目录

[Demo概述 2](#_Toc29223726)

[代码位置 2](#_Toc29223727)

[编译方法 2](#_Toc29223728)

[使用范围 2](#_Toc29223729)

[使用方法 2](#_Toc29223730)

[Demo 代码解析 4](#_Toc29223731)

[main 函数 4](#_Toc29223732)

[初始化Sensor->Vif->Vpe 4](#_Toc29223733)

[init sys 4](#_Toc29223734)

[init Sensor 4](#_Toc29223735)

[init VIF 5](#_Toc29223736)

[init VPE 5](#_Toc29223737)

[Vif Bind Vpe 7](#_Toc29223738)

[退出流程Vpe->Vif->Sensor 7](#_Toc29223739)

[unbind VIF->VPE 7](#_Toc29223740)

[destroy VPE 8](#_Toc29223741)

[destroy VIF 8](#_Toc29223742)

[destroy Sensor 8](#_Toc29223743)

[destroy SYS 8](#_Toc29223744)

[动态切换流程 9](#_Toc29223745)

[切换Sensor 分辨率/Hdr 9](#_Toc29223746)

[切换Rotation/ChnMirror Flip 9](#_Toc29223747)

[切换PortMode 11](#_Toc29223748)

[切换Channel Crop/Port Crop 12](#_Toc29223749)

[切换LDC On/Off 13](#_Toc29223750)

# Demo概述

## 代码位置

存放在sdk/verify/mi\_demo/alderaan/vpe 目录下, ini 用例文件也在该目录下。

## 编译方法

1. 确保整包image 有编译通过
2. 在sdk/verify/mi\_demo/ 目录下make clean && make, 编译整包mi\_demo

或者在sdk/verify/mi\_demo/alderaan/目录下 make vpe， 只编译vpe demo

1. 在sdk/verify/mi\_demo/out/demo/app目录下生成prog\_vpe bin 文件。

## 使用范围

该Demo 适用于325/325DE/327DE，336D/336Q/339G， 335/337DE系列芯片。

Demo 提供如下功能验证：

1. 模块功能： 单sensor，双sensor, 三sensor，YUV sensor，Ldc（Config/bin），Zoom(Vif crop/Vpe Channel crop/Vpe Port Crop 三个点选择)
2. Channel 功能: Sensor resolution，HDR type，Rotation，Channel Mirror/Flip， Channel Crop， Channel 3DnrLevel。
3. Port 功能：Port crop， Port mirror/flip， Resolution， pixel format， IMI/Ring/Frame mode。

## 使用方法

./prog\_vpe param.ini

Prog\_vpe用解析ini 配置文件的方法设置参数， 每一个ini 文件代表一个sensor 通路上模块参数配置。

ini 重点参数解释：

1. UseVenc 为1时vpe bind venc 使用Rtsp preview 看图像；

为0时vpe out 通过写文件的方式存储，隔绝venc。

1. SensorPad/ VifDev 固定搭配如下：



SensorPad =0 ==》 VifDev=0

SensorPad =1 ==》 VifDev=2。

1. RunMode 指Vif -> Vpe 之间绑定状态，通常情况下设定为：

单bayer sensor ==》 RealTime

多Bayer Sensor ==》 FrameMode

YUV Sensor ==》 MIPI接口用Frame Mode, BT656接口用DVRMode

1. ChnPortMode

scl0 from Rdma= 0x01, scl1 from Rdma= 0x02, scl2 from Rdma= 0x04.

ChnPortMode = scl0 Frame Rdma | scl1 Frame Rdma| scl2 Frame Rdma

1. LdcBinPath 不为NULL 时 开LDC， path 后缀为.bin时认为对应的是ldc bin 文件， 后缀为.cfg时 会通过算法转换出对应的bin 文件。

在双sensor 场景时， 执行方法为：

./prog\_vpe param\_snr0.ini param\_snr1.ini

单sensor 场景， 执行方法为：

./prog\_vpe param\_realtime.ini

ldc 场景， 执行方法为：

./prog\_vpe param\_ldc.ini

# Demo 代码解析

## main 函数

ST\_BaseModuleInit(eSnrPad)

初始化Sensor->Vif->Vpe，并且绑定VIF->VPE

ST\_VencStart(u32MaxWidth,u32MaxHeight, vpechn)

初始化Venc，并且绑定VPE port->VENC

ST\_RtspServerStart();

启用RTSP Server

ST\_RtspServerStop();

停止Rtsp Server

ST\_VencStop(vpechn)

VPE port -> venc解绑， 销毁Venc

ST\_BaseModuleUnInit(eSnrPad)

vif->vpe 解绑， vpe->vif->sensor 退出

## 初始化Sensor->Vif->Vpe

ST\_BaseModuleInit

### init sys

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step1: init SYS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(ST\_Sys\_Init());

### init Sensor

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step2: init Sensor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MI\_SNR\_SetPlaneMode(eSnrPad, False);//Hdr 设置为TRUE

MI\_SNR\_QueryResCount(eSnrPadId, &u32ResCount)// 获取支持分辨率数量

for(u8ResIndex=0; u8ResIndex < u32ResCount; u8ResIndex++)

{

MI\_SNR\_GetRes(eSnrPadId, u8ResIndex, &stRes);

//轮询获取支持的Resolution 列表参数，选择需要的Resolution index

}

/\*\*choice Resolution\*\*/

MI\_SNR\_SetRes(eSnrPadId,u8ChocieRes);

MI\_SNR\_Enable(eSnrPadId);

注：上述流程即初始化完成可以运行，如果是用固定Sensor，而且已知分辨率Index, 可省略选分辨率流程; 其它参数没有设置时用默认值， 例如Mirror/Flip 为False， fps为设置resolution时的max fps。

### init VIF

Vif的参数很多都是从Sensor 属性中获取到的。

STCHECKRESULT(MI\_SNR\_GetPadInfo(eSnrPadId, &stPad0Info));

STCHECKRESULT(MI\_SNR\_GetPlaneInfo(eSnrPadId, 0, &stSnrPlane0Info));

u32CapWidth=stSnrPlane0Info.stCapRect.u16Width;//Sensor 输出有效数据宽

u32CapHeight=stSnrPlane0Info.stCapRect.u16Height;//Sensor 输出有效数据高

ePixFormat = (MI\_SYS\_PixelFormat\_e)RGB\_BAYER\_PIXEL(stSnrPlane0Info.ePixPrecision, stSnrPlane0Info.eBayerId);

//将sensor输出的Bayer排列和bit数组合成MI 层定义的pixel format 用来设置。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step3: init VIF

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VIF\_SetDevAttr(vifDev, &stDevAttr));

STCHECKRESULT(MI\_VIF\_EnableDev(vifDev));

STCHECKRESULT(MI\_VIF\_SetChnPortAttr(vifChn, vifPort, &stVifPortInfo));

STCHECKRESULT(MI\_VIF\_EnableChnPort(vifChn, vifPort));

注：

1. 代码的设置为了兼容BT656/MIPI interface， 如果只有其中一种相关可以只设置相关参数即可。

eWorkMode的选择根据sensor 接口和输出格式来决定， 例如：

BT656 Sensor ==》 eWorkMode = E\_MI\_VIF\_WORK\_MODE\_1MULTIPLEX

单颗Mipi Sensor ==》eWorkMode = E\_MI\_VIF\_WORK\_MODE\_RGB\_REALTIME

多颗Mipi Sensor ==》eWorkMode = E\_MI\_VIF\_WORK\_MODE\_RGB\_FRAMEMODE。

2. E\_MI\_VIF\_WORK\_MODE\_2MULTIPLEX,E\_MI\_VIF\_WORK\_MODE\_4MULTIPLEX 仅MSR930芯片支持，设置根据BT656 sensor 分辨率选择。eHdrType 需要与Vpe 的Hdr Type 一致。

### init VPE

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step4: init VPE (create one VPE)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VPE\_CreateChannel(vpechn, &stChannelVpeAttr));

STCHECKRESULT(MI\_VPE\_SetChannelParam(vpechn, &stChannelVpeParam));

STCHECKRESULT(MI\_VPE\_SetChannelRotation(vpechn, pstVpeChnattr->eVpeRotate));

/\*Ldc/Zoom 功能时需要这段， 没有该场景可以不用设置\*/

if(pstVpeChnattr->ldcBinBuffer[0] != NULL)

{

MI\_VPE\_LDCBegViewConfig(vpechn);

for(i=0; i<pstVpeChnattr->u32ViewNum; i++)

{

MI\_VPE\_LDCSetViewConfig(vpechn, pstVpeChnattr->ldcBinBuffer[i], pstVpeChnattr->u32LdcBinSize[i]);

//free(pstVpeChnattr->ldcBinBuffer);

if(mi\_eptz\_buffer\_free(pstVpeChnattr->ldcBinBuffer[i]) != MI\_EPTZ\_ERR\_NONE)

{

printf("[MI EPTZ ERR] %d !! \n", \_\_LINE\_\_);

}

}

STCHECKRESULT(MI\_VPE\_LDCEndViewConfig(vpechn));

}

if(pstVpeChnattr->eVpeRotate == E\_MI\_SYS\_ROTATE\_90

|| pstVpeChnattr->eVpeRotate == E\_MI\_SYS\_ROTATE\_270)

{

if(pstVpeChnattr->stOrgVpeChnCrop.u16Height !=0)

pstVpeChnattr->stVpeChnCrop.u16X = u32CapHeight - pstVpeChnattr->stOrgVpeChnCrop.u16Y-pstVpeChnattr->stOrgVpeChnCrop.u16Height;

else

pstVpeChnattr->stVpeChnCrop.u16X = 0;

pstVpeChnattr->stVpeChnCrop.u16Y = pstVpeChnattr->stOrgVpeChnCrop.u16X;

pstVpeChnattr->stVpeChnCrop.u16Width = pstVpeChnattr->stOrgVpeChnCrop.u16Height;

pstVpeChnattr->stVpeChnCrop.u16Height = pstVpeChnattr->stOrgVpeChnCrop.u16Width;

}

else

{

pstVpeChnattr->stVpeChnCrop.u16X = pstVpeChnattr->stOrgVpeChnCrop.u16X;

pstVpeChnattr->stVpeChnCrop.u16Y = pstVpeChnattr->stOrgVpeChnCrop.u16Y;

pstVpeChnattr->stVpeChnCrop.u16Width = pstVpeChnattr->stOrgVpeChnCrop.u16Width;

pstVpeChnattr->stVpeChnCrop.u16Height = pstVpeChnattr->stOrgVpeChnCrop.u16Height;

}

STCHECKRESULT(MI\_VPE\_SetChannelCrop(vpechn, &pstVpeChnattr->stVpeChnCrop));

STCHECKRESULT(MI\_VPE\_StartChannel (vpechn));

for(i=0; i<ST\_MAX\_PORT\_NUM; i++)

{

STCHECKRESULT(MI\_VPE\_SetPortCrop(vpechn,i,

&pstVpeChnattr->stVpePortAttr[i].stPortCrop);

STCHECKRESULT(MI\_VPE\_SetPortMode(vpechn, i, &stVpeMode));

STCHECKRESULT(MI\_VPE\_EnablePort(vpechn, i));

}

### Vif Bind Vpe

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step5: bind VIF->VPE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ST\_Sys\_BindInfo\_T stBindInfo;

memset(&stBindInfo, 0x0, sizeof(ST\_Sys\_BindInfo\_T));

stBindInfo.stSrcChnPort.eModId = E\_MI\_MODULE\_ID\_VIF;

stBindInfo.stSrcChnPort.u32DevId = vifDev;

stBindInfo.stSrcChnPort.u32ChnId = vifChn;

stBindInfo.stSrcChnPort.u32PortId = 0;

stBindInfo.stDstChnPort.eModId = E\_MI\_MODULE\_ID\_VPE;

stBindInfo.stDstChnPort.u32DevId = 0;

stBindInfo.stDstChnPort.u32ChnId = vpechn;

stBindInfo.stDstChnPort.u32PortId = 0;

stBindInfo.u32SrcFrmrate = 30;

stBindInfo.u32DstFrmrate = 30;

stBindInfo.eBindType = pstVifDevAttr->eBindType;

STCHECKRESULT(ST\_Sys\_Bind(&stBindInfo));

注：Vif Vpe Realtime， Bind Type = E\_MI\_SYS\_BIND\_TYPE\_REALTIME， 否则为E\_MI\_SYS\_BIND\_TYPE\_FRAME\_BASE

## 退出流程Vpe->Vif->Sensor

MI\_S32 ST\_BaseModuleUnInit(MI\_SNR\_PAD\_ID\_e eSnrPad)

### unbind VIF->VPE

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step1: unbind VIF->VPE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

memset(&stBindInfo, 0x0, sizeof(ST\_Sys\_BindInfo\_T));

stBindInfo.stSrcChnPort.eModId = E\_MI\_MODULE\_ID\_VIF;

stBindInfo.stSrcChnPort.u32DevId = vifDev;

stBindInfo.stSrcChnPort.u32ChnId = vifChn;

stBindInfo.stSrcChnPort.u32PortId = 0;

stBindInfo.stDstChnPort.eModId = E\_MI\_MODULE\_ID\_VPE;

stBindInfo.stDstChnPort.u32DevId = 0;

stBindInfo.stDstChnPort.u32ChnId = vpechn;

stBindInfo.stDstChnPort.u32PortId = 0;

stBindInfo.u32SrcFrmrate = 30;

stBindInfo.u32DstFrmrate = 30;

STCHECKRESULT(ST\_Sys\_UnBind(&stBindInfo));

### destroy VPE

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step2: destory VPE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for(i = 0; i < ST\_MAX\_PORT\_NUM; i ++)

{

if(pstVpeChnattr->stVpePortAttr[i].bUsed == TRUE)

{

STCHECKRESULT(MI\_VPE\_DisablePort(vpechn, i));

}

}

STCHECKRESULT(MI\_VPE\_StopChannel(vpechn));

STCHECKRESULT(MI\_VPE\_DestroyChannel(vpechn));

### destroy VIF

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step3: destory VIF

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VIF\_DisableChnPort(vifChn, 0));

STCHECKRESULT(MI\_VIF\_DisableDev(vifDev));

### destroy Sensor

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step4: destory Sensor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_SNR\_Disable(eSnrPad));

### destroy SYS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step5: destory SYS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_SYS\_Exit());

## 动态切换流程

注：这边demo 为了流程最简化，在如下流程中都没有Module 之间unbind ==》bind 操作， 这边实际测试做或者不做对功能都没有影响（因为每一个Module 停下时都会等Driver 将buffer 做完还回来）。因为之前有客户流程已经有用了unbind==》bind 操作， 客户自己选择是否做unbind ==》bind。

### 切换Sensor 分辨率/Hdr

MI\_BOOL ST\_DoChangeHdrRes(MI\_U32 u32SensorNum)

切换HDR 和 Sensor resolution的流程是一样的，流程如下：

停下Vpe Channel=》销毁Vif ==》销毁Sensor ==》初始化Sensor/Vif 设置Hdr/Resolution ==》设置VPE hdr==》启动Vpe Channel

STCHECKRESULT(MI\_VPE\_StopChannel(u8VpeChn));

[destroy VIF](#_destroy_VIF)

[destroy Sensor](#_destroy_Sensor)

[init Sensor](#_init_Sensor)

[init VIF](#_init_VIF)

MI\_VPE\_ChannelPara\_t stVpeChParam;

memset(&stVpeChParam, 0x0, sizeof(MI\_VPE\_ChannelPara\_t));

MI\_VPE\_GetChannelParam(u8VpeChn, &stVpeChParam);

stVpeChParam.eHDRType = pstVpeChnattr->eHdrType;

MI\_VPE\_SetChannelParam(u8VpeChn, &stVpeChParam);

STCHECKRESULT(MI\_VPE\_StartChannel(u8VpeChn));

### 切换Rotation/ChnMirror Flip

MI\_BOOL ST\_DoChangeRotate(MI\_U32 u32SensorNum)

流程如下：

销毁Venc ==》 停下Vpe Channel==》停下Vif ==》Set rotation/ChnMirror Flip ==》Rotation change port size ==》启用Vif ==》启用Vpe Channel==》创建Venc

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step1: Stop Venc (Because rot will change preview resolution)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(gbPreviewByVenc == TRUE)

{

ST\_VencStop(VpeChn);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step2: Stop Vpe (Wait driver all buffer done)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VPE\_StopChannel(VpeChn));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step3: Disable Vif Port

(Realtime mode Change Rot Will Change Isp Cfg, need stop push vif stream)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(pstVpeChnattr->eRunningMode == E\_MI\_VPE\_RUN\_REALTIME\_MODE)

STCHECKRESULT(MI\_VIF\_DisableChnPort(vifChn, VifPort));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step4: Set Vpe Rot/ChnMirror/ChnFlip

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MI\_VPE\_SetChannelRotation(VpeChn, pstVpeChnattr->eVpeRotate);

MI\_VPE\_ChannelPara\_t stChnParam;

memset(&stChnParam, 0x0, sizeof(MI\_VPE\_ChannelPara\_t));

MI\_VPE\_GetChannelParam(VpeChn, &stChnParam);

stChnParam.bMirror = pstVpeChnattr->bChnMirror;

stChnParam.bFlip = pstVpeChnattr->bChnFlip;

MI\_VPE\_SetChannelParam(VpeChn, &stChnParam);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step5: Rot switch Vpe Port size

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for(i=0; i<ST\_MAX\_PORT\_NUM; i++)

{

MI\_VPE\_SetPortCrop(VpeChn, i, &pstVpePortAttr->stPortCrop);

stVpeMode.u16Width = pstVpePortAttr->stPortSize.u16Width;

stVpeMode.u16Height = pstVpePortAttr->stPortSize.u16Height;

MI\_VPE\_SetPortMode(VpeChn , i, &stVpeMode);

pstVencattr[u32VencChn].u32Width = stVpeMode.u16Width;

pstVencattr[u32VencChn].u32Height = stVpeMode.u16Height;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step7: Start Vif

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(pstVpeChnattr->eRunningMode == E\_MI\_VPE\_RUN\_REALTIME\_MODE)

STCHECKRESULT(MI\_VIF\_EnableChnPort(vifChn, VifPort));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step8: Start Vpe

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VPE\_StartChannel (VpeChn));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step9: Start Venc

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(gbPreviewByVenc == TRUE)

{

ST\_VencStart(u32MaxVencWidth, u32MaxVencHeight, VpeChn);

}

注：

1. Venc需要销毁的原因是因为 Rotation之后width/height 很有可能超过Venc初始化设置的Max Width/height。 如果Max width/height 设置过大会消耗更多buffer。

2. 停下VIF/启用VIF 是在VIF->VPE 之间Realtime mode下才需要做。

### 切换PortMode

MI\_BOOL ST\_ChangePortMode(MI\_U32 u32SensorNum)

切换流程如下：

停下VENC ==》 停下Vpe 对应Port ==》重新设置Port 属性 ==》重新设置VENC channel pitch ==》启用VPE 对应port ==》启用VENC

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step1: Stop Venc (Because rot will change preview resolution)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(gbPreviewByVenc == TRUE)

{

if(pstVencattr->bUsed == TRUE)

ExecFunc(MI\_VENC\_StopRecvPic(pstVencattr->vencChn), MI\_SUCCESS);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step2: disable Vpe port

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MI\_VPE\_DisablePort(VpeChn, s32Portid);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step3: Set Port Mode

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

stVpePortMode.bMirror = pstVpePortAttr->bMirror;

stVpePortMode.bFlip = pstVpePortAttr->bFlip;

stVpePortMode.ePixelFormat = pstVpePortAttr->ePixelFormat;

stVpePortMode.u16Width = pstVpePortAttr->stOrigPortSize.u16Width;

stVpePortMode.u16Height = pstVpePortAttr->stOrigPortSize.u16Height;

MI\_VPE\_SetPortMode(VpeChn, s32Portid, &stVpePortMode);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step4: Set Venc Channel Pitch

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(gbPreviewByVenc == TRUE)

{

MI\_VENC\_ChnAttr\_t stChnAttr;

memset(&stChnAttr, 0x0, sizeof(MI\_VENC\_ChnAttr\_t));

ExecFunc(MI\_VENC\_GetChnAttr(pstVencattr->vencChn, &stChnAttr), MI\_SUCCESS);

stChnAttr.stVeAttr.stAttrH264e.u32PicWidth = stVpePortMode.u16Width;

stChnAttr.stVeAttr.stAttrH264e.u32PicHeight = stVpePortMode.u16Height;

ExecFunc(MI\_VENC\_SetChnAttr(pstVencattr->vencChn, &stChnAttr), MI\_SUCCESS);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step5: Enable Vpe Port

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

MI\_VPE\_EnablePort(VpeChn, s32Portid);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step6: Start Venc

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(gbPreviewByVenc == TRUE)

{

if(pstVencattr->bUsed == TRUE)

ExecFunc(MI\_VENC\_StartRecvPic(pstVencattr->vencChn), MI\_SUCCESS);

}

### 切换Channel Crop/Port Crop

Channel Crop 和Port crop 都是比较独立的功能， 和前后MI module 没有关系， 所以直接call 对应的API 接口即可。

MI\_BOOL ST\_DoSetChnCrop(MI\_U32 u32SensorNum)

{

STCHECKRESULT(MI\_VPE\_SetChannelCrop(VpeChn,&stVpeChnCrop));

}

MI\_BOOL ST\_DoSetPortCrop(MI\_U32 u32SensorNum)

{

STCHECKRESULT (MI\_VPE\_SetPortCrop(VpeChn , s32Portid, &stPortCropSize));

}

### 切换LDC On/Off

MI\_S32 ST\_SetLdcOnOff(MI\_U32 u32SensorNum)

切换流程如下：

停下Vpe Channel ==》Set Ldc On/Off ==》Ldc first On, Set ldc bin ==》启用Vpe Channel

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step1: Stop Vpe (Wait driver all buffer done)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VPE\_StopChannel(s32Channelid));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step2: Set Ldc on/off Param

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VPE\_GetChannelParam(s32Channelid, &stVpeChnParam));

printf("get channel param benldc %d, bmirror %d, bflip %d, e3dnrlevel %d, hdrtype %d \n",

stVpeChnParam.bEnLdc, stVpeChnParam.bMirror,stVpeChnParam.bFlip,stVpeChnParam.e3DNRLevel,stVpeChnParam.eHDRType);

stVpeChnParam.bEnLdc = s32LdcOnoff;

STCHECKRESULT(MI\_VPE\_SetChannelParam(s32Channelid, &stVpeChnParam));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step3: if Ldc First On Set Ldc Bin

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(s32LdcOnoff == TRUE && pstVpeChnattr->bEnLdc == FALSE)

{

STCHECKRESULT(MI\_VPE\_LDCBegViewConfig(s32Channelid));

STCHECKRESULT(MI\_VPE\_LDCSetViewConfig(s32Channelid, pstVpeChnattr->ldcBinBuffer[0], pstVpeChnattr->u32LdcBinSize[0]));

STCHECKRESULT(MI\_VPE\_LDCEndViewConfig(s32Channelid));

//free(pstVpeChnattr->ldcBinBuffer);

if(mi\_eptz\_buffer\_free(pstVpeChnattr->ldcBinBuffer[0]) != MI\_EPTZ\_ERR\_NONE)

{

printf("[MI EPTZ ERR] %d !! \n", \_\_LINE\_\_);

}

pstVpeChnattr->bEnLdc = TRUE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step4: Start Vpe

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

STCHECKRESULT(MI\_VPE\_StartChannel(s32Channelid));