# FAQ V1.9

|  |  |  |
| --- | --- | --- |
| **Date** | **Auther** | **Version** |
| 2018/12/05 | Leo Create | 1.0 |
| 2019/07/19 | Leo | 1.9 |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[FAQ V1.9 1](#_Toc14515593)

[1、 BT1120接口无输出 3](#_Toc14515594)

[2、 AI模块通过接口返回检测结果后视频不画框 3](#_Toc14515595)

[3、 使用RS485发送串口数据 4](#_Toc14515596)

[4、 使用串口0做RS485输出 5](#_Toc14515597)

[5、 需要平面格式的YUV420数据 6](#_Toc14515598)

[6、 调用IF\_GetYUVData接口获取YUV数据返回错误 6](#_Toc14515599)

[7、 需要使用海思的视频数据结构体 7](#_Toc14515600)

[8、 客户希望在nsd运行前执行自己的脚本做部分预处理 7](#_Toc14515601)

[9、 客户自开发的动态库打包 8](#_Toc14515602)

[10、 使用DeviceTool升级失败 8](#_Toc14515603)

[11、 使用Hisi3559A设备独立获取每一路的YUV数据 9](#_Toc14515604)

[12、 操作GPIO 9](#_Toc14515605)

[13、 修改内存分配 10](#_Toc14515606)

[14、 SD卡的使用 10](#_Toc14515607)

[15、 加入libailib.so后出现nsd运行崩溃 11](#_Toc14515608)

[16、 使用callback方式获取YUV数据 11](#_Toc14515609)

[17、 如何定制默认值 11](#_Toc14515610)

[18、 修改获取的YUV分辨率 13](#_Toc14515611)

[19、 关闭VO输出 14](#_Toc14515612)

[20、 编译器下载链接 14](#_Toc14515613)

[21、 格式化SD卡失败 14](#_Toc14515614)

[22、 如何获取视频编码数据 15](#_Toc14515615)

[23、 RTSP的URL 15](#_Toc14515616)

[24、 Onvif版本 15](#_Toc14515617)

[25、 硬件AI下的缓存机制 16](#_Toc14515618)

[26、 手动启动nsd没有问题， 通过脚本自动启动nsd则无法启动 16](#_Toc14515619)

[27、 Mipi\_tx接入屏幕无输出 17](#_Toc14515620)

[28、 设置framebuffer的alpha和colorkey 17](#_Toc14515621)

[29、 天瞳设备工作异常 18](#_Toc14515622)

[30、 人脸FTP联动抓图的步骤及注意事项 18](#_Toc14515623)

[31、 希望在nsd运行之后执行自己的脚本 19](#_Toc14515624)

[32、 播放音频文件流程 20](#_Toc14515625)

[33、 双目设备对灯光的控制 22](#_Toc14515626)

[34、 重启设备后参数未被保存 23](#_Toc14515627)

[35、 控制MIPI屏幕亮度 23](#_Toc14515628)

[36、 客户自己开发AI模块编码后的图希望通过FTP上传 23](#_Toc14515629)

[37、 AI模块希望感知日夜切换 24](#_Toc14515630)

[38、 Wiegand驱动使用方式 25](#_Toc14515631)

## BT1120接口无输出

1. 检查cat /mnt/mtd/nsd/config/setting/vout.conf, outdev字段需要为3，on需要为1， outmode字段如果为0xfff0则输入YUV422， 如果为0xfff1则输出YUV420。

/mnt/mtd/nsd/config/setting/video.conf里面system字段如果为1则输出VO\_OUTPUT\_1080P25 如果system字段为2，则输出 VO\_OUTPUT\_1080P30

|  |
| --- |
| typedef enum \_NSD\_VOUT\_DEV\_TYPE\_E  {  NSD\_CVBS=1,  NSD\_YPBPR,  NSD\_HDMI,  NSD\_LCD  }NSD\_VOUT\_DEV\_TYPE\_E;  typedef enum \_NSD\_VOUT\_MODE\_E  {  NSD\_YUV\_SEMIPLANAR\_422 = 0xFFF0,  NSD\_YUV\_SEMIPLANAR\_420 = 0xFFF1,  NSD\_YUV\_420 = 0xFFF2,  NSD\_YUV\_422 = 0xFFF3,  }NSD\_VOUT\_DEV\_MODE\_E; |

## AI模块通过接口返回检测结果后视频不画框

1. 检查/mnt/mtd/code/fixed/sys.def里面externaltype字段, 如果是BT1120送YUV的方式做AI检测， 那么该字段需要为0x40000或者0x80000, 如果是通过IF\_GetYUVData获取YUV方式做AI检测， 那么该字段需要为0x20000
2. 返回的YUVAnalyzeResult中drawLineFlag字段需要为1
3. nsd启动会通过AI\_AddAnalyseCallback注册分析结果回调函数， AI模块分析完YUV后通过注册的回调函数来通知nsd分析的结果，例如：

|  |
| --- |
| YUVAnalyzeResult analyzeResult;  memset(&analyzeResult,0,sizeof(YUVAnalyzeResult));    analyzeResult.faceResult[0].resultPosition.u16X = 50; //绝对坐标  analyzeResult.faceResult[0].resultPosition.u16Y = 50;  analyzeResult.faceResult[0].resultPosition.u16Width = 200;  analyzeResult.faceResult[0].resultPosition.u16Height = 200;  analyzeResult.resultNumber = 1;  analyzeResult.drawLineFlag = 1;  analyzeResult.normalizationX = 1920; //YUV的宽度  analyzeResult.normalizationY = 1080; //YUV的高度  analyzeResult.analyzeType = NSD\_AI\_FACE;    std::map<long, CALLBACK\_FUNCTION\_WITH\_CH>::iterator it = CAiLib::GetInstance().m\_callback\_list.begin();  while (it != CAiLib::GetInstance().m\_callback\_list.end())  {  it->second.callback\_func(it->first, NSD\_AI\_FACE, &analyzeResult);  ++it;  } |

上述示例中，AI模块告诉nsd， 在坐标left 50,top 50,width 200,height 200的地方有物体. normalizationX和normalizationY指的是坐标的归一化系数， 可以用YUV的宽高也赋值，resultNumber为物体的个数， drawLineFlag=1说明需要画框

## 使用RS485发送串口数据

1. 检查/mnt/mtd/code/fixed/sys.def里面rs485字段， 需要设置成1
2. 检测/mnt/mtd/nsd/config/setting/uart.conf文件中 ch=1的uart节点里面各项串口参数是否符合预期
3. 使用IF\_WriteUartData发送数据， 参数nCh给1. 华途HVP中，0是留给球机使用的

|  |
| --- |
| [root]# cat /mnt/mtd/nsd/config/setting/uart.conf  <?xml version="1.0" encoding="GBK"?>  <root>  <uart ch='0' databit='0' stopbit='0' parity='0' flowctrl='0' buadrate='0' devname='' gpio='0' useptz='0'/>  <uart ch='1' databit='3' stopbit='1' parity='0' flowctrl='0' buadrate='2400' devname='/dev/ttyAMA1' gpio='1' useptz='1'/>  </root>  //C代码：  char szData[]={0x1,0x2,0x3}  CAiLib::GetInstance().IF\_WriteUartData(1, szData, 3);  使用该语句可以把szData发送至串口1 (/dev/ttyAMA1) |

## 使用串口0做RS485输出

因为串口0默认是调试串口， 所以为了避免调试输出干扰对端的处理， 可以做如下操作来屏蔽调试打印

1. 检查uboot命令行， bootargs项里面不能有console=ttyAMA0，115200的字段， 如果有， 则需要在uboot启动的时候删除该字段。

例如uboot命令行可以通过如下命令进行设置：

|  |
| --- |
| setenv bootargs 'mem=120M console=ttyAMA0,115200 root=/dev/mtdblock2 rootfstype=yaffs2 mtdparts=hinand:1M(boot),5M(kernel),15M(rootfs),-(user)'  saveenv |

1. 检查/etc/inittab文件::respawn:/sbin/getty -L ttyAMA0 115200 vt100 -n root -I "Auto login as root ..." 这行需要注释掉，然后重启设备

## 需要平面格式的YUV420数据

1. HVP默认给NV21格式的YUV420数据， 如果需要平面格式的YUV420数据， 需要调用接口IF\_SetParam设置PixelFormat为1。
2. 如果PixelFormat为1那么获取到的AI\_VIDEO\_FRAME\_S结构中pYUV[0]是Y数据, pYUV[1]是U数据, pYUV [2]是V数据。 如果PixelFormat为3， 那pYUV[0]为YUV数据， pYUV[1]和pYUV[2]不需要使用

|  |
| --- |
| NSD\_AI\_S ai\_cfg;  ai\_cfg.u16AiImageWidth = 640;  ai\_cfg.u16AiImageHeight = 480;  ai\_cfg.u8PixelFormat = 1;  IF\_SetParam(NSDCFG\_AI,&ai\_cfg,0,0); |

但是如果使用该种格式的YUV420， 板端会做一次格式转化，导致接口的耗时增加， 所以除非特别需要， 不建议使用平面格式的YUV420

## 调用IF\_GetYUVData接口获取YUV数据返回错误

出现该情况的原因有很多种， 可以做如下过滤

1. **情况一**： 有nsd在运行， 但是又手动运行nsd

手动运行nsd之前先ps查看是否进程里面已经有nsd在运行， 如果有的话， 可以修改/mnt/mtd/nsd/program/bin/run.sh禁用nsd，之后重启设备，再手动运行nsd调试

1. **情况二**： 杀掉nsd之后，再重启nsd导致获取不到YUV

只能使用killall -15 nsd或者Ctrl+C的方式退出nsd， 而且在nsd在退出的过程中不能抛出异常，否则海思的资源如果没有回收干净，会导致下一次重启nsd无法获取视频。

建议修改run.sh禁用nsd自启动， 然后每次重启设备手动运行nsd来调试

1. **情况三**： 海思MMZ内存不足

如果客户开发的库中分配了较多的海思内存， 导致海思内存不足， vb不够的情况下也同样无法获取YUV数据， 这个时候可以截取/dev/logmpp中的打印给到华途研发来分析是否为该情况

1. **情况四**: 编译动态库使用的头文件和设备nsd的版本不一致。

因为华途的产品本身也在用HVP作开发， 所以不同版本间的头文件可能会不一致， 所以客户编译动态库(libailib.so, plugin.so)如果用到的华途的中间接口层， 那么就需要和nsd所在版本的头文件保持一致。 nsd的版本可以在/mnt/mtd/nsd/program/bin/version.ini中查看。 头文件的版本为华途HVP提供的HTSolution的版本

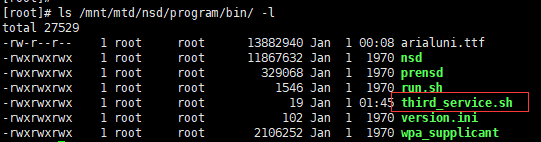
## 需要使用海思的视频数据结构体

有些客户需要用到海思IVE或者NNIE等开发智能算法时需要用到视频的物理地址。那么可以使用AI\_VIDEO\_FRAME\_S结构中的pInnerUse字段， 该指针可以转化成海思的结构直接使用

|  |
| --- |
| VIDEO\_FRAME\_INFO\_S \*frame = (VIDEO\_FRAME\_INFO\_S \*)llpVideoFrame->pInnerUse; |

## 客户希望在nsd运行前执行自己的脚本做部分预处理

华途的运行脚本为/mnt/mtd/nsd/program/bin/run.sh， nsd进程也是在该脚本中拉起来的， 如果客户希望在运行nsd之前做其他预操作， 那么可以创建一个名字叫third\_service.sh的脚本， 放在/mnt/mtd/nsd/program/bin/中。 这样设备启动时会自动调用该脚本。



## 客户自开发的动态库打包

客户开发完自身的libailib.so或者plugin.so以后需要输出一个文件夹以及third\_service.sh给到华途来进行打包。

打包流程为

1. 将客户提供的文件夹全部拷贝到/mnt/mtd/nsd/program/lib下面
2. 把third\_service.sh放到/mnt/mtd/nsd/program/bin下面
3. 提供制作动态库时所用的头文件版本

之后生成客户插件包

升级流程为

1. 升级华途通用主程序包
2. 升级客户插件包

**注意**： 主程序包和插件包的版本必须一致， 也就是说客户制作插件包使用的头文件版本必须和nsd版本一致， 避免出现非预期错误。

nsd的版本可以在/mnt/mtd/nsd/program/bin/version.ini中查看。 头文件的版本为华途HVP提供的HTSolution的版本

## 使用DeviceTool升级失败

1. 确定PC和设备是否在同一网段
2. 通过不断的ping设备确定设备是否在短时间内不断重启
3. 确定升级包是否和设备平台一致，例如3516A的升级包无法升级到3519设备上
4. 进入设备后台查看flash剩余空间(使用df命令)， 设备升级需要额外占用升级包大小的空间， 所以flash的剩余空间必须大于升级包大小。
5. DeveiceTool升级依赖设备端的ldefault进程。 网页升级依赖设备端的nsd进程。所以升级的时候设备上对应的进程必须存在(可以使用ps命令查看)

## 使用Hisi3559A设备独立获取每一路的YUV数据

1. 如果是拼接好的设备， 需要客户自行调用HI\_MPI\_VPSS\_GetChnFrame接口来获取每一路的YUV数据
2. 如果是多路未拼接的设备， 可以通过IF\_GetYUVData接口来获取每一路的YUV， 不建议把通道的宽高设置太大， 否则可能出现内存不足偶尔取不到图片的问题。 如果期望获取的YUV大小和sensor输入的宽高相同， 那么为了节省内存占用同样建议使用海思接口HI\_MPI\_VPSS\_GetChnFrame 来获取每一路的YUV数据。 该接口描述可以参考海思文档。
3. 使用海思接口获取完数据后， 需要调用HI\_MPI\_VPSS\_ReleaseChnFrame是否数据， 否则会导致后续无法再次获取到数据

## 操作GPIO

1. 如果客户在自己开发的库内需要操作GPIO， 那么请确保只是操作期望控制的那一个Bit而不是整个Group， 如果操作到了整组， 则容易操作到其他硬件功能。
2. 如果是Hisi3559A以外的设备(如3516AD，3519等)， 可以使用gpioctl工具来控制某个IO来进行测试， 如果是Hisi3559A设备， 可以参考海思文档《外围设备驱动 操作指南.pdf》了解如何操作GPIO

|  |
| --- |
| //非Hisi3559A平台  [root]# gpioctl  gpioctl 0 group bit value //to write gpio !!  gpioctl 1 group bit value //to read gpio !!  例如操作GPIO12\_0 设置高电平命令为 gpioctl 0 12 0 1  //Hisi3559A平台，例如操作GPIO12\_0, 12\*8+0=96  echo 96 > /sys/class/gpio/export  echo out > /sys/class/gpio/gpio96/direction #out输出 in输入  echo 1 > /sys/class/gpio/gpio96/value #1拉高 0拉低 |

## 修改内存分配

目前海思的板子内存会分为两部分，一部分给操作系统，一部分给海思MMZ。 通过cat /proc/cmdline 查看 mem 字段可以查看到目前分配给操作系统的内存有多少。

如果想自行修改分配比例， 可以**在uboot下修改bootargs环境变量的值**。 后续加载海思驱动的时候会自动识别这个值。

风险提示： 如果内存分配修改的不合理，比如给到MMZ的内存过小，可能会导致海思端无法取出视频流，除非对设备很熟，否则不建议修改该值。

## SD卡的使用

HVP在插入SD卡后会自动挂载到/tmp/sdCard， 如果没有该文件夹， 可能存在的原因

1. 没插卡
2. 卡坏了，或者卡槽坏了
3. 卡的文件系统不被识别。首次使用该卡需要在网页上格式化一下。
4. 查看/mnt/mtd/code/fixed/sys.def里面disknum是否为1

在开发libialib.so的时候如果需要用到SD卡，请先判断卡是否插入，其次是卡是否被挂载后再进行写入。

例如

If ( 卡插入 && isExist(/tmp/sdCard) ) {

//Read or write

}

IF\_IsSDCardExist 该接口可以用来判断是否有卡插入卡槽

## 加入libailib.so后出现nsd运行崩溃

a) 确定编译libailib.so时候用的头文件和升级包是否是同一个版本，如何确定版本见第6条的情况四

b) 使用华途默认的libailib.so是否会出现崩溃现象， 重新升级一下设备主程序即可覆盖libailib.so

c) 分段屏蔽libailib.so中的代码， 查看哪个部分会导致设备崩溃

## 使用callback方式获取YUV数据

HVP提供回调的方式获取YUV数据， 详情见HTSolution里面的ai\_callback例子

|  |
| --- |
| long (\*IF\_AddYuvStreamCallback)(unsigned short nCh, video\_yuvdata\_callback lpCallback);  Int (\*IF\_RemoveYuvStreamCallback)(unsigned short nCh,long hHandle); |

HT流程为

获取海思数据--->调用客户注册的回调函数--->把数据给到编码器

所以如果客户在回调函数中消耗的时间过长， 则会阻塞nsd的线程， 导致降帧

如果该回调始终无法被调用到，那么请查看/mnt/mtd/code/fixed/sys.def中externtype是否为0x40000 或者 0x80000(硬件AI)或者0x20000(软件AI).

注： 使用该方式会导致nsd和libailib.so的耦合性较高， 一旦出现问题不容易定界， 所以不推荐使用该方式获取YUV， 最好的方式还是使用IF\_GetYUVData来获取

## 如何定制默认值

程序加载某功能配置的默认值顺序为

Step1: 读取程序内部预置的默认值

Step2： 读取/mnt/mtd/nsd/config/OEM/000/\*.def

Step3： 读取/mnt/mtd/nsd/config/setting/\*.conf

设备每10秒刷新一次用户当前值到 \*.conf 文件

所以如果想定制某个功能选项， 可以提前在网页上配置好， 待10秒后， 在设备/mnt/mtd/nsd/config/setting/目录中，找到对应的xxx.conf文件， 把该文件名字改为xxx.def, 之后放到/mnt/mtd/nsd/config/OEM/000/中即可。 之后设备在复位时，就会采用xxx.def中的值作为初始值。

例如

在网页上配置好端口或者NTP等信息后等10秒，之后执行下述操作，即可完成定制



功能和配置文件的对应关系为

|  |  |
| --- | --- |
| **功能** | **对应的配置文件名** |
| IP，子网掩码，网关，DNS | eth.conf |
| 端口，NTP，upnp，ddns，rtsp，filter | net.conf |
| OSD | osd.conf |
| 亮度，锐度，曝光等图像参数 | img.conf |
| 串口参数配置 | uart.conf |
| 帧率，码率，分辨率等媒体参数 | video.conf |
| 联动动作 | action.conf |
| 智能配置(人脸识别等) | ai.conf |
| onvif, gb28181等平台参数 | platform.conf |
| 计划参数 | scheme.conf |
| 设备名称 | sys.conf |
| 模拟输出(HDMI/CVBS/BT1120) | vout.conf |
| 用户参数及用户组配置 | user.conf |
| FTP参数配置 | ftp.conf |

## 修改获取的YUV分辨率

1) 如果使用IF\_GetYUVData接口获取YUV数据， 有两种方式修改YUV分辨率

方式一： 使用IF\_SetParam(NSDCFG\_AI,& NSD\_AI\_S,0,0);接口设置智能通道的分辨率， 之后通过IF\_GetYUVData获取到的就是该分辨率的图像

方式二： 使用IF\_Scale接口进行分辨率转化

1. 如果使用回调方式获取YUV数据， 那么只能使用IF\_Scale的方式进行缩放

例子

|  |
| --- |
| int AI\_YUVDATA\_CALLBACK(unsigned short p\_ch, AI\_VIDEO\_FRAME\_S \*p\_yuv)  {  printf("width=%d, height=%d, pts=%ld\n", p\_yuv->nWidth, p\_yuv->nHeight, p\_yuv->nPts);  AI\_VIDEO\_FRAME\_S afterScale;  afterScale.nWidth = 1530;  afterScale.nHeight= 1530;  int nRet = CAiLib::GetInstance().IF\_Scale(p\_yuv, &afterScale);//Scale的宽需要4对齐  if (nRet == 0)  {  printf("after scale width is %d, height is %d \n", afterScale.nWidth, afterScale.nHeight);  }  else  {  printf("scale failed, ret is %d \n", nRet);  }  return 0;  } |

|  |
| --- |
| NSD\_AI\_S ai\_cfg;  ai\_cfg.u16AiImageWidth = 640;  ai\_cfg.u16AiImageHeight = 480;  ai\_cfg.u8PixelFormat = 3;  CAiLib::GetInstance().IF\_SetParam(NSDCFG\_AI,&ai\_cfg,0,0); |

## 关闭VO输出

如果希望不启用CVBS或者BT1120输出， 想自己通过海思接口来控制， 那么可以修改/mnt/mtd/nsd/config/setting/vout.conf， 把on改成0， 之后重启nsd进程， 设备则不会控制海思的VO模块。

或者通过网页，转到设置页面--->视频制式页里面，关闭启用标识

|  |
| --- |
| [root]# cat /mnt/mtd/nsd/config/setting/vout.conf  <?xml version="1.0" encoding="GBK"?>  <root>  <vout ch='0' outdev='1' outmode='65521' on='0'>  </vout>  </root> |

## 编译器下载链接

Hisi3516AD：链接：https://pan.baidu.com/s/1ouYy87WcFRDk-Hq6QyxfmQ 提取码：l07k

Hisi3519V101：链接：https://pan.baidu.com/s/1yom3\_5ZXGLkNEPUiamZhag 提取码：j94f

Hisi3559A：链接：https://pan.baidu.com/s/19-9dJ2FspwSkmox1O4hKMA 提取码：96jg

Hisi3519A：链接：https://pan.baidu.com/s/1GEVEmXUjgH4ouz5ifvCPag提取码：2hs8

## 格式化SD卡失败

1) 用fdisk工具查看SD卡是否有1分区，部分坏卡没有分区无法格式化

2) 手动在shell下执行mkfs.ext4 /dev/mmcblk0p1 (Hisi3559A和Hisi3519A为mmcblk1p1)查看手动格式化SD卡是否可以成功，根据格式化过程的打印信息判断是否是坏卡

3) 格式化SD卡之前会调用AI\_OnSDCardStatusChange通知AI模块要开始格式化了， 如果AI模块正在使用SD卡，需要在该接口内停掉对SD卡的使用并释放句柄。 格式化完成后，会再次通过该接口通知AI模块格式化已完成

4) 在2.7.2.0及之后的版本， AI库需要实现接口

int AI\_OnSDCardStatusChange(unsigned int nStatus)

否则会出现未定义符号，导致nsd进程退出

## 如何获取视频编码数据

使用IF\_ReadVideoFrameFromCache接口可以获取设备编码后的数据， 使用方法见《M\_INTERFACE\_SPEC.pdf》

## RTSP的URL

主码流： Rtsp://IP:Port/media/live/1/1

次码流： Rtsp://IP:Port/media/live/1/2

## Onvif版本

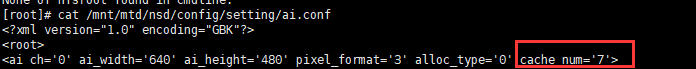
版本为：1612

|  |
| --- |
| <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2" xmlns:xop="http://www.w3.org/2004/08/xop/include" xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" targetNamespace="http://www.onvif.org/ver10/schema" elementFormDefault="qualified" version="16.12"> |

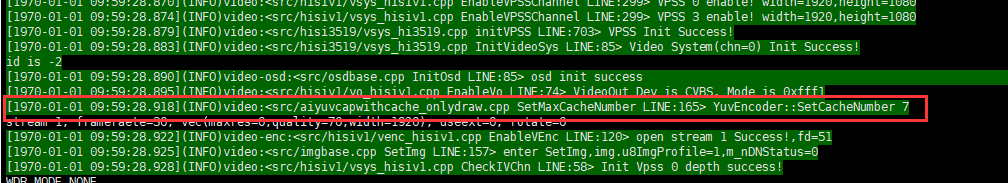
## 硬件AI下的缓存机制

如果externtype为0x40000或者0x80000, nsd会缓存固定的帧数， BT1120输出的第一个字节会被修改成帧ID， 范围是16-235， 之后AI模块分析完数据之后把结果返回， 需要把YUVAnalyzeResult里面的timestamp置为帧ID， 然后nsd就会根据ID在缓存中查找对应的帧然后进行画框或者抠图编码。

/mnt/mtd/nsd/config/setting/ai.conf中的cache\_num字段可以用作控制缓存大小。如果没有该字段，可以手动添加。



Nsd启动的时候这行打印表示缓存数量已经成功设置进去



目前缓存数量只能设置为2 – 8， 超过该范围的话， nsd会调整成该范围内的数值

externtype为0x40000： nsd会画框， 同时进行抠图

externtype为0x80000： nsd会画框， 不抠图

## 手动启动nsd没有问题， 通过脚本自动启动nsd则无法启动

检查libailib.so是否用到了其他动态库， 如果用到的话， 需要把相应的动态库在/lib下面创建软链接， 否则nsd启动的时候无法找到库

## Mipi\_tx接入屏幕无输出

1. 检查/mnt/mtd/nsd/config/setting/vou.conf 中outdev字段需要为5，on字段为1，screentype为0-3

0)： 1200\*1920 8寸屏 TV080WUM-NH1

1)： 800\*1280 10.1寸屏 JD9365\_BOE10.1

2)： 800\*1280 8寸屏 JD9366\_BOE8.0

3)： 800\*1280 7寸屏 JD9366\_BOE7.0

4)： 720\*1280 5寸屏 NT35523

5): 720\*1280 5寸屏 ST7703

1. 检查/mnt/mtd/nsd/config/setting/video.conf里面的rotate字段需要为3(sc2315e rotate为3, imx307 rotate字段为1)， mirror设置为2
2. 检测sys.def里面的hardware\_version字段要为2
3. 程序启动默认不拉LCD屏的背光， 如果需要显示， 需要在libailib.so模块内把GPIO6\_6拉高， 代码内可以调用IF\_IOCtl(NSDIO\_EXTIO,1,4,0);

## 设置framebuffer的alpha和colorkey

修改/mnt/mtd/nsd/config/setting/vout.conf, 比如修改0xff00ff为透明色， 则参考示例， 修改后重启设备

|  |
| --- |
| <?xml version="1.0" encoding="GBK"?>  <root>  <vout ch='0' outdev='5' outmode='65521' on='1' changetype='0' rotatemode='0' mirror='0' screentype='0' alpha='255' colorkeyenable='1' colorkey='0xff00ff'>  </vout>  </root> |

## 天瞳设备工作异常

1. sys.def中rs232字段需要为1， facetype需要为0x4404, externaltype需要为0x20020, channelnum为2, inner\_vi为8
2. uart.conf里面不应该配置/dev/ttyAMA3的设备
3. /mnt/mtd/code/fixed/tiantong/里面为标定文件目录，检查是否有标定文件

## 人脸FTP联动抓图的步骤及注意事项

<一>设置人脸基本参数及人脸图片的参数

1、开启人脸识别、设置人脸检测时间、人脸检测区域ROI、画框提示。

2、设置基本人脸参数

3、设置人脸图片的参数

<二>联动FTP抓图

1. 开启FTP并且填入正确参数,如图1所示



图1 FTP参数设置

1. 增加联动动作，选择联动方式为FTP，**注意：联动的FTP的间隔很小且联动的张数大于人脸抓拍的张数，确保上传给FTP时间充足**。如图2的联动动作

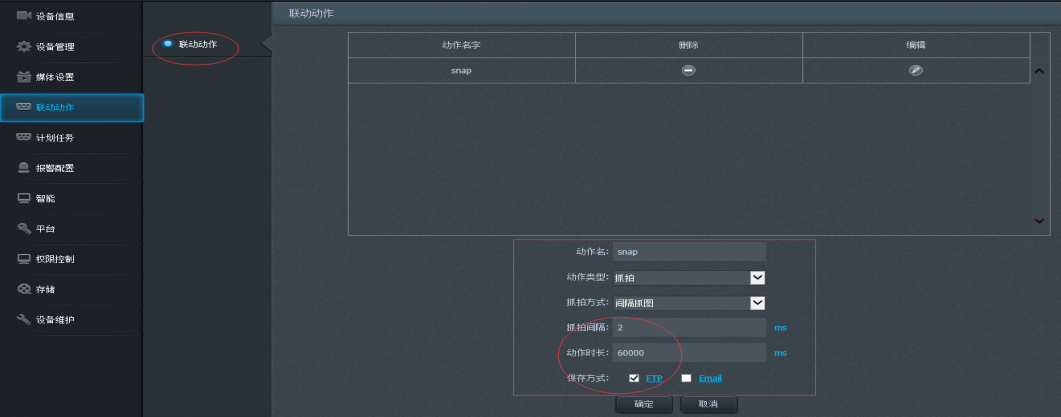


图2 的联动动作

附:图3人脸参数设置图



图3 人脸参数设置图

## 希望在nsd运行之后执行自己的脚本

华途的运行脚本为/mnt/mtd/nsd/program/bin/run.sh， nsd进程也是在该脚本中拉起来的， 如果客户希望在运行nsd之后做其他操作， 那么可以创建一个名字叫thirdback\_service.sh的脚本， 放在/mnt/mtd/nsd/program/bin/中。 这样设备启动时会自动调用该脚本。

注： 在2.8.3.2及之后版本支持该脚本

## 播放音频文件流程

1. 调用IF\_StartTalk接口打开音频输出
2. 读取音频文件，目前支持G711A、G711U、PCM格式的音频，采样率8K，单声道，16位，建议每次读取320个字节
3. 调用IF\_WriteTalkFrameToCache写入音频数据
4. 用完后调用IF\_StopTalk关闭音频输出

注：

1. IF\_WriteTalkFrameToCache《M\_INTERFACE\_SPEC.docx》内有调用示例
2. NSD\_FRAME\_HEAD\_S 内的u32FrameNo是内部检索标志，所以每帧音频该值务必递增例如第一次调用IF\_WriteTalkFrameToCache塞入320个字节，u32FrameNo为1，那么第二次调用IF\_WriteTalkFrameToCache再次塞入320个字节数据，u32FrameNo就要为2，以此类推
3. 建议在网页上把输出声音减小， 避免电流声过大
4. /mnt/mtd/code/fixed/sys.def 内enable\_talk字段必须为1

|  |
| --- |
| #define UART\_DATA\_SIZE 320  static void \*TheThread(void \*param)  {  CAiLib::GetInstance().m\_bThreadRunning = true;  while (!CAiLib::GetInstance().m\_stop\_flag)  {  int m\_frame\_no = 1;  CAiLib::GetInstance().IF\_StartTalk();  FILE\* fp = fopen("/root/222.pcm","rb");  fseek(fp,0L,SEEK\_END);  unsigned char buf[64\*1024] = {0};  int len=ftell(fp);  int times = (len%UART\_DATA\_SIZE==0)?len/UART\_DATA\_SIZE:len/UART\_DATA\_SIZE+1;  fseek(fp,0L,SEEK\_SET);  for(int i = 0 ; i < times ; i ++)  {  int readRet = fread(buf,1,UART\_DATA\_SIZE,fp);  if (readRet <= 0)  {  break;  }  NSD\_FRAME\_HEAD\_S head;  memset(&head,0,sizeof(NSD\_FRAME\_HEAD\_S));  head.u16FrameFlag = FE\_MAGIC\_FLAG;  head.u8FrameType = NSD\_FRAME\_A;  head.u32FrameNo = m\_frame\_no++;  head.u32TimeStamp = 0;  head.u32FrameSize = UART\_DATA\_SIZE;  head.audioInfo.u8EncodeType = NSD\_CODEC\_PCM;  head.audioInfo.u8AudioBits = 16;  head.audioInfo.u32AudioSamples = 8000;  CAiLib::GetInstance().IF\_WriteTalkFrameToCache(&head,buf,readRet);  usleep(10000);  }  fclose(fp);  CAiLib::GetInstance().IF\_StopTalk();  }  CAiLib::GetInstance().m\_bThreadRunning = false;  return 0;  } |

## 双目设备对灯光的控制

确认/mnt/mtd/code/fixed/sys.def里面hardware\_version需要为2

接口调用示例(需要在2.8.3.5及之后的版本使用)

|  |
| --- |
| //绿灯  IF\_IOCtl(NSDIO\_EXTIO,0,0,0);  IF\_IOCtl(NSDIO\_EXTIO,1,1,0);  //红灯  IF\_IOCtl(NSDIO\_EXTIO,1,0,0);  IF\_IOCtl(NSDIO\_EXTIO,0,1,0);  //关灯  IF\_IOCtl(NSDIO\_EXTIO,1,0,0);  IF\_IOCtl(NSDIO\_EXTIO,1,1,0);  //红外开  IF\_IOCtl(NSDIO\_EXTIO,1,2,0);  //红外关  IF\_IOCtl(NSDIO\_EXTIO,0,2,0);  //白光开  IF\_IOCtl(NSDIO\_EXTIO,1,3,0);  //白光关  IF\_IOCtl(NSDIO\_EXTIO,0,3,0);  //LCD屏背光开启 要在2.8.4.1及之后版本使用  IF\_IOCtl(NSDIO\_EXTIO,1,4,0);  //LCD屏背光关闭 要在2.8.4.1及之后版本使用  IF\_IOCtl(NSDIO\_EXTIO,0,4,0); |

## 重启设备后参数未被保存

重启设备需要调用接口CAiLib::GetInstance().IF\_ProcCmd(NSDCMD\_REBOOT, NULL,0, 0, 0);

不能直接调用reboot， 因为我们参数的保存周期是10秒， 所以如果中途异常重启的话， 会有可能没有保存到最后设置的参数

## 控制MIPI屏幕亮度

示例：

int brightness = 85;

IF\_ProcCmd(NSDCMD\_SCREEN\_LUMA,(void \*) brightness,0,0,0);

可以把屏幕亮度调节成85.

接口范围是0 – 100

## 客户自己开发AI模块编码后的图希望通过FTP上传

1. 配置好FTP的联动机制， 参考第30条
2. 有人脸信息后通过IF\_Alarm上报告警， 从而把联动参数传递给HVP
3. 人脸图编好以后塞入缓存

示例：

|  |
| --- |
| NSD\_SNAP\_HEAD\_S head;  memset(&head, 0, sizeof(NSD\_SNAP\_HEAD\_S));  head.u16Magic = 0x7470;  head.u8Type = NSD\_AI\_FACE;  head.u8Company = NSD\_COMPANY\_UNKNOWN;  head.u32SessionId=nSessionId;  head.u8TotleNum=nTotleNum;  head.u8No=nNo;  struct tm t;  dt.GetTM(t);  head.u32TimeStamp = mktime(&t);  head.u16MicroTimeStamp = dt.GetMilliSecond();    head.faceResult.u16FaceId = -1;  head.faceResult.u8IsOrigPicture=1;  head.faceResult.u8Quality = nQuality;    head.faceResult.resultPosition.u16X =0;  head.faceResult.resultPosition.u16Y =0;  head.faceResult.resultPosition.u16Width = nWidth;  head.faceResult.resultPosition.u16Height = nHeight;    CAiLib::GetInstance().IF\_Alarm(NSD\_ALARM\_FACE,1,0);  CAiLib::GetInstance().IF\_WritePictureFrame(nChn, &head, pJpg,nJpgLen); |

## AI模块希望感知日夜切换

1. 通过接口IF\_AddLocalMsgCallback注册本地消息
2. 日夜切换时会通过回调通知注册者， MsgType为NSD\_MSG\_DAY\_NIGHT\_SWITCH
3. 解析消息体， 格式为

<?xml version=”1.0”>

<root>

<msg>

<chn/>

<status/>

</msg>

</root>

Status值定义为

typedef enum \_NSD\_DAYNIGHT\_STATUS\_E

{

NSD\_DN\_STATUS\_NONE,

NSD\_DN\_STATUS\_DAY,

NSD\_DN\_STATUS\_NIGHT,

}NSD\_DAYNIGHT\_STATUS\_E;

## Wiegand驱动使用方式

/mnt/mtd/nsd/system/drivers/hi3516cv500\_wiegand.ko 为wiegand输入，目前只支持wiegand26

/mnt/mtd/nsd/system/drivers/hi3516cv500\_wiegandout.ko 为wiegand输出， 支持26和34

1). 韦根输出默认加载， 韦根输入默认不加载

2). 两个驱动不能共存，需要一个的时候要卸载另外一个

3). 输出的使用方式

echo 12345678 > /sys/sysfs\_weigand/wiegand34 发34的协议

echo 12345678 > /sys/sysfs\_weigand/wiegand26 发26的协议

4). 输入的使用方式

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

int main()

{

int fd = 0;

char dst[12] ={ 0};

int result =0;

fd = open("/dev/wiegand", O\_RDWR);

if(fd < 0)

{

printf("file open error ! \n");

return -1;

}

while(1)

{

result=read(fd, dst, sizeof(dst));

printf("dst0 is %s, size=%d\n", dst,result);

sleep(1);

}

close(fd);

return 0;

}