AX NT 开发指南

文档版本：V2.0

发布日期：2024/02/26

目 录

权利声明

注意

您购买的产品、服务或特性等应受商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非商业合同另有约定，本公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

前 言

适用产品

- AX650A/N
- AX620E 系列产品（AX630C、AX620Q）

适读人群

- 技术支持工程师
- 软件开发工程师

符号与格式定义

符号/格式	说明
xxx	表示您可以执行的命令行。
斜体	表示变量。如，“*安装目录*/AX_SDK_Vx.x.x/build 目录”中的“安装目录”是一个变量，由您的实际环境决定。
说明/备注：	表示您在使用产品的过程中，我们向您说明的事项。
注意：	表示您在使用产品的过程中，需要您特别注意的事项。

修订历史

文档版本	发布时间	修订说明
V1.0	2022/11/10	文档初版
V1.1	2022/11/11	1. 修改文档名称，去掉 API 相关字符; 2. 英文词语统一为大写
V1.2	2023/5/12	完善功能描述章节内容
V1.3	2023/5/29	1. SetSteamSource 时机调用说明 2. 添加 Log 开启方法说明
V2.0	2024/02/26	AX620E、AX650 文档合并，在相关内容处添加 AX630C/AX620Q 区别内容说明

1 概述

1.1 概述

NetTransfer（简称 NT）通过 Socket 的 Client/Server 的方式实现 PC 客户端与设备侧 Server 的通信。支持通过 TCP 协议建立客户端和设备端的连接。

1.2 重要概念

表1-1 重要概念

缩写	全称
NT	Net Transfer 网络数据流通信模块
DEV	视频输入设备
PIPE	PIPE 包含了 ISP Pipeline 的所有模块，是 Pipeline 在软件层面的统称，负责对 DEV 出来的数据进行流水线处理

1.3 功能介绍

1.3.1 功能描述

整个 ISP 参数调试和预览，分为 PC 端的 ispTuning 工具和 NT 设备端两个部分。NT 是设备端的服务，需要用户在程序中集成 NT 相关的接口，用户程序在初始化时调用 NT 接口即完成了 NT 在设备端的配置过程。

NT 包括两个库，分别是 CTRL 库和 STREAM 库。CTRL 库各接口函数所在头文件见 SDK 中的 ax_nt_ctrl_api.h（在 msp/out/include 路径下），STREAM 库各接口函数所在头文件见 SDK 中的 ax_nt_stream_api.h（在 msp/out/include 路径下）。

- **Ctrl 库**

提供设置/获取 ISP 各模块参数、获取版本信息、灌图等功能。其中灌图支持单路灌图。

- **Stream 库**

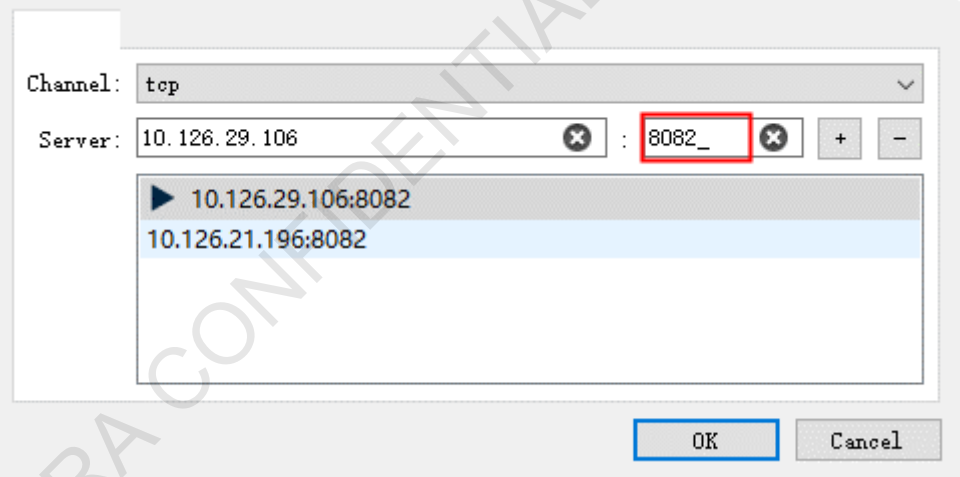
提供预览和抓图的功能，支持多种格式和多个通道的视频流预览（包括多个节点的 RAW 数据、VIN CHN 输出的 YUV 数据、编码之后的 H.264 码流），支持连续抓图和单张抓图。

! 注意：连续抓图模式下，可抓取的最大张数，取决板端内存大小以及分配的 CMM 空间大小。

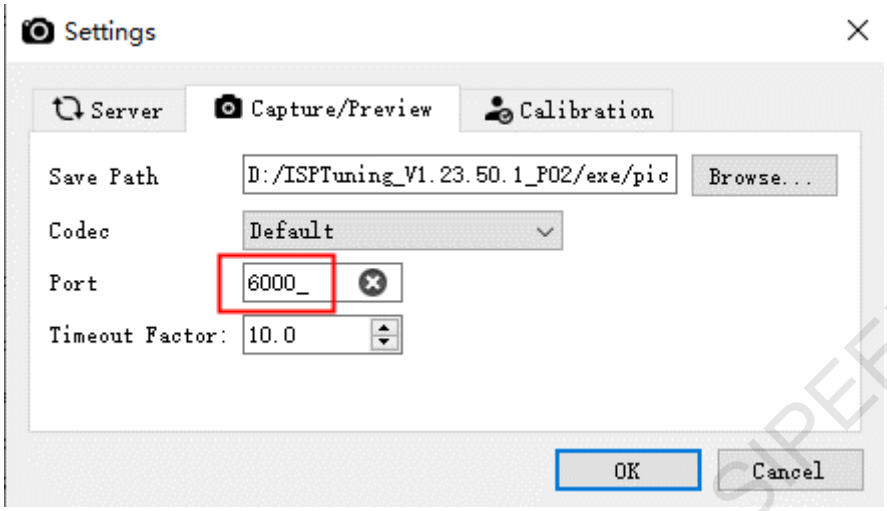
需要注意的是，DEV 属性中的 eDevMode 字段配置为 ONLINE 模式时，不支持 DEV 节点的预览和抓图。DEV 节点的含义请参考文档《AX VIN API 开发指南.docx》。

- **端口号**

Ctrl 库的默认端口号是 8082，端口号支持用户自定义，用户在调用 AX_NT_CtrlInit 接口时，设置需要的端口号即可。



Stream 库的默认端口号是 6000，端口号支持用户自定义，用户在调用 AX_NT_StreamInit 接口时，设置需要的端口号即可。



! 注意：

- 如果修改默认端口号，在 ispTuning 工具连接时候要保证工具设置的端口号与自定义端口号保持一致，否则无法连接板端！
- ispTuning 界面中端口号后面的下划线无需输入！

NT 初始化流程如下图所示，用户在开发时，可参考 SDK 中的源代码 sample_vin.c（在 msp/sample/vin 路径下）。

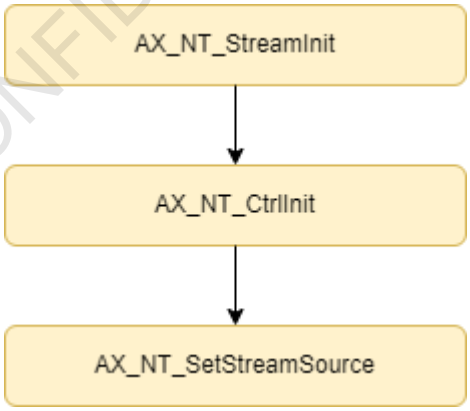


图1-1 NT 模块初始化流程图

! 注意：

用户如果需要集成 NT 的服务，设备端的 CTRL 库和 STREAM 库的初始化接口都需要调用，单独调用其中一个库的接口，会存在连接失败的问题。

2 API 参考

AX_NT_CtrlInit

【描述】

初始化 NT 控制的端口。

【语法】

AX_S32 AX_NT_CtrlInit(AX_U32 nPort)

【参数】

参数名称	描述	输入/输出
nPort	NT 控制的端口	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码

【需求】

- 头文件：ax_nt_ctrl_api.h
- 库文件：libax_nt_ctrl.so

【注意】

该接口指定的端口号，需要与 PC ispTuning 工具指定的端口号一致，端口号为默认的 8082，如果修改了此端口号，则在连接时，ispTuning 工具修改为对应的端口号即可。

【举例】

无

【相关主题】

无

AX_NT_CtrlDeInit

【描述】

退出 NT 控制的端口。

【语法】

AX_S32 AX_NT_CtrlDeInit(void)

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码

【需求】

- 头文件：ax_nt_ctrl_api.h
- 库文件：libax_nt_ctrl.so

【注意】

无

【举例】

无

【相关主题】

无

AX_NT_StreamInit

【描述】

初始化 NT 预览流的端口。

【语法】

AX_S32 AX_NT_StreamInit(AX_U32 nStreamPort)

【参数】

参数名称	描述	输入/输出
nStreamPort	NT 预览流的端口	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码

【需求】

- 头文件：ax_nt_stream_api.h
- 库文件：libax_nt_stream.so

【注意】

该接口指定的端口号，为 SDK 的默认值 6000。如果修改了此端口，则需要在 ispTuning 工具中修改对应的端口号。

【举例】

无

【相关主题】

无

AX_NT_StreamDeInit

【描述】

退出 NT 预览流的端口。

【语法】

AX_S32 AX_NT_StreamDeInit(void)

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码

【需求】

- 头文件：ax_nt_stream_api.h
- 库文件：libax_nt_stream.so

【注意】

无

【举例】

无

【相关主题】

无

AX_NT_SetStreamSource

【描述】

设置预览数据流的来源。

【语法】

AX_S32 AX_NT_SetStreamSource(AX_U8 pipe)

【参数】

参数名称	描述	输入/输出
pipe	Pipe id	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码

【需求】

- 头文件：ax_nt_stream_api.h
- 库文件：libax_nt_stream.so

【注意】

1. 该接口内部会获取 VIN 模块的 PIPE/CHN 属性，以确定哪些数据流处于使能状态。处于使能状态的节点，在 PC 工具端会显示出来。需要相关接口配置完才能调用。
2. 需要在 Stream 和 Ctrl 两个模块初始化接口调用之后调用。
3. 此接口一般需要在 VIN 初始化之后调用。

【举例】

无

【相关主题】

无

3 LOG 开启方式

3.1 具体操作方法

3.1.1 查看模块打印 Level 和 Target

通过 cat /proc/ax_proc/logctl 命令查看对应 ID 的打印 Level，不同平台，ID 会有所不同，如(AX650A/N 平台的 NT 模块 ID 为 35，打印 Level 为 4)：

```
/root # cat /proc/ax_proc/logctl | tail -n 20
      IVPS    13     4         4
      MIPI    14     4         4
      ADEC    15     4         4
       DMA    16     4         4
       VIN    17     4         4
      USER    18     4         4
      IVES    19     4         4
      SKEL    20     4         4
       IVE    21     4         4
        3A    25     4         4
     AUDIO    26     4         4
        AI    32     4         4
        AO    33     4         4
    SENSOR    34     4         4
        NT    35     4         4
       TDP    36     4         4
       VPP    37     4         4
       VGP    38     4         4
       GDC    39     4         4
      BASE    40     4         4
/root #
```

通过 cat /proc/ax_proc/logctl 命令查看对应 ID 的打印 Target，如

```
/root # cat /proc/ax_proc/logctl
*************  Axera log control *************
Usage:
  echo [ulog/klog] [id] [level] > /proc/ax_proc/logctl
  ulog: user log klog: kernel log
       id: [0-40, all] level: [0-7])
  echo ulog/klog/all [on, off] > /proc/ax_proc/logctl
  echo ulog [file, console, null] > /proc/ax_proc/logctl(user log only)
-------------------------------------------
  klog_state:  on
  ulog_state:  on
  ulog_target: file
-------------------------------------------
    module    id  level(U)  level(K)
       ISP     1     4         4
```

可以看到 ulog_target 的目标为 file。

3.1.2 设置 Level 等级和 Target 目标

通过 echo ulog [id] [level] > /proc/ax_proc/logctl 命令修改对应等级，如：

```
oot #  echo ulog 35 7 > /proc/ax_proc/logctl
oot #
oot #
oot # cat /proc/ax_proc/logctl | tail -n 20
    IVPS    13     4         4
    MIPI    14     4         4
    ADEC    15     4         4
     DMA    16     4         4
     VIN    17     4         4
    USER    18     4         4
    IVES    19     4         4
    SKEL    20     4         4
     IVE    21     4         4
      3A    25     4         4
   AUDIO    26     4         4
      AI    32     4         4
      AO    33     4         4
  SENSOR    34     4         4
      NT    35     7         4
     TDP    36     4         4
     VPP    37     4         4
     VGP    38     4         4
     GDC    39     4         4
    BASE    40     4         4
```

可以看到 NT 对应的 User 层的 Level 已经被修改为 7。

通过 echo ulog console > /proc/ax_proc/logctl 命令修改 Target 目标为终端屏幕，如：

```
/root # echo ulog console > /proc/ax_proc/logctl
/root #
/root #
/root # cat /proc/ax_proc/logctl
*************  Axera log control ************
Usage:
  echo [ulog/klog] [id] [level] > /proc/ax_proc/logctl
  ulog: user log klog: kernel log
       id: [0-40, all] level: [0-7])
  echo ulog/klog/all [on, off] > /proc/ax_proc/logctl
  echo ulog [file, console, null] > /proc/ax_proc/logctl(user log only)
-----------------------------------------
  klog_state:  on
  ulog_state:  on
  ulog_target: console
```

可以看到 ulog_target 从文件修改为 console，且 NT 模块打印也已经打印到串口，如：

```
64821.992078] [AXVENC][W][13460] [W][AX_VENC_CreateChn][379]: VencChn 2: create encoder channel success.
64821.992153] [NT][D][13460] [D][                AX_NT_CtrlInit][  32]: CTRL: AX_NT_Ctrl Build at May 26 2023 13:51:55
64821.992309] [NT][I][13460] [I][           __net_transfer_init][  80]: __mainloop run
 = 14
64821.992399] [NT][I][13460] [I][                start_listen][  53]: start listen ok ,Listen to port[8082]
64821.992947] [NT][I][13460] [I][           __net_transfer_init][  80]: __mainloop run
 = 15
64821.993056] [NT][I][13460] [I][                start_listen][  53]: start listen ok ,Listen to port[6000]
 80
64821.993103] [NT][D][13460] [D][     __stream_plat_find_node_idx][  20]: __stream_plat_find_node_idx 20: idx = 0
 80
64821.993129] [NT][D][13460] [D][     __stream_plat_find_node_idx][  20]: __stream_plat_find_node_idx 20: idx = 1
0
64821.993151] [NT][D][13460] [D][     __stream_plat_find_node_idx][  20]: __stream_plat_find_node_idx 20: idx = 2
64821.993190] [NT][D][13460] [D][     __stream_plat_find_node_idx][  20]: __stream_plat_find_node_idx 20: idx = 3
64821.993226] [NT][D][13460] [D][     __stream_plat_find_node_idx][  20]: __stream_plat_find_node_idx 20: idx = 4
64821.993260] [NT][D][13460] [D][              __nt_update_source][ 116]: ax_stream_plat_get_pipe_ai3dnr_ability failed! pipe
64821.993309] [NT][D][13460] [D][              __nt_update_source][ 121]: ax_stream_plat_get_pipe_rltm_ability failed! pipe =
64821.993327] [NT][D][13460] [D][              __nt_update_source][ 126]: ax_stream_plat_get_pipe_hdr_ability failed! pipe =
64821.993373] [NT][D][13460] [D][     __stream_plat_find_node_idx][  20]: __stream_plat_find_node_idx 20: idx = 5
```

可以看到比我们设置的小的 Level 都打印出来了。

4 错误码

无