

# Rockchip DVR&DMS 产品方案使用说明

---

文件标识: RK-SM-YF-398

发布版本: V1.0.1

日期: 2021-03-01

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

## 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 © 2021 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](http://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](mailto:fae@rock-chips.com)

## 前言

## 概述

DVR&DMS 产品方案使用说明。

## 产品版本

平台名称	内核版本
Linux	4.4
Linux	4.19

## 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

日期	版本	作者	修改说明
2021-01-31	V1.0.0	Vicent Chi, Zhihua Wang, Zhichao Yu	初始版本
2021-03-01	V1.0.1	Vicent Chi	添加MIPI+DVP方案描述

## 目录

### Rockchip DVR&DMS 产品方案使用说明

1. 瑞芯微DVR/DMS产品方案说明
  - 1.1 RV1126芯片平台开发DVR/DMS的产品优势
  - 1.2 模拟高清RX芯片选型列表
  - 1.3 RV1126 DVR/DMS产品应用框图
2. 模拟高清RX芯片驱动开发说明
  - 2.1 内核config配置
  - 2.2 内核dts配置
3. 数据流通路说明
  - 3.1 双路方案通路
  - 3.2 通道对应的video格式限制
    - 3.2.1 VICAP通路
    - 3.2.2 ISP通路
  - 3.3 通道对应的video节点枚举
    - 3.3.1 VICAP通路
    - 3.3.2 ISP通路
  - 3.4 通道对应的video采集限制
    - 3.4.1 ISP通路
  - 3.5 通道对应的分辨率查询、视频信号查询
  - 3.6 实时查询热拔插接口
4. rkmedia\_vmix\_vo\_dvr\_test应用说明
  - 4.1 支持8路视频采集、H264编码
  - 4.2 支持8路视频合成显示
  - 4.3 支持8路视频切换为前4路、后4路显示
  - 4.4 支持区域画框
  - 4.5 支持RGN Cover
  - 4.6 支持屏幕OSD
  - 4.7 支持通道显示、隐藏
  - 4.8 支持通道的区域亮度获取

# 1. 瑞芯微DVR/DMS产品方案说明

RV1126芯片有两路MIPI接口以及一路DVP接口，另外提供强大的编码性能最高支持8路1080@15fps同时编码，内置2T算力NPU，因此非常适合开发DVR/DMS产品。

## 1.1 RV1126芯片平台开发DVR/DMS的产品优势

- 支持最高8路1080P模拟高清视频输入；
- 强大的AI处理能力，能够支持DMS+ADAS算法同时运行；
- 强大的编码能力，最高支持8路1080P@15fps同时编码；
- 支持8路视频OSD叠加；
- 支持8路视频分屏显示Demo；

## 1.2 模拟高清RX芯片选型列表

目前RV1126平台已经适配了比较多的模拟高清RX芯片，并已经在SDK中集成了这些芯片的驱动，可以通过下表选择：

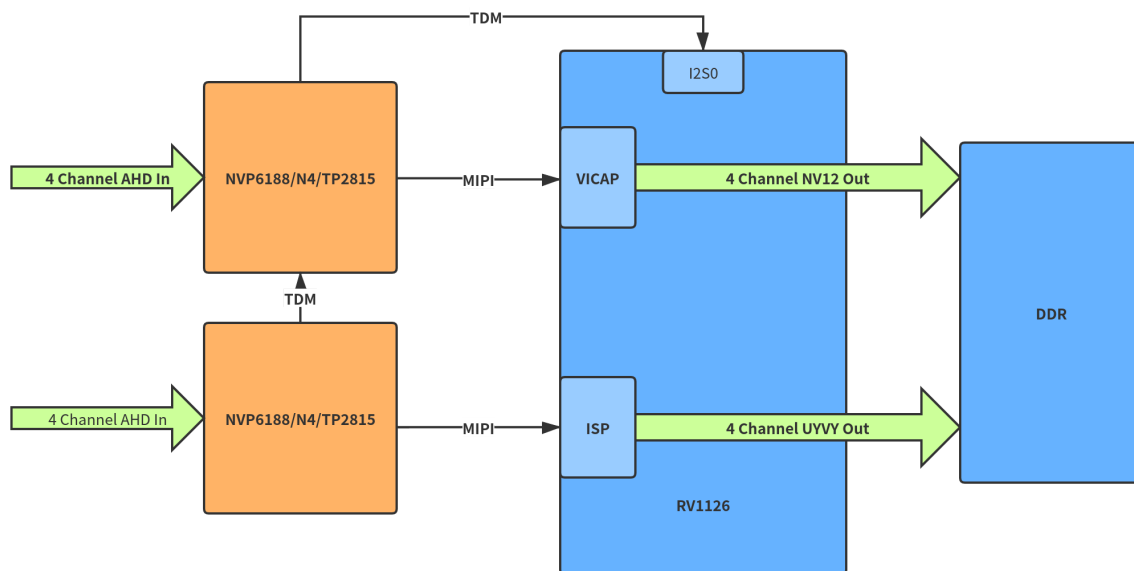
型号	厂家	接口	通道数	最大支持分辨率
NVP6188	Nextchip	MIPI	4	4K
N4	Nextchip	MIPI	4	1080P
NVP6158C	Nextchip	DVP	4	2K
TP2815	Techpoint	MIPI	4	1080P
TP2855	Techpoint	MIPI	4	1080P
TP9930	Techpoint	DVP	4	2K

## 1.3 RV1126 DVR/DMS产品应用框图

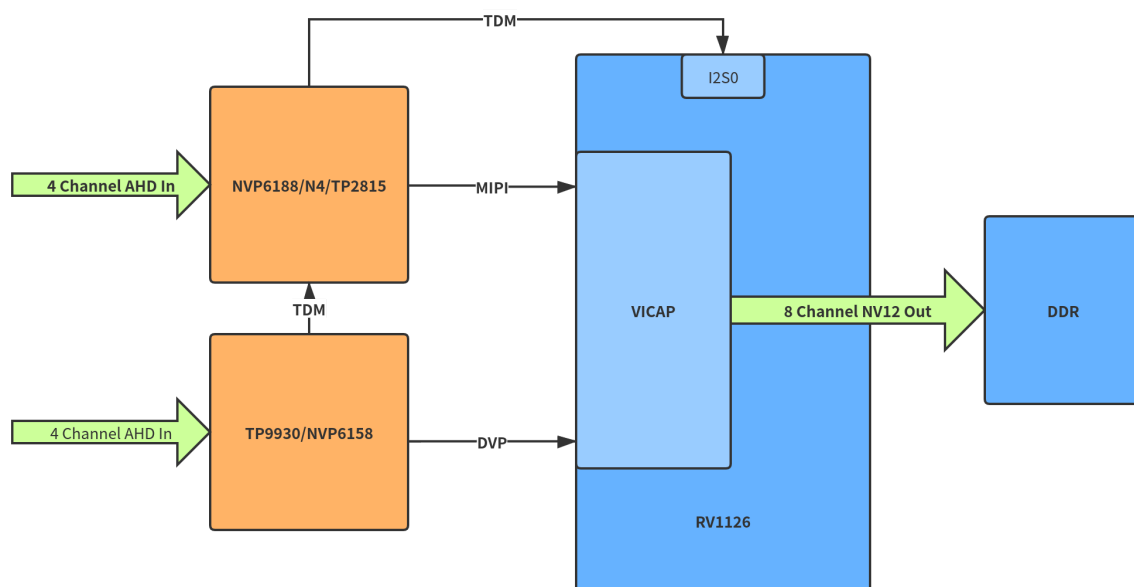
由于RV1126芯片限制，如果选择两个MIPI接口RX芯片输入，其中4路的格式会被限制成UYVY格式，这会导致软件处理带宽增加。因此我们推荐客户选择MIPI+DVP的输入方案，这种方案有如下优势：

- 8路YUV视频可以统一为NV16格式，方便应用层处理；
- DVP RX芯片相比MIPI接口RX芯片价格更便宜，可以节省方案成本；

MIPI+MIPI输入的方案框图如下：



MIPI+DVP输入的方案框图如下：



## 2. 模拟高清RX芯片驱动开发说明

### 2.1 内核config配置

根据需求打开RX芯片相关config配置：

```
1
2 CONFIG_VIDEO_NVP6188=y
3
```

## 2.2 内核dts配置

以NVP6188 RX芯片为例:

```
1
2     nvp6188_0: nvp6188_0@30 {
3         compatible = "nvp6188";
4         reg = <0x30>;
5         clocks = <&cru CLK_MIPICSI_OUT>;
6         clock-names = "xvclk";
7         power-domains = <&power RV1126_PD_VI>;
8         pinctrl-names = "rockchip,camera_default";
9         pinctrl-0 = <&mipicsi_clk0>;
10        reset-gpios = <&gpio4 RK_PA0 GPIO_ACTIVE_HIGH>;
11        power-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
12        vi-gpios = <&gpio3 RK_PC0 GPIO_ACTIVE_HIGH>;
13        rockchip,camera-module-index = <0>;
14        rockchip,camera-module-facing = "front";
15        rockchip,camera-module-name = "nvp6188";
16        rockchip,camera-module-lens-name = "nvp6188";
17        port {
18            ucaml_out0: endpoint {
19                remote-endpoint = <&mipi_in_ucam0>;
20                data-lanes = <1 2 3 4>;
21            };
22        };
23    };
24
25    nvp6188_1: nvp6188_1@32 {
26        compatible = "nvp6188";
27        reg = <0x32>;
28        clocks = <&cru CLK_MIPICSI_OUT>;
29        clock-names = "xvclk";
30        power-domains = <&power RV1126_PD_VI>;
31        pinctrl-names = "rockchip,camera_default";
32        pinctrl-0 = <&mipicsi_clk1>;
33        reset-gpios = <&gpio4 RK_PA1 GPIO_ACTIVE_HIGH>;
34        vi-gpios = <&gpio3 RK_PC1 GPIO_ACTIVE_HIGH>;
35        rockchip,camera-module-index = <1>;
36        rockchip,camera-module-facing = "back";
37        rockchip,camera-module-name = "nvp6188";
38        rockchip,camera-module-lens-name = "nvp6188";
39        port {
40            ucaml_out1: endpoint {
41                remote-endpoint = <&csi_dphy1_input>;
42                data-lanes = <1 2 3 4>;
43            };
44        };
45    };
46
```

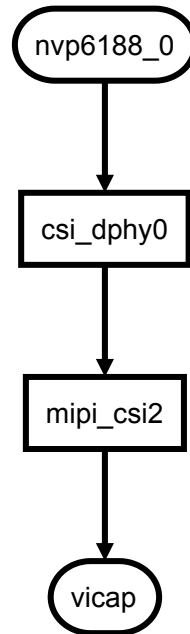
## 3. 数据流通路说明

### 3.1 双路方案通路

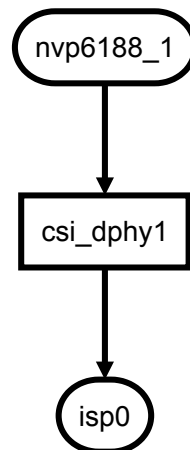
#### 双MIPI方案

以双片NVP6188为例:

- VICAP通路0



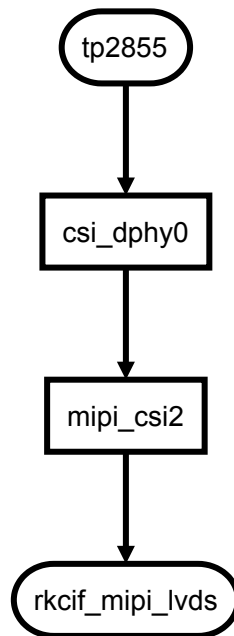
- ISP通路1



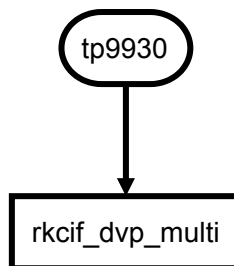
#### MIPI+DVP方案

以TP9930+TP2855为例:

- VICAP通路0



- VICAP通路1



## 3.2 通道对应的video格式限制

### 3.2.1 VICAP通路

- 要统一使用NV16格式采集

### 3.2.2 ISP通路

- 要统一使用UYVY格式采集

## 3.3 通道对应的video节点枚举

### 3.3.1 VICAP通路

- `media-ctl -p -d /dev/mediaX` 获取 `stream_cif_mipi_id0/1/2/3` 的 device node name

```
1 |  
2 | [root@RV1126_RV1109:/]# media-ctl -p -d /dev/media0  
3 | Media controller API version 4.19.111  
4 |
```



```

5 Media device information
6 -----
7 driver          rkcif
8 model           rkcif_mipi_lvds
9 serial
10 bus info
11 hw revision     0x0
12 driver version  4.19.111
13
14 Device topology
15 - entity 1: stream_cif_mipi_id0 (1 pad, 4 links)
16     type Node subtype V4L flags 0
17     device node name /dev/video0
18     pad0: Sink
19         <- "rockchip-mipi-csi2":1 [ENABLED]
20         <- "rockchip-mipi-csi2":2 []
21         <- "rockchip-mipi-csi2":3 []
22         <- "rockchip-mipi-csi2":4 []
23
24 - entity 5: stream_cif_mipi_id1 (1 pad, 4 links)
25     type Node subtype V4L flags 0
26     device node name /dev/video1
27     pad0: Sink
28         <- "rockchip-mipi-csi2":1 []
29         <- "rockchip-mipi-csi2":2 [ENABLED]
30         <- "rockchip-mipi-csi2":3 []
31         <- "rockchip-mipi-csi2":4 []
32
33 - entity 9: stream_cif_mipi_id2 (1 pad, 4 links)
34     type Node subtype V4L flags 0
35     device node name /dev/video2
36     pad0: Sink
37         <- "rockchip-mipi-csi2":1 []
38         <- "rockchip-mipi-csi2":2 []
39         <- "rockchip-mipi-csi2":3 [ENABLED]
40         <- "rockchip-mipi-csi2":4 []
41
42 - entity 13: stream_cif_mipi_id3 (1 pad, 4 links)
43     type Node subtype V4L flags 0
44     device node name /dev/video3
45     pad0: Sink
46         <- "rockchip-mipi-csi2":1 []
47         <- "rockchip-mipi-csi2":2 []
48         <- "rockchip-mipi-csi2":3 []
49         <- "rockchip-mipi-csi2":4 [ENABLED]
50

```

### 3.3.2 ISP通路

- media-ctl -p -d /dev/mediaX 获取 rkisp\_mainpath、rkisp\_rawwr0/1/2 的 device node name

```

1
2 media-ctl -p -d /dev/media1
3 Media controller API version 4.19.111
4

```

```

5  Media device information
6  -----
7  driver          rkisp
8  model           rkisp0
9  serial
10 bus info
11 hw revision      0x0
12 driver version   4.19.111
13
14 Device topology
15 - entity 17: rkisp_mainpath (1 pad, 1 link)
16     type Node subtype V4L flags 0
17     device node name /dev/video5
18     pad0: Sink
19         <- "rkisp-isp-subdev":2 [ENABLED]
20
21 - entity 29: rkisp_rawwr0 (1 pad, 1 link)
22     type Node subtype V4L flags 0
23     device node name /dev/video7
24     pad0: Sink
25         <- "rkisp-csi-subdev":2 [ENABLED]
26
27 - entity 35: rkisp_rawwr1 (1 pad, 1 link)
28     type Node subtype V4L flags 0
29     device node name /dev/video8
30     pad0: Sink
31         <- "rkisp-csi-subdev":3 [ENABLED]
32
33 - entity 41: rkisp_rawwr2 (1 pad, 1 link)
34     type Node subtype V4L flags 0
35     device node name /dev/video9
36     pad0: Sink
37         <- "rkisp-csi-subdev":4 [ENABLED]
38

```

## 3.4 通道对应的video采集限制

### 3.4.1 ISP通路

- pipeline 切换

```

1
2     开机启动脚本需要添加    (/dev/media1根据实际isp注册情况决定)
3
4     media-ctl -d /dev/media1 -l '"rkisp-isp-subdev":2->"rkisp-bridge-
ispp":0[0]'
5     media-ctl -d /dev/media1 -l '"rkisp-isp-subdev":2->"rkisp_mainpath":0[1]'
6

```

- stream on 开关

```
1
2  因为rkisp_mainpath、rkisp_rawwr0/1/2 四个通道的stream on开关没有单独的开关，
3  因此如果要采集rkisp_rawwr0/1/2 三路通道任一路，都要需要保证rkisp_mainpath通道已
4  经在stream on状态之后，该三路才会出流。
```

- rkisp\_mainpath 格式切换

```
1
2  默认输出1080p，如果要格式切换720p，需要先执行：
3
4  media-ctl -d /dev/media1 --set-v4l2 '"m01_b_nvp6188 1-
0032":0[fmt:UYVY8_2X8/1280x720] '
5  media-ctl -d /dev/media1 --set-v4l2 '"rkisp-csi-
subdev":1[fmt:UYVY8_2X8/1280x720] '
6  media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-
subdev":0[fmt:YUYV8_2X8/1280x720] '
7  media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-subdev":0[crop:
(0,0)/1280x720] '
8  media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-
subdev":2[fmt:YUYV8_2X8/1280x720] '
9  media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-subdev":2[crop:
(0,0)/1280x720] '
10
```

### 3.5 通道对应的分辨率查询、视频信号查询

- media-ctl -p -d /dev/mediaX 获取 sensor 的 subdev node name

```
1
2  media-ctl -p -d /dev/media1
3  Media controller API version 4.19.111
4
5  Media device information
6  -----
7  driver          rkisp
8  model           rkisp0
9  serial
10 bus info
11 hw revision     0x0
12 driver version  4.19.111
13
14 Device topology
15
16 ....
17
18 - entity 92: m01_b_nvp6188 1-0032 (1 pad, 1 link)
19     type V4L2 subdev subtype Sensor flags 0
20     device node name /dev/v4l-subdev6
21     pad0: Source
22         [fmt:UYVY8_2X8/1920x1080 field:none]
23     -> "rockchip-mipi-dphy-rx":0 [ENABLED]
24
25
```

- open通道之前获取分辨率

```
1
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <sys/stat.h>
6 #include <sys/types.h>
7 #include <sys/time.h>
8 #include <sys/mman.h>
9 #include <sys/ioctl.h>
10 #include <linux/videodev2.h>
11
12 #define RKMODULE_MAX_VC_CH      4
13
14 struct rkmodule_vc_fmt_info {
15     __u32 width[RKMODULE_MAX_VC_CH];
16     __u32 height[RKMODULE_MAX_VC_CH];
17     __u32 fps[RKMODULE_MAX_VC_CH];
18 } __attribute__((packed));
19
20 struct rkmodule_vc_hotplug_info {
21     __u8 detect_status;
22 } __attribute__((packed));
23
24 #define RKMODULE_GET_VC_FMT_INFO \
25     _IOR('V', BASE_VIDIOC_PRIVATE + 12, struct rkmodule_vc_fmt_info)
26
27 #define RKMODULE_GET_VC_HOTPLUG_INFO \
28     _IOR('V', BASE_VIDIOC_PRIVATE + 13, struct rkmodule_vc_hotplug_info)
29
30 int main(int argc, char *argv[]) {
31     int ch = 0;
32     struct rkmodule_vc_hotplug_info status;
33     struct rkmodule_vc_fmt_info fmt;
34     int fd = open("/dev/v4l-subdev2", O_RDWR, 0);
35     ioctl(fd, RKMODULE_GET_VC_FMT_INFO, &fmt);
36     ioctl(fd, RKMODULE_GET_VC_HOTPLUG_INFO, &status);
37     for(ch = 0; ch < 4; ch++) {
38         printf("# ch: %d\n", ch);
39         printf("\t width: %d\n", fmt.width[ch]);
40         printf("\t height: %d\n", fmt.height[ch]);
41         printf("\t fps: %d\n", fmt.fps[ch]);
42         printf("\t plug in: %d\n", (status.detect_status & (1 << ch)) ? 1 :
0);
43     }
44     close(fd);
45     return 0;
46 }
47
```

## 3.6 实时查询热拔插接口

- 提供sysfs节点给用户层进行读查询。

```
1
2 /sys/devices/platform/ff510000.i2c/i2c-1/1-0032/hotplug_status
3 /sys/devices/platform/ff510000.i2c/i2c-1/1-0030/hotplug_status
4
```

## 4. rkmedia\_vmix\_vo\_dvr\_test应用说明

rkmedia\_vmix\_vo\_dvr\_test主要实现8路视频采集、编码，8路视频合成显示。源代码位于SDK/external/rkmedia/examples。

### 4.1 支持8路视频采集、H264编码

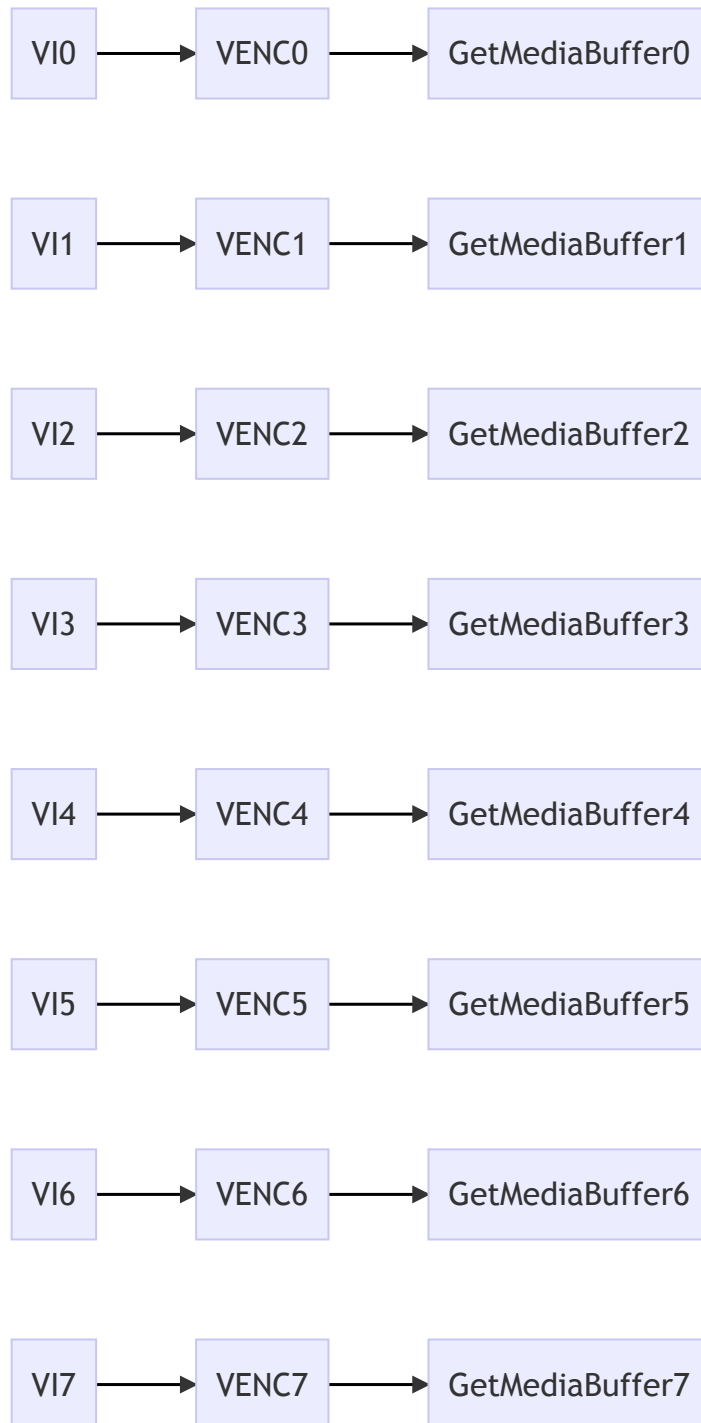
8路视频采集节点、分辨率、格式通过数组配置，方便用户修改调试：

```
1  stDvr dvr8[8] = {
2      {0, "/dev/video30", 1920, 1080, IMAGE_TYPE_NV12, 0},
3      {1, "/dev/video31", 1920, 1080, IMAGE_TYPE_NV12, 0},
4      {2, "/dev/video32", 1920, 1080, IMAGE_TYPE_NV12, 0},
5      {3, "/dev/video33", 1920, 1080, IMAGE_TYPE_NV12, 0},
6      {4, "/dev/video37", 1920, 1080, IMAGE_TYPE_NV12, 0},
7      {5, "/dev/video38", 1920, 1080, IMAGE_TYPE_NV12, 0},
8      {6, "/dev/video39", 1920, 1080, IMAGE_TYPE_NV12, 0},
9      {7, "/dev/video40", 1920, 1080, IMAGE_TYPE_NV12, 0},
10 };
```

根据双mipi方案的推荐，需要修改为：

```
1  stDvr dvr8[8] = {
2      {0, "/dev/video0", 1920, 1080, IMAGE_TYPE_NV16, 0},
3      {1, "/dev/video1", 1920, 1080, IMAGE_TYPE_NV16, 0},
4      {2, "/dev/video2", 1920, 1080, IMAGE_TYPE_NV16, 0},
5      {3, "/dev/video3", 1920, 1080, IMAGE_TYPE_NV16, 0},
6      {4, "/dev/video5", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
7      {5, "/dev/video7", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
8      {6, "/dev/video8", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
9      {7, "/dev/video9", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
10 };
```

8路视频VI通过bind VENC实现8路H264编码，通过GetMediaBuffer线程可以获取到8路VENC编码后的数据，用户可以在这个基础上实现视频传输需求。



## 4.2 支持8路视频合成显示

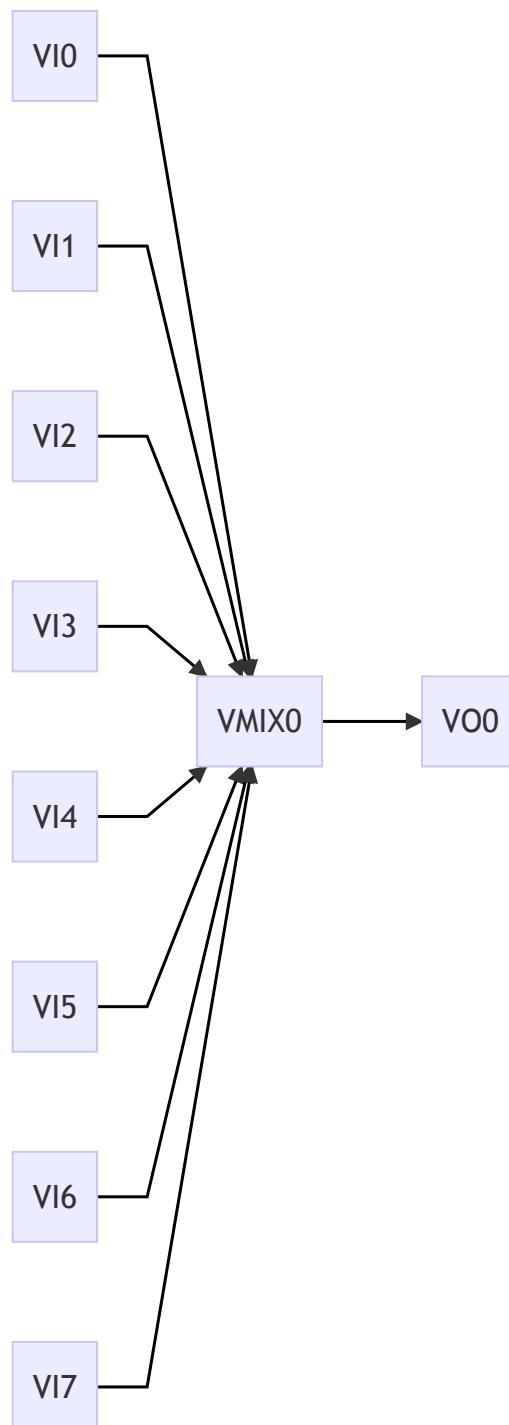
8路视频通过数组指定屏幕显示矩形区域，方便用户修改调试：

```

1  RECT_S area_2x4[8] = {
2      {0, 0, WIDTH / 2, HEIGHT / 4},
3      {WIDTH / 2, 0, WIDTH / 2, HEIGHT / 4},
4      {0, HEIGHT / 4, WIDTH / 2, HEIGHT / 4},
5      {WIDTH / 2, HEIGHT / 4, WIDTH / 2, HEIGHT / 4},
6      {0, HEIGHT / 2, WIDTH / 2, HEIGHT / 4},
7      {WIDTH / 2, HEIGHT / 2, WIDTH / 2, HEIGHT / 4},
8      {0, HEIGHT * 3 / 4, WIDTH / 2, HEIGHT / 4},
9      {WIDTH / 2, HEIGHT * 3 / 4, WIDTH / 2, HEIGHT / 4},
10 };

```

8路视频合成显示通过VMIX+VO模块实现:



## 4.3 支持8路视频切换为前4路、后4路显示

通过dvr\_bind、dvr\_unbind实现8路视频切换为前4路、后4路显示，用户只需要定义前4路和后4路的矩形显示区域即可。

## 4.4 支持区域画框

通过对整个屏幕画线实现区域画框，增加区域边界，通过数组指定画线区域，线宽最小为2，要求偶数：

```
1 RECT_S line_2x4[4] = {
2     {0, HEIGHT / 4, WIDTH, 2},
3     {0, HEIGHT / 2, WIDTH, 2},
4     {0, HEIGHT * 3 / 4, WIDTH, 2},
5     {WIDTH / 2, 0, 2, HEIGHT},
6 };
```

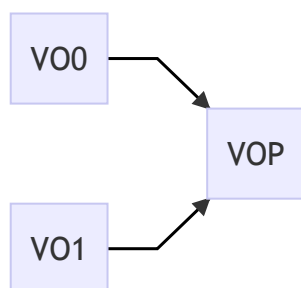
通过RK\_MPI\_VMX\_SetLineInfo设置画线区域。

## 4.5 支持RGN Cover

支持对每个通道设置敏感区域，通过RK\_MPI\_VMX\_RGN\_SetCover实现。

## 4.6 支持屏幕OSD

通过VO1实现屏幕OSD，用户可以把OSD绘制在buffer（格式为ABGR）里面，送VO1后通过Alpha即可以实现OSD叠加在VO0视频上面显示的效果。应用里面在osd\_thread线程里面每隔500ms绘制一块两条色块的OSD切换显示。



## 4.7 支持通道显示、隐藏

通过RK\_MPI\_VMX\_ShowChn、RK\_MPI\_VMX\_HideChn实现通道显示、隐藏。



## 4.8 支持通道的区域亮度获取

通过RK\_MPI\_VMIX\_GetChnRegionLuma实现通道的区域亮度获取，每次最多可以获取64个区域亮度，每个通道的区域的坐标都是相对通道的区域起始坐标，不是相对屏幕的起始坐标。可以通过区域亮度实现屏幕OSD反色效果。