Linux KSC
开发指南

版本号: 1.0
发布日期: 2025.03.07

版本历史

版本号	日期	制/修订人	内容描述
1.0	2025.03.07	AWA1221	1. 创建该文档

目　录

插　图

1 前言

1.1 文档简介

本文主要介绍 KSC 模块方案设计

1.2 目标读者

- KSC 驱动相关开发者
- 应用层的 KSC 相关开发者

1.3 适用范围

表 1-1: 适用产品列表

内核版本	驱动文件
Linux-5.15	ksc_drv.c

1.4 相关术语介绍

表 1-2: 术语

术语	解释说明
KSC	KeyStone Correction
REC	Rectification

2 模块介绍

TV301 的 KSC 支持在线模式和离线模式，在线模式下有以下特性：

1. 数据格式：RGB888，8/10bit
2. 内部采样数据格式:yuv444sp/yuv422sp, 8bit/10bit
3. 支持 quadrangular trapezoid 矫正
4. 最小的 shrink 比例是 0.1 倍
5. 支持最大 8 倍图像放大
6. 支持图像边缘抗锯齿
7. 支持图像锐化
8. 支持 180 度旋转
9. 支持水平/垂直镜像
10. 支持图像 crop
11. 支持 FE-only 模式，用于将处理后的数据写入到 DDR，格式是 yuv44sp/yuv422sp 格式
12. 输入分辨率范围：64x64~2048x2048
13. 输出分辨率范围：64x64~2048x2048
14. 性能：1080p@60fps

离线模式特性：

1. 数据格式：8bit:yuv420sp/yuv422sp/yuv444sp/argb8888. 10bits:yuv420sp
2. 支持 quadrangular trapezoid 矫正 (KSC100)
3. 最小的 shrink 比例是 0.2(KSC100) 倍
4. 支持最大 8 倍图像放大 (KSC100)
5. 支持图像边缘抗锯齿 (KSC100)
6. 支持图像锐化 (KSC100)
7. 支持 180 度旋转 (KSC100)
8. 支持水平/垂直镜像 (KSC100)
9. 支持图像 crop(KSC100)
10. 支持将图像数据写到指定内存区域。(KSC100)
11. 输入分辨率范围：64x64~4096x4096
12. 输出分辨率范围：64x64~4096x4096
13. 性能：1080p@60fps
14. 任意角度旋转 (KSC110)

2.1 原理

KSC 梯形矫正功能用于当投影机与屏幕不垂直导致画面产生偏移的时候，对图像进行矫正使画面调整成矩形。

在投影仪产品中，由于投影仪安装位置通常难以严格正对投影面，导致投影画面与光轴有一定的倾斜角，使得方形的图像画面被投影成梯形，影响观看体验。

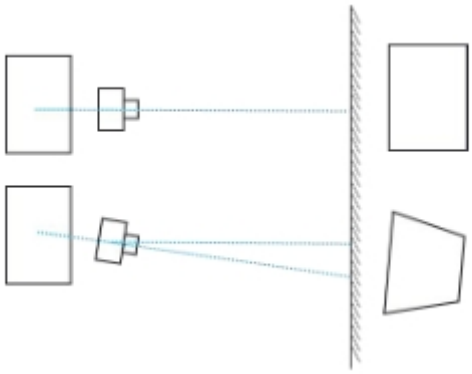


图 2-1: 投影 1

为了解决该问题，常用做法是采用梯形矫正算法对投影画面进行矫正处理，在成像端将投影图像矫正为原投影图像的逆向梯形图，从而抵消掉透视投影原本产生的梯形投影效果，使得最终的投影画面满足观看需求。

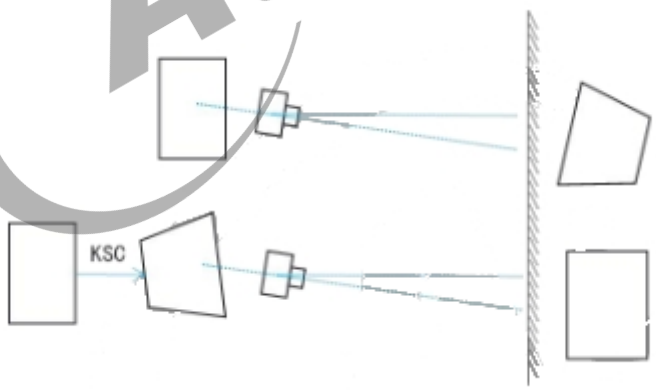


图 2-2: 投影 2

2.2 工作模式

KSC IP 总共支持三种工作模式:

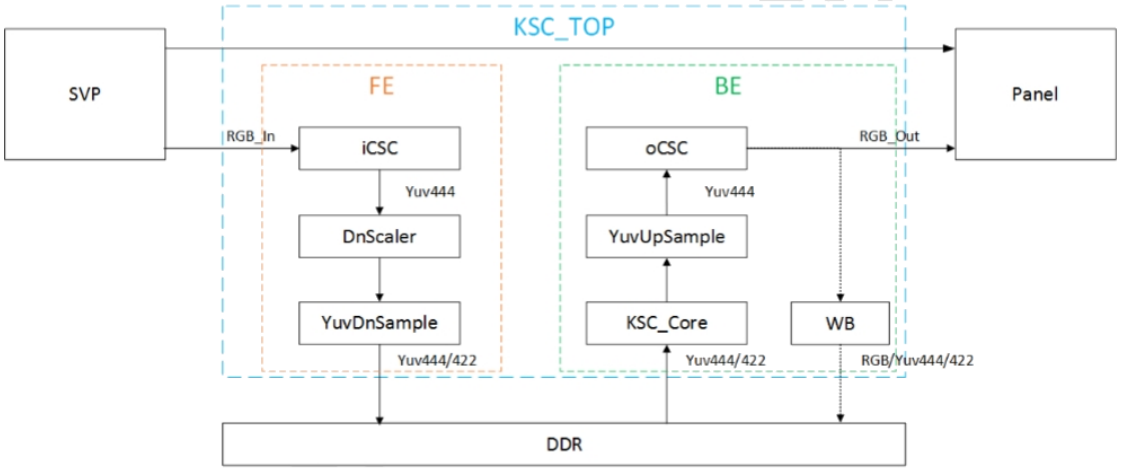


Fig1-1. KSC100-Online mode function block diagram

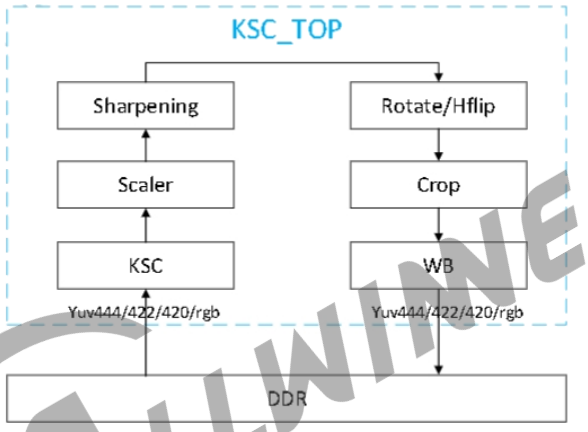


图 2-3: 在线模式和离线模式 2

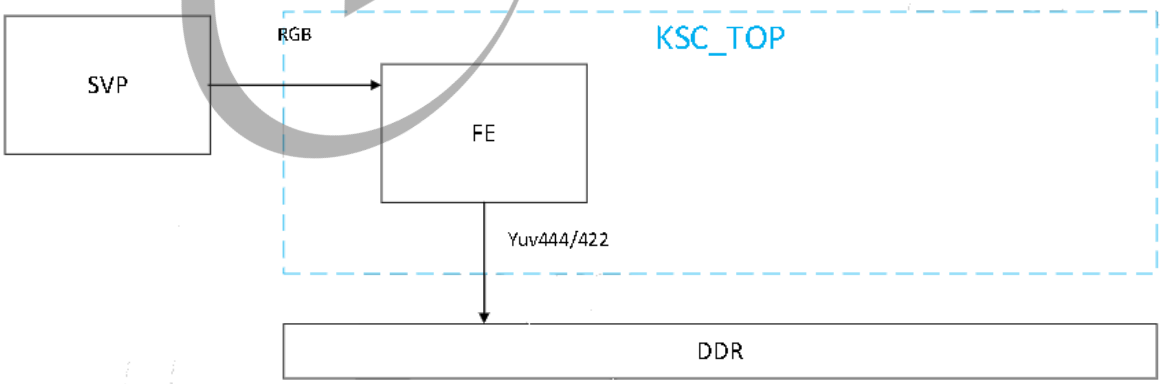


图 2-4: 离线模式 1

1. 在线模式。在线模式下，KSC ip 位于图像合成 IP 与 Panel 之间，内部划分为 BE 和 FE 两部分，FE 在线将上一级 IP 的输出的 RGB 数据经过预处理回写到 DDR，BE 从 DDR 中读取预处理的数据进行梯形矫正，最后保持相同格式送显。
2. 离线模式 1。该模式和在线模式的区别就是关闭 BE 不送显，然后将 FE 处理结果回写到 DDR，可以称之为半在线模式。

3. 离线模式 2。这个是真正的离线模式，不依赖于图像合成 IP 与 Panel。KSC 从 DDR 读取图像经过一系列处理再以相同格式写回到 DDR。

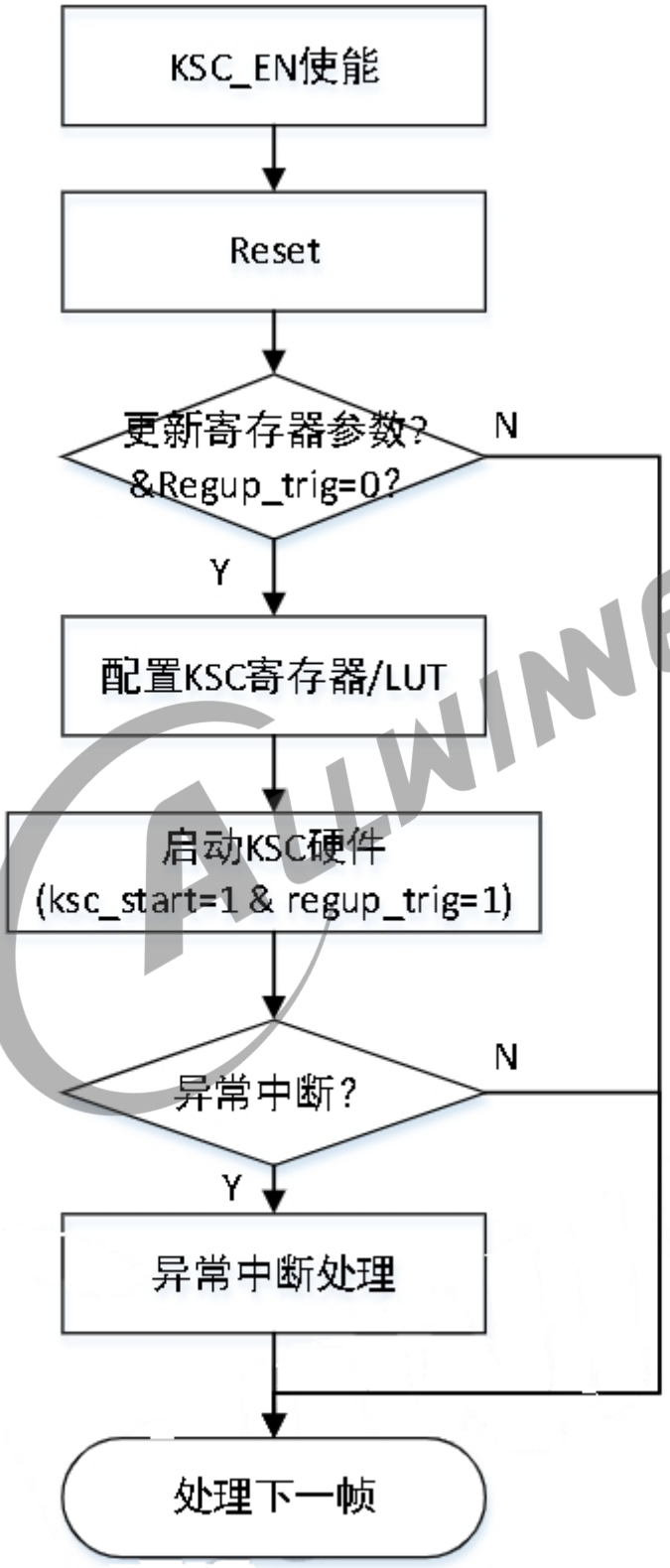


图 2-5: 在线模式工作流程

简单的说：更新寄存器前需要查询 0x10 寄存器的 KSC_REG_TRIG 寄存器，如果是 0 则表示空闲可更新，否则需要等待。更新寄存器后，需要将 KSC_REG_TRIG 置 1。KSC 启动后，将由 vsync 自动触发。

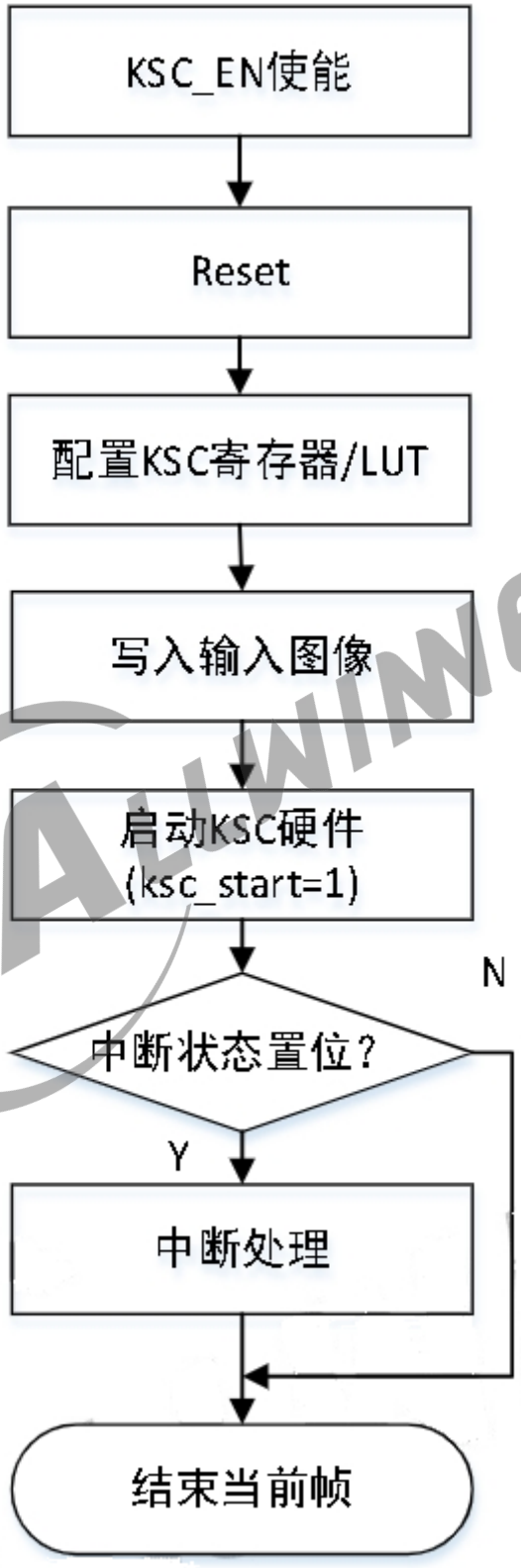


图 2-6: 离线模式 2 工作流程

简单的说，软件每次配置 KSC_START=1，硬件就处理一帧，处理完之后就停止。

在线模式下，KSC 位于 SVP 和 Panel 之间，内部划分为 FE 和 BE 两个部分，FE 将 SVP 的输出的 RGB 数据经过预处理回写到 DDR 中，BE 从 DDR 读取经 FE 预处理后的数据进行梯形矫正，最后保持与输入相同的 RGB 格式送显。

离线模式下，该模式将任意数据经过 FE 回写到 DDR，再经过 BE 进行梯形矫正，最后送显示。

另外还有一种模式，介于在线与离线模式之间，这个是 be 不工作，fe 将处理结果写到 DDR 中，该模式主要用于调试，无实际应用场景。

2.3 内存操作

ksc IP 即使是在线模式也需要读写 DDR，其中包含以下作用的内存读写。

1. 图像缓存。总共需要两片图像内存，大小一样。用于做乒乓操作。该乒乓操作由硬件自动完成，软件无需参与，软件只负责申请内存和配置内存地址即可。注意，这两片内存只用于在线模式。
2. 算法内存。Downsacle 系数和 lut 查找表两片内存。如果离线支持梯形矫正，则需要这两片内存。
3. 离线的输入图像和输出图像内存。
4. 在线回写的输出图像内存。

2.4 像素格式

KSC 不同模式处理过程涉及到几种像素格式。

1. FE 模块输出的像素格式。在线的情况，它是一个中间处理过程中的像素格式，不同格式的目的仅在于能否节省带宽，一般是 YUV422。离线模式 1，则是回写到 DDR 文件的像素格式。
2. FE 的输入图像格式。在线的情况下，图像格式依赖于上一级 IP 对图像格式的支持情况。
3. BE 的输出图像格式。在线的情况 FE 的输入图像格式与 BE 的输出图像格式一致。离线模式 2 的情况下则无须保持一致。

2.5 梯形矫正

梯形矫正用户接口就是四个顶点坐标，如下图所示，采用在安卓上广泛采用的坐标来定义梯形矫正，左上顶点为原点。

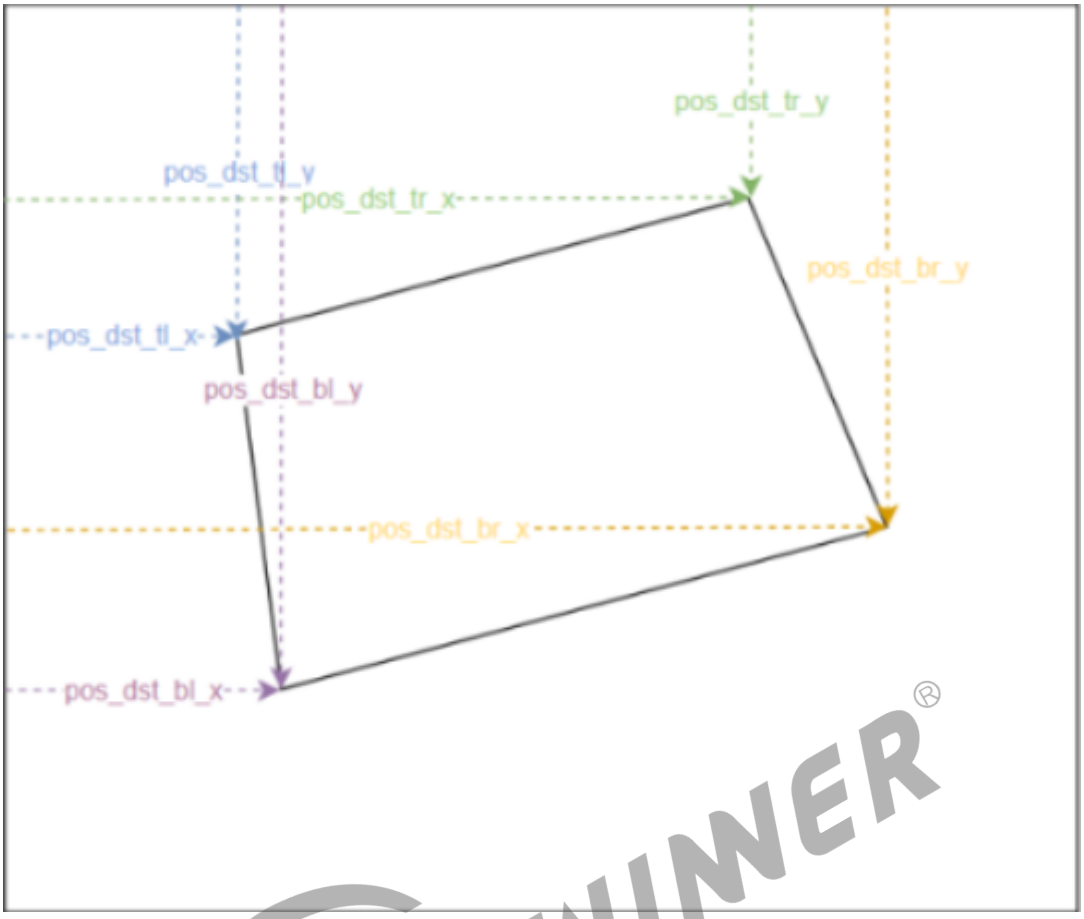


图 2-7: 梯形矫正

图 2-8: 梯形矫正示例

2.6 OST 画面平移

用户需要指定左上角顶点坐标。

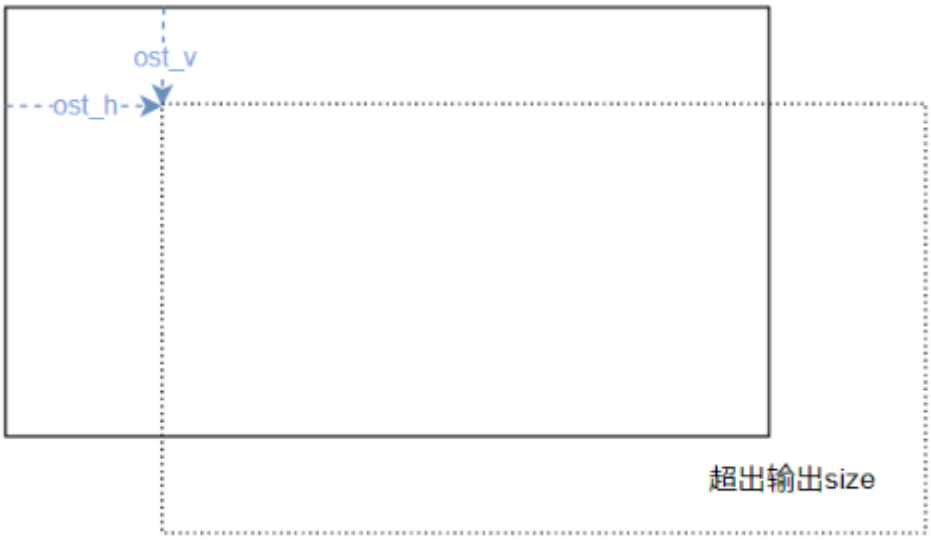


图 2-9: ost 示意

图：

图 2-10: ost 示意图

2.7 虚拟变焦

原图：

图 2-11: 原图

图 2-12: 变焦

2.8 旋转与镜像

图 2-13: 水平镜像

关于旋转，在线只支持 180 度旋转，相当于水平和垂直镜像同时使能。

2.9 任意角度旋转

仅 ksc110 支持。

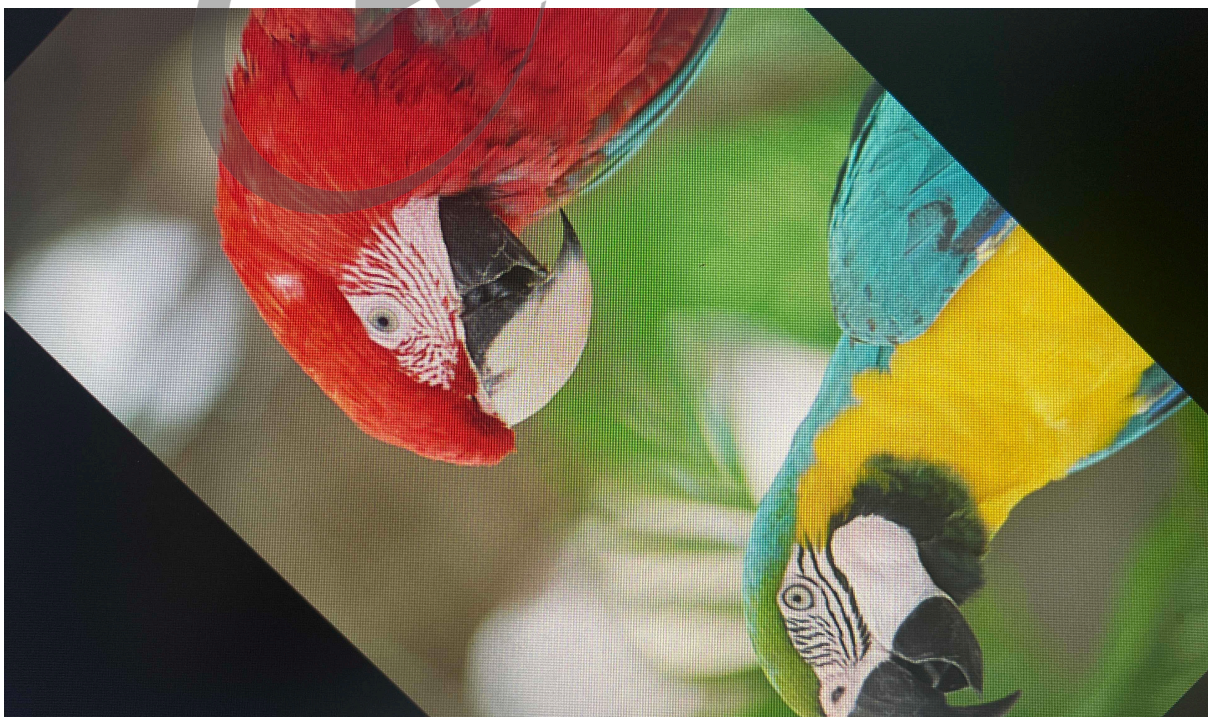


图 2-14: 任意角度旋转

2.10 离线 crop

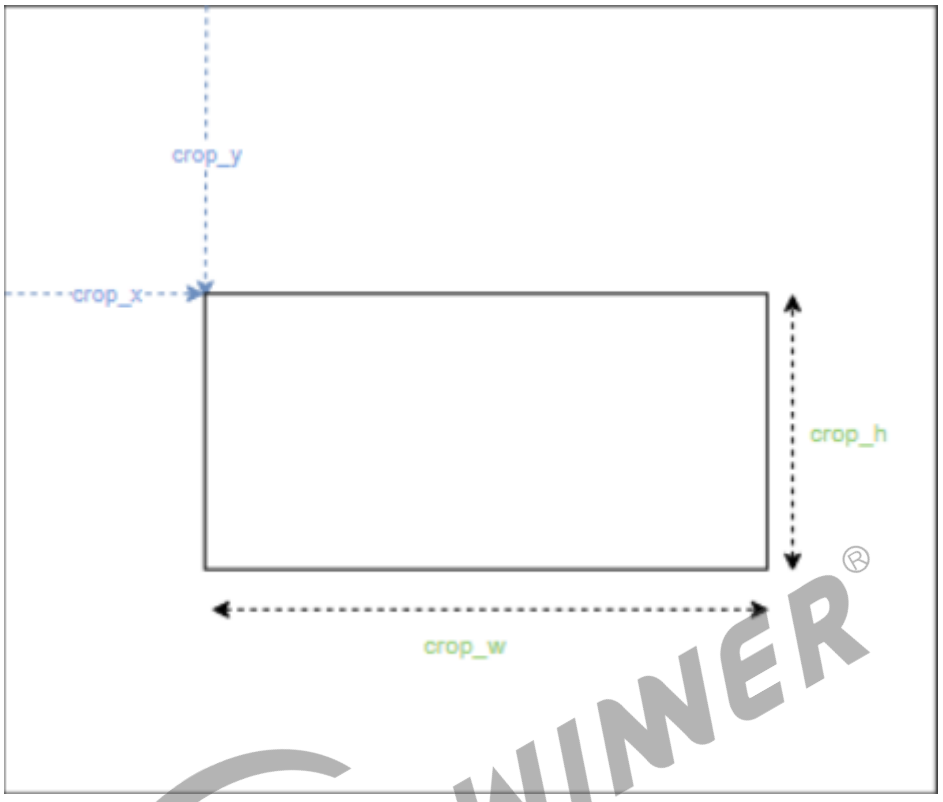


图 2-15: crop

图：

图 2-16: crop 示例

2.11 缩放

FE 模块有缩小功能，而 BE 有放大功能，在在线的情况，由于接口分辨率是一旦确定就是不变的，这个功能通常是为了减少带宽，fe 缩小后，再由 BE 放大回分辨率大小。

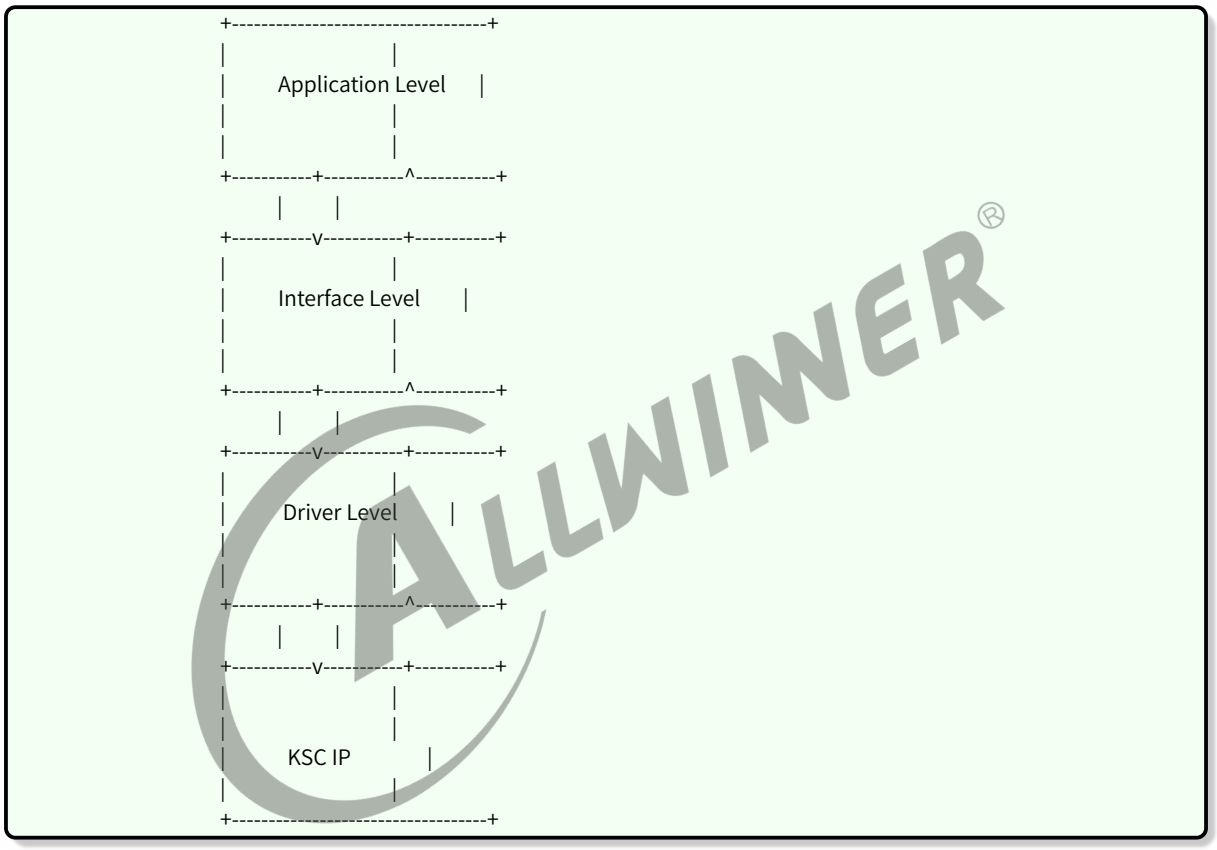
在离线模式下，则可以实现用来缩小或者放大图片，实现真正缩放功能。

说明

仅 ksc100 支持放大。

3 方案设计

由于算法参数是存放在内存中由 IP 读取，然后算法参数涉及到一堆浮点型运算，而 Linux 内核对浮点型支持比较麻烦，基于这个特点，决定将 KSC 分成两层实现。驱动层和接口层。驱动层负责寄存器，中断等处理，不负责具体逻辑，接口层负责提供用户接口然后生成算法系数和寄存器所需要的参数送给驱动层，这里面涉及到浮点型运算。



3.1 内存接口

接口层需要生成算法系数给驱动层，算法系数最终是放在内存中由硬件进行读取。

需要解决的问题是：

1. 内存怎么申请
2. 接口层如何访问该内存又如何将该内存传给驱动层。

传统方法是通过 dma_heap 或者 ion 接口在应用层进行申请，然后传递 fd 给驱动层，缺点是驱动需要进行反复的 dma map 和 unmap，该操作会比较耗时。

另外一个需要面对的问题是，只有驱动知道这块内存什么时候释放比较合适。

从在线模式工作流程我们知道，有一个信号标志着寄存器更新完毕，也就是 reg done 信号，我们在连续改变梯形矫正参数的时候，需要注意的就是后面一次设置不能影响上一帧的显示。

除了寄存器之外，算法内存也是不能在显示过程中被窜改的。

为此软件需要为算法内存申请两份，实现 double buffer，确保后帧不干扰前帧显示，按照下面顺序来进行连续设置。

鉴于上面，决定在驱动内部使用 cma 申请内存，然后应用层通过 mmap 接口访问该内存。

第 1 帧（使用 buf1 地址)–> 第 2 帧，等 regup done ，切换为 buf2 地址，, 设置新寄存器 –> 第 3 帧，等 regup done，切换为 buf1 地址，设置新寄存器

3.2 平滑过渡问题

KSC IP 位于图像合成 IP 之后，显示接口之前，那么显示通路使能流程，理论上就是: 图像合成 IP–>KSC–> 显示接口，如果不符合这个流程，怕引起闪烁问题。

这意味着在应用层起来之前，ksc 就得开始工作，但是驱动又无法进行浮点运算，这个时候就需要让 KSC 工作在一种模块叫做 be bypass 模式，该模式下，BE 的功能全部关闭，算法内存不需要读取，梯形矫正功能也是关闭。

第 1 帧 be bypass （使用 buf1 地址)–> 第 2 帧，等 regup done ，切换为 buf2 地址，, 设置新寄存器 –> 第 3 帧，等 regup done，切换为 buf1 地址，, 设置新寄存器

该操作不需要用户干预，只要内核使能了 ksc 模块，那么就会在显示通路过程中使能 ksc be bypass 模式。

这样做的另外一个原因是：显示接口分辨率只有使能显示通路后才知道，ksc 需要第一时间知道其输入输出分辨率，而嵌入到驱动中是一个比较好的做法。

4 用户接口

用户核心结构体如下：

```
struct ksc_para {
    bool ksc_en;

    //for debug purpose, write back output to DDR
    bool online_wb_en;

    //for bandwidth saving purpose
    bool auto_scale_en;
    /*
       enum ksc_run_mode {
           ONLINE_MODE,
           OFFLINE_MODE1,
           OFFLINE_MODE2,
       };
       */
    enum ksc_run_mode mode;
/*

(0, 0)                                                          x
+-------------------------------------------------------------->
|                                                               |
|                                                               |
|            (tl_x,tl_y)   /-----------------\    (tr_x, tr_y) |
|                        /                     \                |
|                       /                       \               |
|                      /                         \              |
|                     /                           \             |
|                    /                             \            |
|                   /                               \           |
|                  /                                 \          |
|                 /                                   \         |
|                /                                     \        |
|               /                                       \       |
|    (bl_x, bl_y)/---------------------------------------\ (br_x, br_y)
|                                                               |
|                                                               |
v---------------------------------------------------------------+
 y


*/

    //x:0~src_w
    //y:0~src_h
    //top left  coordinates
    int pos_dst_tl_x;
    int pos_dst_tl_y;

    //top right coordinates
    int pos_dst_tr_x;
```

```
    int pos_dst_tr_y;

    //bottom left coordinates
    int pos_dst_bl_x;
    int pos_dst_bl_y;

    //bottom right coordinates
    int pos_dst_br_x;
    int pos_dst_br_y;

    /*
     * Online:size of KSC input and output
     * decided by display's resoulution, read only
     * Offline: input size of KSC
     */
    int src_w;
    int src_h;

    /*
     * Online:output size of FE module of KSC
     * FE module can down scale image for saveing bandwith purpose
     * Offline: output size of KSC
     */
    int dst_w;
    int dst_h;

    /*
     * Digital zoom function
     * percentage:12.5 ~ 800, default is 100 (100%)
     */
    float zoom_h;
    float zoom_v;

    //0:rotate resolution by rot_angle. 1:rotate view by view_angle(offline only).
    unsigned char rot_mode;
    /*
        enum rot_angle {
            ROT_0 = 0x0,
            ROT_90,
            ROT_180,
            ROT_270,
        };
        */
    enum rot_angle rot_angle;
    short view_angle;//[-359,359]

    //Set 1 to enable Horizontal Mirroring
    unsigned char flip_h;
    //Set 1 to enable Vertical Mirroring
    unsigned char flip_v;

    /*
     * Image Translation
+-------------^---------------------+
|             |                     |
|     blank   ost_v                 |
|             |       blank         |
|             v                     |
|<--- ost_h---->+-------------------+-------------+
|             |                     |             |
```

```
|                |                          |                |
|    blank       |      part of image       |                |
|                |                          |                |
|                |                          |                |
+----------------+--------------------------+                |
                 |                                           |
                 |           outside of screen               |
                 |                                           |
                 |                                           |
                 |                                           |
                 +-------------------------------------------+

     */
    short ost_h;//[- w,w]
    short ost_v;//[- h,h]
    short phase_h;//[-15,15]
    short phase_v;//[-15,15]

    //offline  para
    //offline  input and output image info
    struct  ksc_image_t src_img;//offline input
    struct  ksc_image_t dst_img;//offline output

/*
(0,  0)
+-------------------------------------------------------------->                x
|                                                                                |
|                                                                                |
|                                                                                |
|              (crop_x, crop_y)                                                  |
|                   +-------------------------+                                  |
|                   |                         |                                  |
|                   |                         |                                  |
|                   |                         |  crop_h                          |
|                   |                         |                                  |
|                   |                         |                                  |
|                   |                         |                                  |
|                   |                         |                                  |
|                   +-------------------------+                                  |
|                             crop_w                                             |
|                                                                                |
|                                                                                |
|                                                                                |
v--------------------------------------------------------------------------------+
 y

*/

    unsigned char crop_en;
    unsigned short crop_x;
    unsigned short crop_y;
    unsigned short crop_w;
    unsigned short crop_h;

    //picture para
    struct  ksc_pq_para pq_para;
};
```

```
struct  sunxi_ksc_online_para {
    bool enable;
    enum ksc_pix_fmt in_fmt;
```

```
    int w;
    int h;
    int bit_depth;
};
```

4.1 libksc_get_para

```
int libksc_get_para(struct ksc_para *para);
```

在线模式下比较有用，用于获取 be pass 模式下已经设置好的参数（比如输入输出分辨率）。

4.2 libksc_set_para

```
int libksc_set_para(struct ksc_para *para);
```

设置具体的 ksc 参数，请参考上面的说的核心结构体。

4.3 libksc_init

对于使用 ksc 的应用层，必须调用该接口并判断返回值为 0，才能继续其它操作。

```
int libksc_init (void);
```

4.4 libksc_alloc_memory

使用 dma_heap 申请内存接口，方便用户在离线模式下使用。

```
int libksc_alloc_memory(int *fd, unsigned int mem_size);
```

4.5 libksc_free_memory

使用 dma_heap 内存释放接口，方便用户在离线模式下使用。

```
int libksc_free_memory(int fd);
```

4.6 libksc_online_enable

TV323，显示通路是在应用层完成操作的，这时候需要一个接口让在应用层能使能 ksc be bypass 功能。

```
int libksc_online_enable(struct sunxi_ksc_online_para *para);
```

著作权声明

商标声明

免责声明