```
Void Test_Stream()
{
UInt32 i;
CaptureLink_CreateParams capturePrm;
IpcLink_CreateParams ipcOutVpssPrm;
IpcLink_CreateParams ipcInVideoPrm;
EncLink_CreateParams encPrm;
DisplayLink_CreateParams displayPrm_SD;
IpcBitsOutLinkRTOS_CreateParams ipcBitsOutVideoPrm;
IpcBitsInLinkHLOS_CreateParams ipcBitsInHostPrm0;
IpcFramesOutLinkRTOS_CreateParams ipcFramesOutVpssPrm;
IpcFramesInLinkRTOS_CreateParams ipcFramesInDspPrm;

CaptureLink_VipInstParams *pCaptureInstPrm;
CaptureLink_OutParams *pCaptureOutPrm;

UInt32 ipcOutVpssId;
UInt32 ipcInVideoId;

/* IPC struct init */
MULTICH_INIT_STRUCT(IpcLink_CreateParams, ipcOutVpssPrm);
MULTICH_INIT_STRUCT(IpcLink_CreateParams, ipcInVideoPrm);
MULTICH_INIT_STRUCT(IpcBitsOutLinkRTOS_CreateParams, ipcBitsOutVideoPrm);
MULTICH_INIT_STRUCT(IpcBitsInLinkHLOS_CreateParams, ipcBitsInHostPrm0);
MULTICH_INIT_STRUCT(IpcFramesOutLinkRTOS_CreateParams, ipcFramesOutVpssPrm);
MULTICH_INIT_STRUCT(IpcFramesInLinkRTOS_CreateParams, ipcFramesInDspPrm);

OSA_printf("\n********** TEST USECASE ********\n");
MultiCh_detectBoard();
System_linkControl(SYSTEM_LINK_ID_M3VPSS,
SYSTEM_M3VPSS_CMD_RESET_VIDEO_DEVICES, NULL, 0, TRUE);
System_linkControl(SYSTEM_LINK_ID_M3VIDEO,
SYSTEM_COMMON_CMD_SET_CH2IVAHD_MAP_TBL,
&systemVid_encDecIvaChMapTbl,
sizeof(SystemVideo_Ivahd2ChMap_Tbl), TRUE);
gVcapModuleContext.captureId = SYSTEM_LINK_ID_CAPTURE;
gVencModuleContext.encId = SYSTEM_LINK_ID_VENC_0;
gVdisModuleContext.displayId[VDIS_DEV_SD] = SYSTEM_LINK_ID_DISPLAY_2;

ipcOutVpssId = SYSTEM_VPSS_LINK_ID_IPC_OUT_M3_0;
ipcInVideoId = SYSTEM_VIDEO_LINK_ID_IPC_IN_M3_0;
gVencModuleContext.ipcBitsOutRTOSId = SYSTEM_VIDEO_LINK_ID_IPC_BITS_OUT_0;
gVencModuleContext.ipcBitsInHLOSId = SYSTEM_HOST_LINK_ID_IPC_BITS_IN_0;

gUI_mcfw_config.noisefilterMode = DSS_VNF_ON;

/* Capture Link params */
CaptureLink_CreateParams_Init(&capturePrm);
capturePrm.outQueParams[0].nextLink = ipcOutVpssId;
capturePrm.outQueParams[1].nextLink = gVdisModuleContext.displayId[VDIS_DEV_SD];
capturePrm.tilerEnable = FALSE;
pCaptureInstPrm = &capturePrm.vipInst[0];
```

```
pCaptureInstPrm->vipInstId = SYSTEM_CAPTURE_INST_VIP0_PORTA;
pCaptureInstPrm->inDataFormat = SYSTEM_DF_YUV422P;
pCaptureInstPrm->standard = SYSTEM_STD_1080P_60;
pCaptureInstPrm->numOutput = 2;
/* First stream */
pCaptureOutPrm = &pCaptureInstPrm->outParams[0];
pCaptureOutPrm->dataFormat = SYSTEM_DF_YUV422I_UYVY;
pCaptureOutPrm->scEnable = FALSE;
pCaptureOutPrm->scOutWidth = 1920;
pCaptureOutPrm->scOutHeight = 1080;
pCaptureOutPrm->outQueId = 0;
// Second stream
pCaptureOutPrm = &pCaptureInstPrm->outParams[1];
pCaptureOutPrm->dataFormat = SYSTEM_DF_YUV420SP_UV;
pCaptureOutPrm->scEnable = FALSE;
pCaptureOutPrm->scOutWidth = 720;
pCaptureOutPrm->scOutHeight = 480;
pCaptureOutPrm->outQueId = 1;
// display link params
MULTICH_INIT_STRUCT(DisplayLink_CreateParams,displayPrm_SD);
displayPrm_SD.inQueParams[0].prevLinkId = gVcapModuleContext.captureId;
displayPrm_SD.inQueParams[0].prevLinkQueId = 2;
displayPrm_SD.displayRes =
gVdisModuleContext.vdisConfig.deviceParams[VDIS_DEV_SD].resolution;
displayPrm_SD.displayId = DISPLAY_LINK_DISPLAY_SD;


/* IPC Out VPSS link params */
ipcOutVpssPrm.inQueParams.prevLinkId = gVcapModuleContext.captureId;
ipcOutVpssPrm.inQueParams.prevLinkQueId = 0;
ipcOutVpssPrm.numOutQue = 1;
ipcOutVpssPrm.outQueParams[0].nextLink = ipcInVideoId;
ipcOutVpssPrm.notifyNextLink = TRUE;
ipcOutVpssPrm.notifyPrevLink = TRUE;
ipcOutVpssPrm.noNotifyMode = FALSE;

/* IPC In VIDEO params */
ipcInVideoPrm.inQueParams.prevLinkId = ipcOutVpssId;
ipcInVideoPrm.inQueParams.prevLinkQueId = 0;
ipcInVideoPrm.numOutQue = 1;
ipcInVideoPrm.outQueParams[0].nextLink = gVencModuleContext.encId;
ipcInVideoPrm.notifyNextLink = TRUE;
ipcInVideoPrm.notifyPrevLink = TRUE;
ipcInVideoPrm.noNotifyMode = FALSE;

/* Video Encoder Link params */
MULTICH_INIT_STRUCT(EncLink_CreateParams, encPrm);
{
EncLink_ChCreateParams *pLinkChPrm;
EncLink_ChDynamicParams *pLinkDynPrm;
VENC_CHN_DYNAMIC_PARAM_S *pDynPrm;
VENC_CHN_PARAMS_S *pChPrm;
```

```
for (i = 0; i < VENC_PRIMARY_CHANNELS; i++)
{
pLinkChPrm = &encPrm.chCreateParams[i];
pLinkDynPrm = &pLinkChPrm->defaultDynamicParams;

pChPrm = &gVencModuleContext.vencConfig.encChannelParams[i];
pDynPrm = &pChPrm->dynamicParam;

switch(gUI_mcfw_config.demoCfg.codec_combo) {
case 0: pLinkChPrm->format = IVIDEO_H264HP; break; //"SINGLE_H264"
case 1: pLinkChPrm->format = IVIDEO_MPEG4SP; break; //"SINGLE_MPEG4"
case 2: pLinkChPrm->format = IVIDEO_MJPEG; break; //"SINGLE_JPEG"
case 3: pLinkChPrm->format = (i==0)? IVIDEO_H264HP:IVIDEO_MJPEG;
break; //"H264_JPEG"
case 4: pLinkChPrm->format = (i==0)? IVIDEO_MPEG4SP:IVIDEO_MJPEG;
break; //"MPEG4_JPEG"
case 5: pLinkChPrm->format = (i==0)? IVIDEO_H264HP:IVIDEO_H264HP;
break; //"DUAL_H264"
case 6: pLinkChPrm->format = (i==0)? IVIDEO_MPEG4SP:IVIDEO_MPEG4SP;
break; //"DUAL_MPEG4"
case 7: pLinkChPrm->format = (i==0)? IVIDEO_H264HP:IVIDEO_MPEG4SP;
break; //"H264_MPEG4"
case 8: pLinkChPrm->format = (i==0)? IVIDEO_H264HP:IVIDEO_H264HP;
break; //"TRIPLE_H264"
case 9: pLinkChPrm->format = (i==0)? IVIDEO_MPEG4SP:IVIDEO_MPEG4SP;
break; //"TRIPLE_MPEG4"
default: pLinkChPrm->format = IVIDEO_H264HP;
}

pLinkChPrm->profile = gVencModuleContext.vencConfig.h264Profile[i];
pLinkChPrm->dataLayout = IVIDEO_PROGRESSIVE;
pLinkChPrm->fieldMergeEncodeEnable = FALSE;
pLinkChPrm->enableAnalyticinfo = pChPrm->enableAnalyticinfo;
pLinkChPrm->maxBitRate = pChPrm->maxBitRate;
pLinkChPrm->encodingPreset = pChPrm->encodingPreset;
pLinkChPrm->rateControlPreset = IVIDEO_USER_DEFINED; //pChPrm->rcType;
pLinkChPrm->enableHighSpeed = FALSE;
pLinkChPrm->enableWaterMarking = pChPrm->enableWaterMarking;
pLinkChPrm->StreamPreset = gUI_mcfw_config.StreamPreset[i];

pLinkDynPrm->intraFrameInterval = pDynPrm->intraFrameInterval;
pLinkDynPrm->targetBitRate = pDynPrm->targetBitRate;
pLinkDynPrm->interFrameInterval = 1;
pLinkDynPrm->mvAccuracy = IVIDENC2_MOTIONVECTOR_QUARTERPEL;
pLinkDynPrm->inputFrameRate = pDynPrm->inputFrameRate;
pLinkDynPrm->rcAlg = pDynPrm->rcAlg;
pLinkDynPrm->qpMin = pDynPrm->qpMin;
pLinkDynPrm->qpMax = pDynPrm->qpMax;
pLinkDynPrm->qpInit = pDynPrm->qpInit;
pLinkDynPrm->vbrDuration = pDynPrm->vbrDuration;
pLinkDynPrm->vbrSensitivity = pDynPrm->vbrSensitivity;
```

```
encPrm.numBufPerCh[i] = 4;
gVencModuleContext.encFormat[i] = pLinkChPrm->format;
}
}

/* Video Encoder Framerate */

encPrm.chCreateParams[0].defaultDynamicParams.inputFrameRate = 30;
ENC_LINK_DEFAULT_ALGPARAMS_INPUTFRAMERATE;
encPrm.chCreateParams[1].defaultDynamicParams.inputFrameRate = 30;
ENC_LINK_DEFAULT_ALGPARAMS_INPUTFRAMERATE;


if(gUI_mcfw_config.vaUseCase == TRUE)
{
encPrm.isVaUseCase = 1;
}
else
{
encPrm.isVaUseCase = 0;
}

for (i = VENC_PRIMARY_CHANNELS; i < (VENC_CHN_MAX - 1); i++)
{
encPrm.chCreateParams[i].format = IVIDEO_MJPEG;
encPrm.chCreateParams[i].profile = 0;
encPrm.chCreateParams[i].dataLayout = IVIDEO_PROGRESSIVE;
encPrm.chCreateParams[i].fieldMergeEncodeEnable = FALSE;
encPrm.chCreateParams[i].defaultDynamicParams.intraFrameInterval = 0;
encPrm.chCreateParams[i].encodingPreset = 0;
encPrm.chCreateParams[i].enableAnalyticinfo = 0;
encPrm.chCreateParams[i].enableWaterMarking = 0;
encPrm.chCreateParams[i].defaultDynamicParams.inputFrameRate = 60;
encPrm.chCreateParams[i].rateControlPreset = 0;
encPrm.chCreateParams[i].defaultDynamicParams.targetBitRate = 100 * 1000;
encPrm.chCreateParams[i].defaultDynamicParams.interFrameInterval = 0;
encPrm.chCreateParams[i].defaultDynamicParams.mvAccuracy = 0;
gVencModuleContext.encFormat[i] = encPrm.chCreateParams[i].format;
}
encPrm.inQueParams.prevLinkId = ipcInVideoId;
encPrm.inQueParams.prevLinkQueId = 0;
encPrm.outQueParams.nextLink = gVencModuleContext.ipcBitsOutRTOSId;

/* IPC Bits Out VIDEO Link params */
ipcBitsOutVideoPrm.baseCreateParams.inQueParams.prevLinkId = gVencModuleContext.encId;
ipcBitsOutVideoPrm.baseCreateParams.inQueParams.prevLinkQueId = 0;
ipcBitsOutVideoPrm.baseCreateParams.numOutQue = 1;
ipcBitsOutVideoPrm.baseCreateParams.outQueParams[0].nextLink =
gVencModuleContext.ipcBitsInHLOSId;
MultiCh_ipcBitsInitCreateParams_BitsOutRTOS(&ipcBitsOutVideoPrm, TRUE);

/* IPC Bits In HOST Link params */
```

```
ipcBitsInHostPrm0.baseCreateParams.inQueParams.prevLinkId =
gVencModuleContext.ipcBitsOutRTOSId;
ipcBitsInHostPrm0.baseCreateParams.inQueParams.prevLinkQueId = 0;
MultiCh_ipcBitsInitCreateParams_BitsInHLOS(&ipcBitsInHostPrm0);

/* Links Creation */
/* Capture Link */

System_linkCreate(gVcapModuleContext.captureId, &capturePrm, sizeof(capturePrm));
System_linkControl(gVcapModuleContext.captureId,
CAPTURE_LINK_CMD_DETECT_VIDEO, NULL, 0, TRUE);

/* IPC Links */
System_linkCreate(ipcOutVpssId, &ipcOutVpssPrm, sizeof(ipcOutVpssPrm));
System_linkCreate(ipcInVideoId, &ipcInVideoPrm, sizeof(ipcInVideoPrm));

/* Video Encoder Link */
System_linkCreate(gVencModuleContext.encId, &encPrm, sizeof(encPrm));

/* IPC Bits Links */
System_linkCreate(gVencModuleContext.ipcBitsOutRTOSId, &ipcBitsOutVideoPrm,
sizeof(ipcBitsOutVideoPrm));
System_linkCreate(gVencModuleContext.ipcBitsInHLOSId, &ipcBitsInHostPrm0,
sizeof(ipcBitsInHostPrm0));

/* display link */
System_linkCreate(gVdisModuleContext.displayId[VDIS_DEV_SD], &displayPrm_SD,
sizeof(displayPrm_SD));
gNoiseFilterMode = gUI_mcfw_config.noisefilterMode;
OSA_printf("USECASE SETUP DONE\n");
}
```