测试Demo编译以及使用说明

文档将对DVR和NVR场景的测试程序使用进行描述，描述会包括当前已经融入进去的case，后续如果有增加case，测试方法一样。

测试程序位置：

sdk/verify/feature/mi\_demo/naboo

编译方法：

cd sdk/verify/feature/mi\_demo/

source build.sh

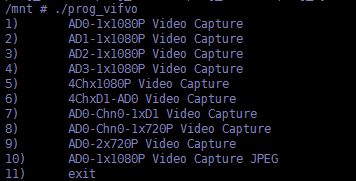
make

这样out/demo/app下会生成一系列prog\_\*的测试文件。

DVR测试（VIF）：

拷贝prog\_vifvo到目标板：

./prog\_vifvo



这时候可以看到如下提示，在公板上跑对应的case需要接入对应的视频流到对应的BNC接口，否则会无法出图。

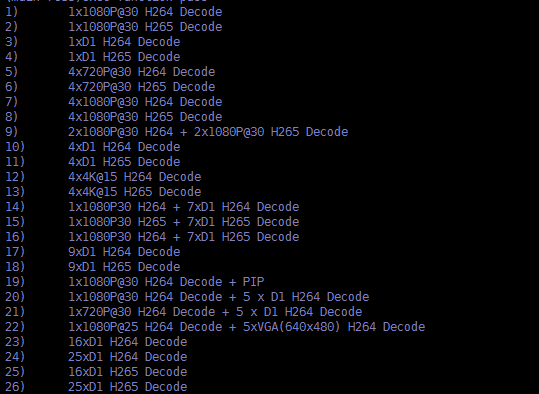
AD对应的Port关系请参考文档底部的附图说明，在测试对应的case时需要确保对应的模拟信号的format是正确的，否则将无法出图。

数据流：VIF->VPE->DISP->HDMI

NVR测试（VDEC）：

拷贝prog\_vdec到目标板：

./prog\_vdec



对应的测试项会播放对应的ES流，ES流存放在172.21.29.221的Temp\es\_for\_verify目录中

如果测试目录和文件名有变化，那么需要修改st\_common.h中的path定义

#define PREFIX\_PATH "/mnt/"

#define ST\_4K\_H264\_FILE "4K.h264"

#define ST\_1080P\_H264\_25\_FILE "1080P25.h264"

#define ST\_1080P\_H265\_25\_FILE "1080P25.h265"

……

#define ST\_MJPEG\_1080P\_P\_FILE "1080P\_Progessive.mjpeg"

数据流：VDEC->DIVP->DISP->HDMI

FrameBuffer测试：

拷贝prog\_fb到目标板：

./prog\_fb

这样可以看到显示器有显示对应鼠标，矩形等，有colorkey、alpha等属性

鼠标文件cursor.raw存放在testFile目录下。

编码测试(VENC)：

拷贝prog\_venc到目标板：

数据来源于YUV文件

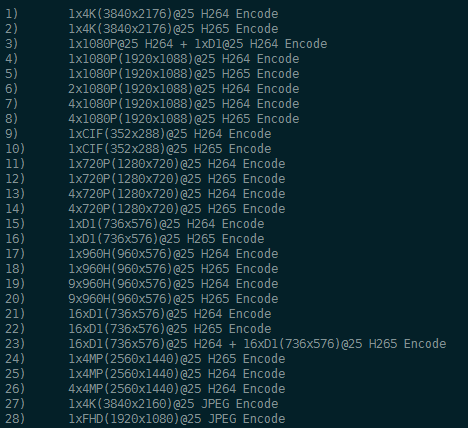
./prog\_venc -s 1920 1088 ../1920x1088\_nv12.yuv

参数说明

-s 宽 高

1920x1088\_nv12.yuv 编码用的YUV文件，不同的case需要用不同的YUV数据

编码后生成的es流文件保存在跟prog\_venc统一目录下



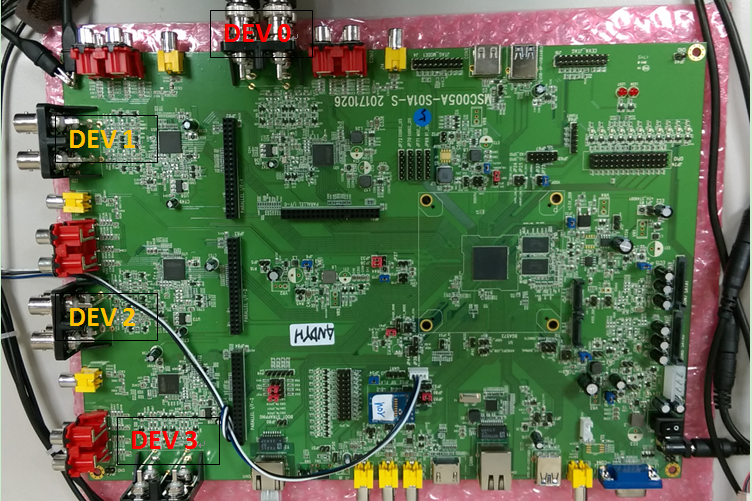
数据来源于VIF

需要将在st\_main\_vifvi.c中定义#define SUPPORT\_VIDEO\_ENCODE，这样VIF的数据过VPE后会有一个Port的数据送到DISP，一个Port的数据送至VENC。这样可以将VIF实时采集的画面编码成ES流保存起来。

数据流：VIF->VPE->(DISP/VENC)：port0->DISP，prot2->VENC

附AD图：

4颗AD的顺序。



Port口的顺序，A B C D对应的是4个port口的顺序，0-1-2-3 or 4-5-6-7 or 8-9-10-11 or 12-13-14-15

