视频更流畅的感官体验。插帧示意图如下图所示。



目前支持的MEMC模式如下：

MEMC模式	输入输出分辨率	输入帧率 / Fs	输出帧率 / FPS	备注
1080p	1920x1080	30	60	
2160p	3840x2160	30	60	

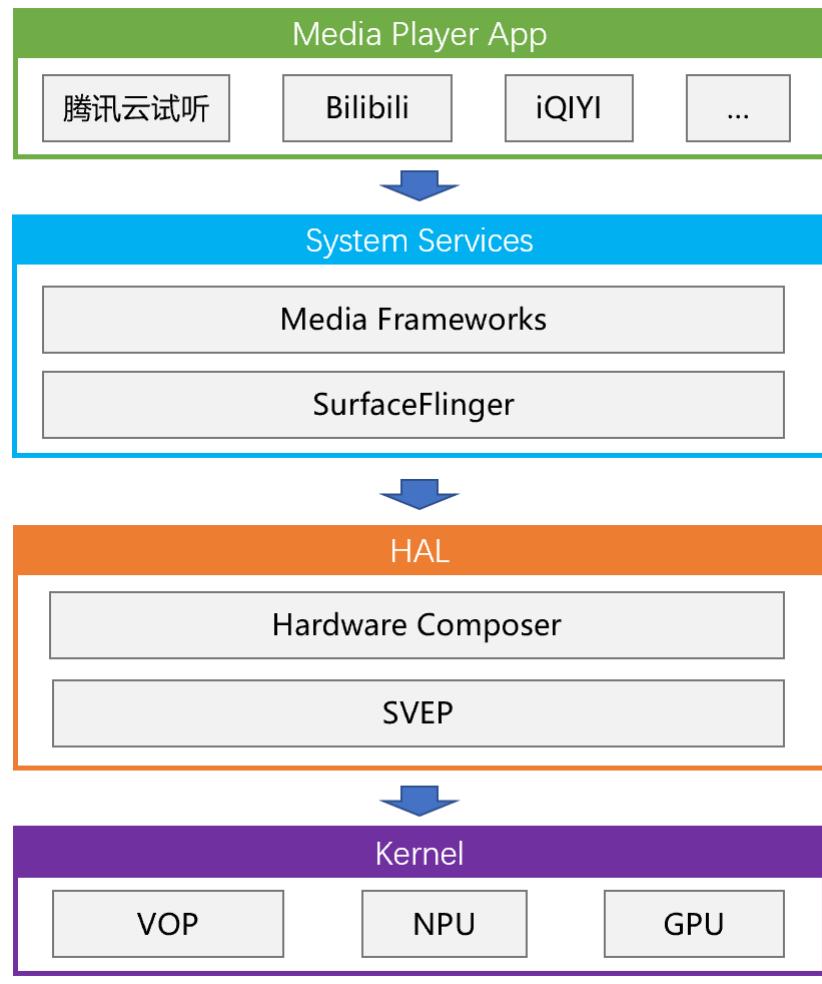
注意1：若 **输入分辨率不满足** 标准MEMC模型的分辨率，系统端会适配到更大一级的分辨率模型进行运动补偿，例如：若输入1024x600分辨率视频，则采用1080p MEMC模型进行处理；

注意2： MEMC算法不限制输入帧率，在性能允许的情况下，会按输入帧率的2倍进行输出。

1.1 系统集成架构

目前MEMC集成在 Android 系统显示框架内部,

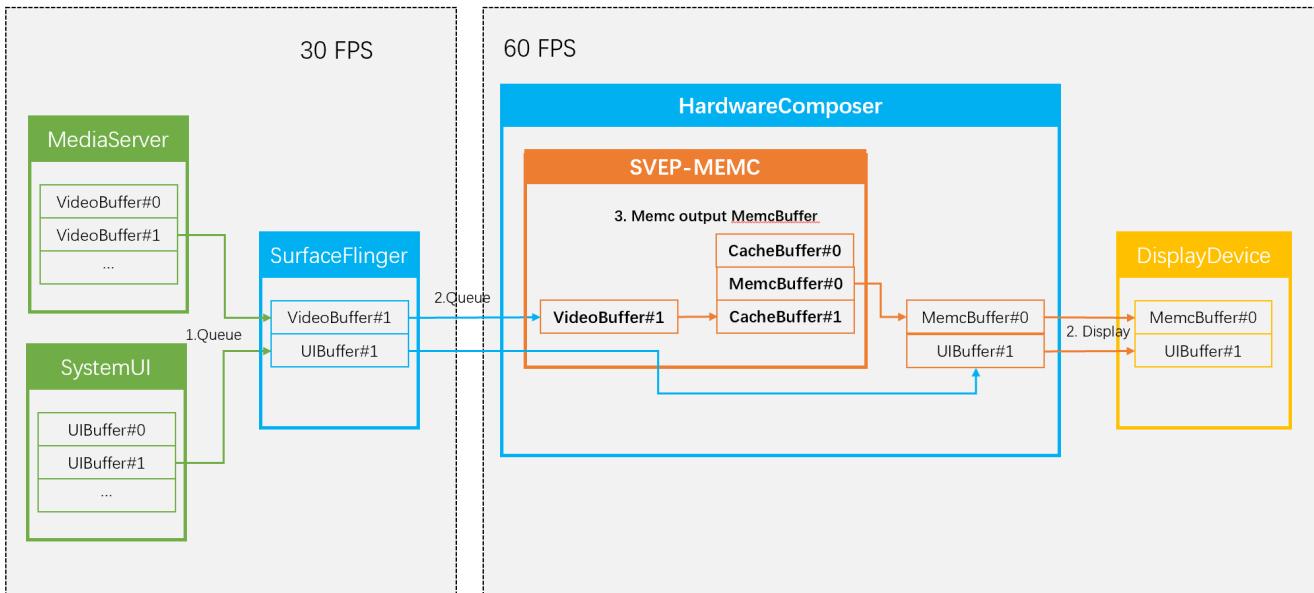
- 整体架构图:



1. 媒体应用请求Media Frameworks播放视频;
2. Media Frameworks向SurfaceFlinger提交视频帧, SurfaceFlinger请求HardwareComposer服务处理视频帧;
3. HardwareComposer服务调用MEMC库对视频帧执行运动补偿插帧操作;
4. HardwareComposer 获得视频补偿帧, 提交屏幕送显;

1.2 数据流处理说明

MEMC 数据流处理说明如下图所示：



1. 视频播放场景，系统同时提交VideoBuffer#0与UIBuffer#0图像到SurfaceFlinger 服务，请求送显；
2. SurfaceFlinger收集当前应用提交的所有待显示图像（VideoBuffer#0与UIBuffer#0），打包送给 HardwareComposer服务；
3. **Vsync0:** HardwareComposer接收到VideoBuffer#0，由于是第一帧，无参考帧，故HWC直接将 VideoBuffer#0送显DisplayDevices；
4. **Vsync1:** 由于视频帧率为30帧，故当前Vsync周期，DisplayDevices依然显示VideoBuffer#0；
5. **Vsync2:** HardwareComposer接收到VideoBuffer#1，结合VideoBuffer#0调用MEMC算法输出补偿帧 MemcBuffer#0送显DisplayDevices，DisplayDevices显示MemcBuffer#0；
6. **Vsync3:** 由于视频帧率为30帧，故当前Vsync周期，没有新的视频帧送显，故HWC将VideoBuffer#1送显给 DisplayDevices；
7. **Vsync4:** HardwareComposer接收到VideoBuffer#2，结合VideoBuffer#1调用MEMC算法输出补偿帧 MemcBuffer#1送显DisplayDevices，DisplayDevices显示MemcBuffer#1；
8. 由此，系统将30帧片源补偿为60帧，送显DisplayDevices；

注意: 上述流程描述暂不考虑MEMC算法耗时，仅用于描述MEMC算法实现流程；

二、MEMC 系统功能接口说明

MEMC技术目前已深度集成在系统显示后端服务，应用通过Android Property设置MEMC状态，即可实现MEMC相关功能的配置：

- 版本查询接口：vendor.memc.version
- 功能模式接口：
 - MEMC功能开关接口：persist.sys.memc.mode
 - 左右对比模式接口：persist.sys.memc.contrast_mode
 - OSD关闭模式接口：persist.sys.svep.disable_memc_osd
 - MEMC授权ID配置：ro.vendor.memc.vsid

2.1 版本号查询接口

属性名称	典型值	值说明
vendor.memc.version	Memc-1.4.2	MEMC接口库版本号
vendor.ghwc.version	HWC2-1.5.140	HWC版本号

可通过以下命令获取版本信息：

```
adb shell getprop vendor.memc.version
## 输出版本信息为: Memc-1.4.2 , 具体版本号等于实际的交付版本号
adb shell getprop vendor.ghwc.version
## 输出版本信息为: HWC2-1.5.140 , 具体版本号等于实际的交付版本号
```

注意：必须要同时确认HWC与MEMC库版本号，若版本库不匹配可能会导致异常；

2.2 模式使能接口

属性名称	缺省值	值范围	值说明
persist.sys.memc.mode	0	0 / 1	MEMC模式全局开关（用户使用） 0：关闭MEMC模式，不进行任何画质增强处理 1：开启MEMC模式，仅针对视频播放图层
sys.svep.runtime_disable	0	0 / 1	MEMC运行时强制关闭（系统使用） 0：正常MEMC处理流程 1：强制关闭MEMC模式

注意：两项模式使能接口使用建议如下：

- MEMC模式全局开关：通常作为MEMC全局开关，例如Setting内部的MEMC开关选项；
- MEMC运行时强制关闭：通常提供给系统服务（解码模块）强制关闭MEMC模式，例如播放60帧片源时建议设置为1

可通过以下命令获取并设置MEMC使能模式：

```
## 获取:  
adb shell getprop persist.sys.memc.mode  
adb shell getprop sys.svep.runtime_disable  
  
## 设置:  
adb shell setprop persist.sys.memc.mode 1  
adb shell setprop sys.svep.runtime_disable 1
```

MEMC使能效果如下图：



其中OSD字幕如下图，存在此OSD说明模式正常开启：



详细参数说明：

- RKNPU-SVEP-MEMC: OSD默认字幕，自定义接口可查看《2.6 OSD字幕接口》进行修改；
- Input: 实际输入片源尺寸；
- Output: 实际增强输出尺寸；

2.3 左右对比模式

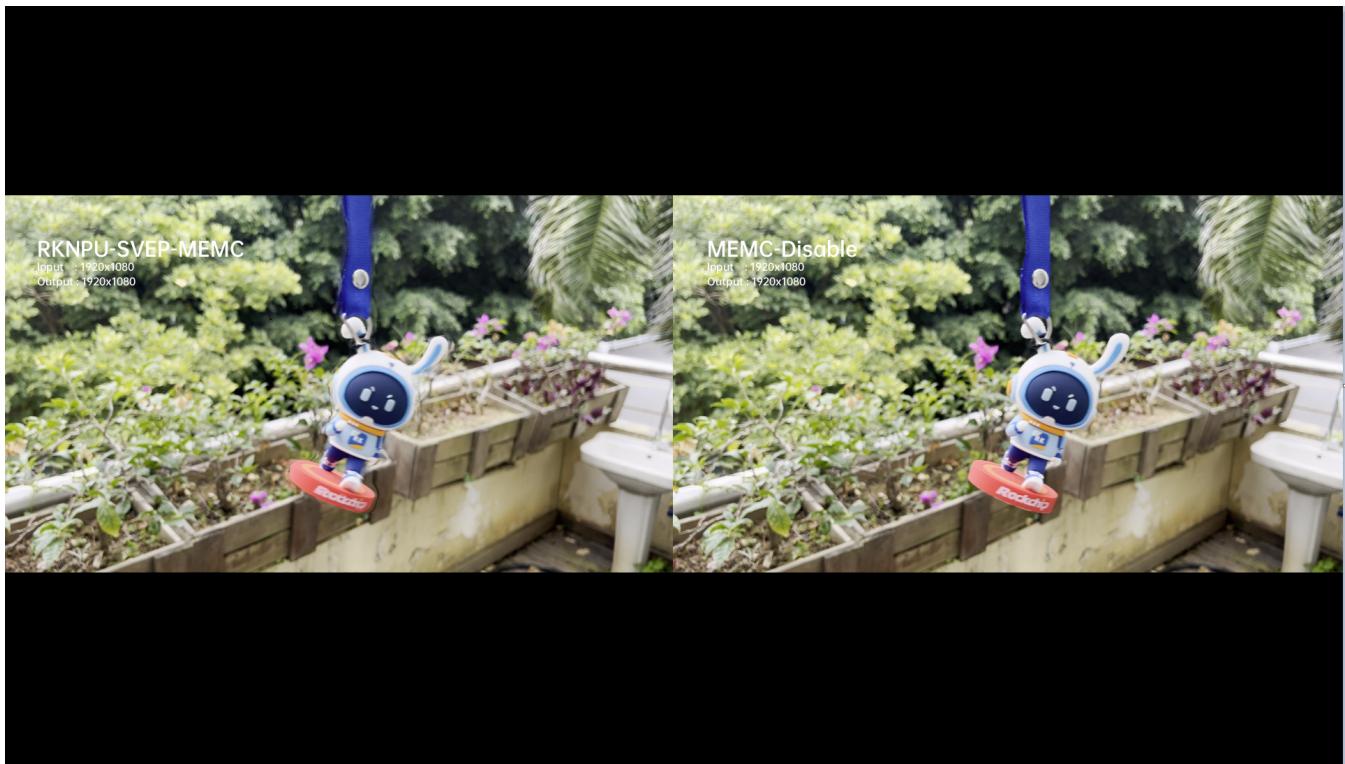
属性名称	缺省值	值范围	值说明
persist.sys.memc.contrast_mode	0	0 / 1	图像左右对比模式开关 分割线以左为MEMC效果 分割线以右为原效果 0: 关闭图像对比模式 1: 开启图像对比模式

persist.sys.memc.contrast_mode 说明:

MEM左右对比模式开关，左图像为MEMC处理后的60帧图像，右图原30帧图像

```
## 获取:  
adb shell getprop persist.sys.memc.contrast_mode  
  
## 设置使能MEMC左右对比模式:  
adb shell setprop persist.sys.memc.contrast_mode 1  
## 设置关闭MEMC左右对比模式:  
adb shell setprop persist.sys.memc.contrast_mode 0
```

使能MEMC左右对比模式显示效果如下：



注意：左右对比模式是利用RGA实现，会大幅增加系统DDR负载，仅建议在演示项目中使用；

2.5 VendorStorage 授权接口

MEMC算法正常执行需要授权激活码，只有授权激活通过的设备才可以体验MEMC算法。

注意：激活码激活目标设备后即与目标芯片OTP绑定，无法再激活其他芯片，请谨慎保管。

VendorStorage 分区介绍：可参考《Rockchip_Application_Notes_Storage_CN.pdf》文档；

2.5.1 VendorStorage 分区介绍

MEMC算法激活码分区规划：如下表：

算法模块	ID	Len	分区说明
MEMC	62	50	MEMC激活码存放ID号

分区说明： VendorStorage ID 0-31 保留为通用 SDK 功能使用，用户自定义区间为 32-65535任意ID。

SVEP包含以下两个图像算法：

- SR (Super Resolution, 超级分辨率) ；
- MEMC (Motion Estimation and Motion Compensation, 运动估计与运动补偿) ；

对应的已使用的ID号如下表：

模块	ID	len	内容说明
SR	17	50	SR 旧激活码存放位置，V2.0.0 版本已迁移到ID=60，但是会兼容检查 ID=17 的激活码
SR	60	50	SR 新激活码存放位置
MEMC	62	50	MEMC激活码存放ID号

2.5.2 VendorStorage Write 实例代码

MEMC 默认使用 **VendorStorage ID=62** 分区，如果客户需要定制化修改，可以通过配置如下ro属性：

```
# 可将 ro.vendor.memc.vsid=100(举例说明) 写入 /vendor/build.prop 文件中
# 指定ID=100作为MEMC激活码存放分区
ro.vendor.memc.vsid
```

规定VendorStorage 长度设置为 50 char

```
// 关键信息: ID=62, Len=50
#define VENDOR_RKNPU_AUTHOR_ID 62
int vendor_storage_write_test (const char* str){
    u32 i;
```

```

int ret, sys_fd;
u8 p_buf [2048]; /* malloc req buffer or used extern buffer */
struct rk_vendor_req *req;
req = (struct rk_vendor_req *) p_buf;
sys_fd = open ("/dev/vendor_storage", O_RDWR, 0);
if (sys_fd < 0){
    printf("vendor_storage open fail\n");
    return -1;
}
req->tag = VENDOR_REQ_TAG;
req->id = VENDOR_RKNPU_AUTHOR_ID;
req->len = 50; /* data len */

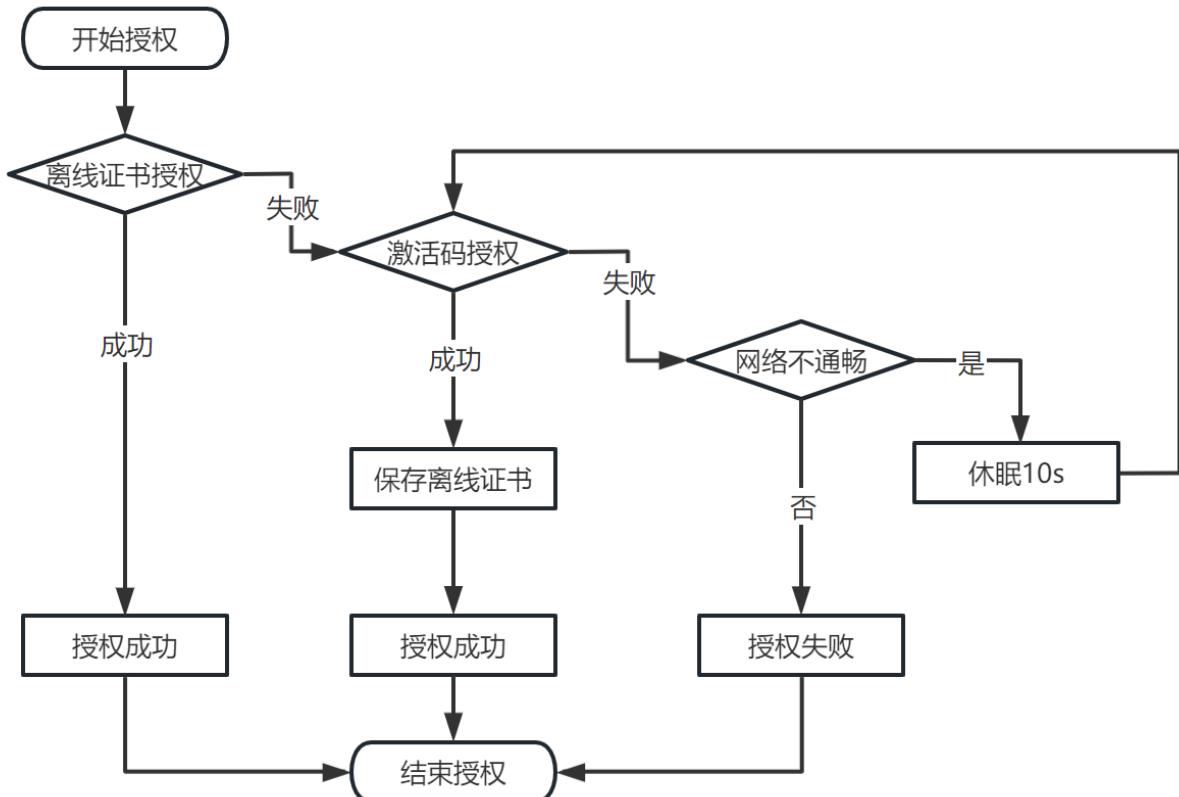
for (i = 0; i < strlen(str); i++)
    req->data [i] = str[i];

print_hex_data ("vendor write:", req, req->len);
ret = ioctl (sys_fd, VENDOR_WRITE_IO, req);
if (ret){
    printf ("vendor write error\n");
    return -1;
}
return 0;
}

```

2.5.3 MEMC 授权流程

算法授权实现流程图如下：



1. 离线证书授权：初次授权，由于离线证书不存在，则离线证书授权失败，进入激活码授权流程；

2. 激活码授权：获取VendorStorage ID=62 激活码，进行网络授权：
 - 若由于网络原因导致授权失败，则休眠10s再尝试激活码授权；
 - 若激活码无效则直接判断授权失败；
3. 初次授权成功后，系统会将离线授权证书/**/data/system/memc_key.lic**位置，用于下次离线授权；
4. 授权成功；
5. 下次授权直接使用离线授权证书进行授权，离线授权证书为授权激活码与设备识别码生成的唯一加密码，仅可在授权设备上可用；

注意：

部分设备的CPU序列号为0，即未正确设置OTP的芯片，此类芯片可能在授权的过程中存在问题，建议更换有正常的序列号设备进行授权。

2.6 OSD 字幕接口

2.6.1 OSD修改方法介绍

1. 按如下补丁修改：

```
hardware/rockchip/libssvep$ git diff
diff --git a/include/memc/MemcType.h b/include/memc/MemcType.h
index 382e602..a58143a 100755
--- a/include/memc/MemcType.h
+++ b/include/memc/MemcType.h
@@ -41,7 +41,7 @@ namespace android {
 // Vendor Storage ID.
#define MEMC_VENDOR_AUTHOR_ID "ro.vendor.memc.vsid"
// OSD string interface.
-#define MEMC OSD VIDEO STR L"RKNPU-SVEP-MEMC"
+#define MEMC OSD VIDEO STR L"RKNPU-SVEP-MEMC-TEST"

enum Memc_Error {
    MEMC_NO_ERROR = 0,
```

2. 重新编译 hardware/rockchip/hwcomposer/drmhw2 目录：

```
mmm hardware/rockchip/hwcomposer/drmhw2 -j4
```

3. 更新 hwcomposer.rk30board.so 文件，即可修改字幕：

2.6.2 字幕限制信息

长度：自定义OSD对长度存在限制，限制信息如下：

字体类型	大写英文字母	小写英文字母	简体中文	阿拉伯数字
最大支持长度	19	22	14	29

其他：目前仅支持对第一行OSD字幕进行修改，剩余 Input/Output 信息无法修改

2.6.3 关闭OSD字幕

属性名称	缺省值	值范围	值说明
persist.sys.svep.disable_memc_osd	0	0 / 1	0：使能OSD字幕 1：关闭OSD字幕

2.6.4 OSD-Oneline 模式

OSD-Oneline模式即仅保留一行字幕，考虑到部分场景OSD字幕可能会遮挡视频，影响客户观看体验，故提供OSD-Oneline模式减少OSD字幕。相关属性配置如下：

属性名称	默认值	值范围	值说明
persist.sys.memc.enable_oneline_osd	0	0-1	0: 关闭 1: 开启
persist.sys.svep.oneline_osd_wait_second	12	> 0	正常OSD切换到Oneline模式需要等待的秒数

可通过以下命令设置OSD-Oneline模式：

```
## 开启OSD-Oneline模式，并设置5s后进入online模式
adb shell setprop persist.sys.svep.enable_oneline_osd 1
adb shell setprop persist.sys.svep.oneline_osd_wait_second 5

## 关闭OSD-Oneline模式
adb shell setprop persist.sys.svep.enable_oneline_osd 0
```

效果展示：配置情况如下：

- persist.sys.svep.enable_oneline_osd=1
- persist.sys.svep.oneline_osd_wait_second=12

1. 播放视频OSD，播放时间 < 12s，OSD显示为正常的三行字幕，如下图：



2. 视频播放时间 > 12s，OSD切换为单行字幕，如下图：



OSD-Oneline字幕自定义方法：

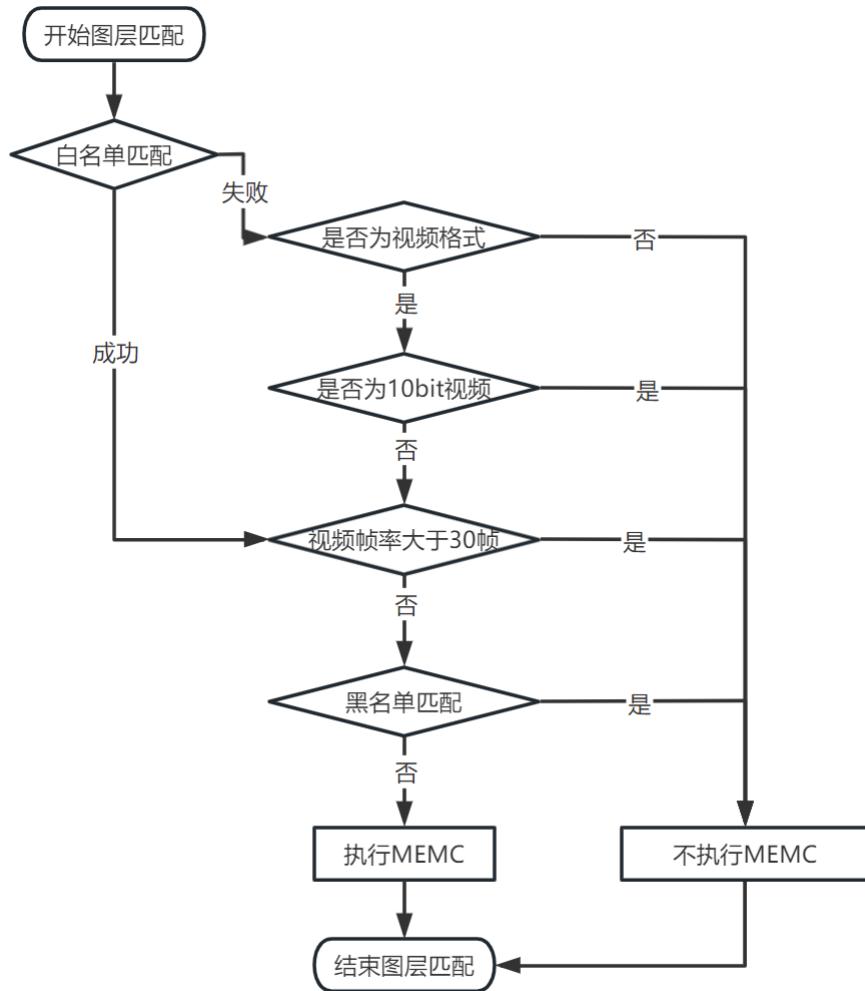
```
hardware/rockchip/libssvep$ git diff
diff --git a/include/SvepType.h b/include/svepType.h
index 5d9c0b1..5bec76c 100644
--- a/include/SvepType.h
+++ b/include/SvepType.h
@@ -43,7 +43,7 @@ namespace android {
 // OSD string interface.
 #define SVEP OSD_VIDEO_STR L"RKNPU-AI视频增强"
 // one line OSD
-#define SVEP OSD_VIDEO_ONELINE_STR L"AI"
+#define SVEP OSD_VIDEO_ONELINE_STR L"AI-OnelineMode"
```

2.7 MEMC 使能流程

目前MEMC图层匹配流程集成在HardwareComposer服务内部，主要针对刷新率为30帧的视频图层进行MEMC处理，不建议对UI图层进行MEMC操作，对应的处理逻辑如“2.7.1 图层匹配逻辑”介绍。

2.7.1 通用图层匹配逻辑

通用图层匹配逻辑如下图，该逻辑目前集成在Android HardwareComposer服务内部：



1. 白名单匹配成功，则跳过视频格式判断，直接进行帧率与黑名单判断逻辑；
2. 图层匹配默认匹配视频图层，且不支持10bit片源，HDR片源；
3. 理论上帧率大于30帧不执行MEMC操作；

```
# 图层刷新率可通过以下命令查看:  
adb shell dumpsys SurfaceFlinger  
# 查看 HWC 对应图层的 mFps 参数，如下图:
```

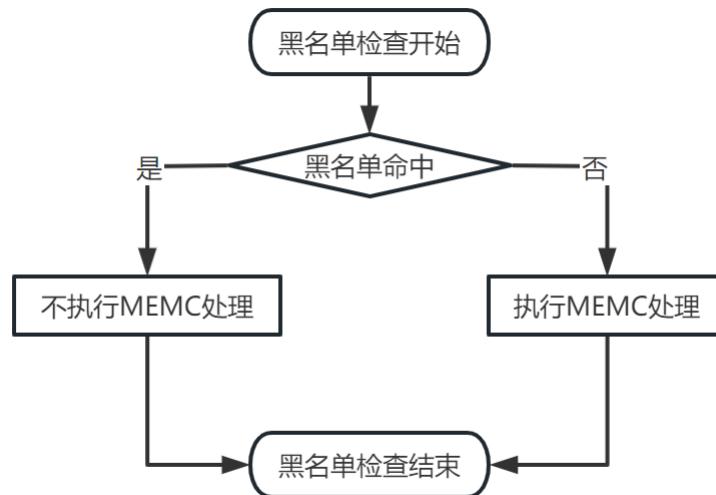
DrmHwcLayer Dump:											
id	z	sf-type	hwc-type	handle	transform	blind	source crop (l,t,r,b)	frame	dataspace	mfps	name
0022	1000	Device	Device	00b40000741101da10	None	None	0.0, 0.0, 864.0, 486.0	0, 0, 1920, 1080	10010000	29.8	surfaceView[com.ktcp.launcher/com.ktcp.videoDetail.coverActivity]#9(BLAST CONSUMER)
0012	1001	Device	Device	00b40000741102e930	None	Premultipli	0.0, 0.0, 1920.0, 1080.0	0, 0, 1920, 1080	0	0.2	viewRootImpl[DetailCoverActivity]#3(BLAST CONSUMER)

4. 黑名单匹配图层不执行MEMC操作；

2.7.2 黑名单处理流程说明

部分应用可能存在无需对视频进行MEMC处理的场景，如要求低延迟的云游戏场景，故提供 /vendor/etc/HwcSvepMemcEnv.xml 配置，允许客户将应用图层名加入黑名单，系统MEMC处理流程检测黑名单上的图层后，不进行MEMC处理增强。

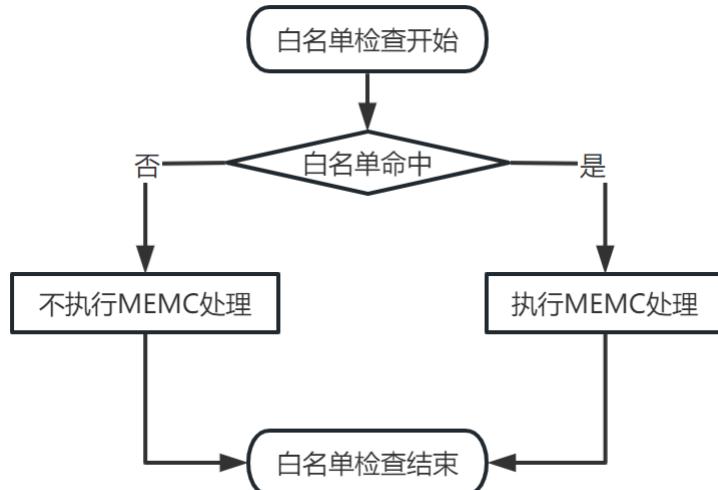
黑名单配置具体处理流程如下图：



2.7.3 白名单处理流程说明

部分客户应用可能不满足通用图层匹配逻辑中的视频格式逻辑，故提供白名单，绕过这部分限制。

白名单配置具体处理流程如下图：



2.7.3 配置文件说明

参考文件目录位于以下目录：

```
# SDK工程路径:  
SDK_Project/hardware/rockchip/hwcomposer/drmhwc2/res/HwcSvepMemcEnv.xml  
  
# 设备路径:  
/vendor/etc/HwcSvepMemcEnv.xml
```

XML文件格式说明如下：

```
<?xml version="1.0" encoding="utf-8"?>  
<!-- Svep xml -->  
<HwcSvepEnv Version="1.1.1" >  
  
<Blacklist> <!-- 黑名单头部结构 -->  
    <!-- 黑名单的关键词，通常为应用图层名字 -->  
    <BlackKeywords>  
        SurfaceView[android.rk.RockvideoPlayer/android.rk.RockvideoPlayer.VideoPlayActivity]</BlackKeywords>  
    <!-- 可添加多个并列 BlackKeywords -->  
    <BlackKeywords> RockvideoPlayer</BlackKeywords>  
    <!-- 可添加多个 BlackKeywords -->  
    <BlackKeywords> android.rk.RockvideoPlayer.videoPlayActivity</BlackKeywords>  
</Blacklist>  
  
<whitelist>  
    <!-- 爱奇艺：标准、Lite、巴布版本适用如下白名单 -->  
    <whiteKeywords>SurfaceView[com.qiyi.video]</whiteKeywords>  
    <!-- 优酷视频 -->  
    <whiteKeywords>SurfaceView[com.youku.phone]</whiteKeywords>  
    <!-- 哔哩哔哩 -->  
    <whiteKeywords>SurfaceView[tv.danmaku.bili]</whiteKeywords>  
    <!-- 虎牙直播 -->  
    <whiteKeywords>SurfaceView[com.duowan.kiwi]</whiteKeywords>  
    <!-- 腾讯课堂 -->  
    <whiteKeywords>ViewRootImpl[RecruitAvActivity]</whiteKeywords>  
    <whiteKeywords>ViewRootImpl[FCourseDetailActivity]</whiteKeywords>  
    <whiteKeywords>ViewRootImpl[webOpenUrlActivity]</whiteKeywords>  
    <!-- 掌门一对一辅导 -->  
    <whiteKeywords>ViewRootImpl[RecordedLessonDetailActivity]</whiteKeywords>  
</whitelist>  
</HwcSvepEnv>
```

应用图层名字获取：

1. 在目标场景按如下命令抓打印日志：

```
adb shell dumpsys SurfaceFlinger
```

2. 输出日志文件的如下红框部分则为目标应用图层名字，如下红框部分：

```

Planner is disabled
h/w composer state:
h/w composer enabled
-- HWC2 Version HWC2-1.3.40 by bin.li@rock-chips.com --
DisplayId=0, Connector 431, Type = DSI-1, Connector state = DRM_MODE_CONNECTED
NumHwLayers=3, activeModeId=16, 1080x1920p60.00, colorMode = 0, bStandardSwitchResolution=0
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | z | sf-type | hwc-type | handle | transform | bind | source crop (l,t,r,b) | frame | dataspace | mfps | name |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
0033 | 000 | Device | Device | 00b400007b560bfa70 | None | None | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 1080, 1920 | 10001 | 20.8 | SurfaceView[android.rk.RockVideoPlayer/android.rk.RockVideoPlayer.VideoPlayActivity]#8(BLAST Consumer)8
0006 | 001 | Client | Client | 00b400007b560bdaf0 | None | Premultipl | 0.0, 0.0, 1080.0, 48.0 | 0, 0, 1080, 48 | 0 | 0.0 | ViewRootImpl[ScreenDecorOverlay]#0(BLAST Consumer)
0005 | 002 | Client | Client | 00b400007b560b4710 | None | Premultipl | 0.0, 0.0, 1080.0, 48.0 | 0, 1872, 1080, 1920 | 0 | 0.0 | ViewRootImpl[ScreenDecorOverlayBottom]#2(BLAST Consumer)
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

3. 完整应用名称为如下字符串：

```

SurfaceView[android.rk.RockvideoPlayer/android.rk.RockVideoPlayer.VideoPlayActivity]#8(BLAST Consumer)8
# 其中,
# "#"后的字符串"8(BLAST Consumer)8" 为Android系统自动添加, 不建议作为识别目标图层的字符串
# "#"以前的字符串可作为目标图层的关键字符串

```

注意：只要应用名称包含 BlackKeywords，则认为黑名单图层命中，则不进行MEMC插帧处理，采用直通模式显示；

例如： android.rk.RockVideoPlayer 或 android.rk.RockVideoPlayer.VideoPlayActivity 均可对完整 SurfaceView[android.rk.RockVideoPlayer/android.rk.RockVideoPlayer.VideoPlayActivity] 应用图层名进行黑名单命中；

三、FAQ

3.1 简单介绍MEMC具体做了哪些处理？

算法利用深度神经网络从前后两帧原始帧计算出中间的预测帧，提高视频帧率以获得较原始视频更流畅的感官体验。示意图请参考“概述”章节。

3.2 如何确认MEMC授权成功？

MEMC模块支持需要申请授权账号并且授权与硬件SOC绑定，一块SOC授权将占用一个授权名额，请谨慎授权；

1. 将授权激活码写入 /dev/vendor_storage 节点，具体可参考《Rockchip_Application_Notes_Storage_CN.pdf》文档自行开发烧写激活码工具，下面示例为自行编写的 vendor-storage-test 用例进行烧写激活码：

```
rk3588_box:/ # vendor-storage-test 9106BC0B-BCB6-17CE-9A3E-DD4C462B356F
## vendor write:
##   tag=0x56524551 id=62 len=50
##   9106BC0B-BCB6-17CE-9A3E-DD4C462B356F
## vendor read:
##   tag=0x56524551 id=62 len=50
##   9106BC0B-BCB6-17CE-9A3E-DD4C462B356F
```

2. 重新开机后，确认联网，可通过以下日志查看授权情况：

```
## 查看是否存在以下日志:
adb shell logcat -s libsvpmemc-auth

## 初次授权成功日志
I libsvpmemc-auth: ReadAuthCode, line=430 MEMC-AuthCode: tag=0x56524551 id=62 len=50 Code:
2632597B-76F9-345A-5244-6AAA28211261
I libsvpmemc-auth: DoAuth, line=347 rkauth_activate2 auth code success, write license to
/data/system/memc_key.lic
## 确认 “rkauth_activate2 auth code success” 日志，即表示授权成功，并将离线授权文件
## 保存到 /data/system/memc_key.lic位置

## 再次授权成功日志，使用 VendorStorage ID=63 离线授权证书授权成功
I libsvpmemc-auth: DoAuth, line=327 rkauth_verify_license /data/system/memc_key.lic
Success.
```

3. 使能 MEMC 模式，播放待测片源，若MEMC运行正常，则片源左上角会出现 “RKNPU-SVEP-MEMC” OSD字样：

```
## 利用以下命令使能 MEMC 模式
adb shell setprop persist.sys.memc.mode 1

## 若MEMC正常初始化，则通过以下命令可查询MEMC版本信息
adb shell getprop vendor.memc.version
```

注意：部分客户由于特殊的产品需求可能会将HWC功能关闭，导致无法正确进入memc模式，可检查以下属性是否正确：

```
## 确认 vendor.hwc.compose_policy 属性值为 1  
adb shell getprop vendor.hwc.compose_policy  
  
## 若为其他值，请将属性设置为1  
adb shell setprop vendor.hwc.compose_policy 1
```

3.3 如何判断是否正确集成并开启MEMC功能？

1. 确认MEMC授权成功，参照 "FAQ 3.2 如何确认MEMC授权成功？" 章节确认授权成功
2. 若授权检查成功，直接播放本地视频或者网络视频，MEMC正确运行的正常，即可在视频左上角发现"RKNPU-SVEP-MEMC" OSD字样，并且包含输入输出尺寸大小，如下图：



3. 确认MEMC模式是否开启，通过 persist.sys.memc.mode 属性检查，确保设置值为1

```
## 检查MEMC模式是否为1  
adb shell getprop persist.sys.memc.mode  
  
## 若不为1，则通过以下命令设置为1  
adb shell setprop persist.sys.memc.mode 1
```

4. 若上述检查均正常，请按以下要求抓打印日志，提供RK工程师分析：

```
adb shell setprop vendor.hwc.log debug  
adb shell setprop vendor.memc.log 1  
adb shell logcat -c  
adb shell logcat > hwc.log
```

3.4 Android SDK如何使能MEMC功能？

SDK工程使能 MEMC 功能请按照如下步骤：

1. MEMC接口库位于SDK以下目录：

```
hardware/rockchip/libsvp
```

2. HWC源码位于SDK以下目录：

```
hardware/rockchip/hwcomposer/drmhwc2
```

3. 使能MEMC编译选项:

```
# 修改以下Android.mk文件  
hardware/rockchip/hwcomposer/drmhwc2/Android.mk  
hardware/rockchip/libsvep/Android.mk  
  
# 添加MEMC编译选项  
BOARD_USES_LIBSVEP_MEMC := true  
  
# 或者, 可在对应product目录下, 添加 MEMC 编译选项, 例如BOX工程可直接修改以下mk文件:  
device/rockchip/rk3588/rk3588_box/rk3588_box.mk  
+ BOARD_USES_LIBSVEP_MEMC := true
```

4. 编译SDK工程;

另外: MEMC功能要正常使用还需要授权, 请向RK获取授权激活码, 请参考"2.5 VendorStorage 授权接口" 章节。