



HiSVP

API 参考

文档版本 09

发布日期 2021-01-28

版权所有 © 上海海思技术有限公司 2021。保留一切权利。

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司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。除非合同另有约定，海思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。

上海海思技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： <http://www.hisilicon.com/cn/>

客户服务邮箱： support@hisilicon.com



前 言

概述

本文档为使用上海海思媒体处理芯片的 SVP 平台智能分析方案开发的程序员而写，目的是供您在开发过程中查阅 SVP 支持的各种参考信息，包括 API、头文件、错误码等。

说明

- 未有特殊说明，Hi3559CV100，Hi3569V100 与 Hi3559AV100 内容一致。
- 未有特殊说明，Hi3568V100 与 Hi3519AV100 内容一致。
- 未有特殊说明，Hi3516AV300，Hi3516DV300，Hi3562V100，Hi3566V100 与 Hi3516CV500 内容一致。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3516C	V500
Hi3516D	V300
Hi3559	V200
Hi3516A	V300
Hi3562	V100
Hi3566	V100



产品名称	产品版本
Hi3569	V100
Hi3568	V100






读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。



修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	版本	修改描述
2021-01-28	09	1.3 小节, 新增 HI_MPI_SVP_DSP_Query_Timeout 2.3 小节, 新增 HI_MPI_SVP_NNIE_Query_Timeout
2020-06-08	08	添加 Hi3568V100 相关内容
2020-03-31	07	1.3 小节, 新增 HI_MPI_SVP_DSP_SetConfigAttr 和 HI_MPI_SVP_DSP_GetConfigAttr 1.4 小节, 新增 SVP_DSP_CONFIG_ATTR_S 2.3 小节, HI_MPI_SVP_NNIE_Forward【注意】涉及修改
2019-11-30	06	添加 Hi3562V100 和 Hi3566V200 芯片
2019-06-25	05	2.1 小节中的说明涉及修改 3.3 小节, 新增 HI_NodePlugin_getNodeType 和 HI_NodePlugin_Compute, 每个接口中的【举例】涉及 修改 3.4 小节, 新增 HI_NodePlugin_Shape_S、 HI_NodePlugin_ElemType_E、 HI_NodePlugin_Operand_S 和 HI_NodePlugin_NodeParam_S
2019-05-20	04	3.5 小节涉及修改
2019-04-30	03	新增 1.2.2 小节 1.6.2 小节涉及修改 2.4 小节, SVP_BLOB_S 和 SVP_NNIE_NODE_S【成员】 涉及修改



日期	版本	修改描述
2019-03-12	02	2.1 小节新增 Hi3559V200 NNIE 与 GDC 互斥说明 2.3 小节, HI_MPI_SVP_NNIE_LoadModel【注意】涉及修改; HI_MPI_SVP_NNIE_Forward 和 HI_MPI_SVP_NNIE_ForwardWithBbox【参数】涉及修改 2.4 小节, SVP_BLOB_S、SVP_NNIE_NODE_S、 SVP_NNIE_FORWARD_CTRL_S 和 SVP_NNIE_FORWARD_WITHBBOX_CTRL_S【成员】涉及修改 3.3 小节, HI_SVPRT_RUNTIME_Init【参数】涉及修改 3.6.2 小节涉及修改 添加 Hi3559V200 芯片支持
2018-12-10	01	1.4 小节, 新增 SVP_DSP_HANDLE 2.4 小节, SVP_NNIE_NODE_S【定义】和【成员】涉及修改; 新增 SVP_NNIE_NODE_NAME_LEN 2.5 小节, 表 2-2 涉及修改
2018-11-12	00B09	3.3 小节, HI_SVPRT_RUNTIME_LoadModelGroupSync、 HI_SVPRT_RUNTIME_ForwardGroupSync、 HI_SVPRT_RUNTIME_ForwardGroupASync、 HI_SVPRT_RUNTIME_UnloadModelGroup【语法】和 【参数】涉及修改 3.4 小节, HI_RUNTIME_GROUP_HANDLE 涉及修改; 新增 HI_RUNTIME_FORWARD_STATUS_CALLBACK_E、 HI_RUNTIME_Forward_Callback 和 HI_RUNTIME_Connector_Compute



日期	版本	修改描述
2018-10-15	00B08	2.3 小节, HI_MPI_SVP_NNIE_Forward、 HI_MPI_SVP_NNIE_ForwardWithBbox、 HI_MPI_SVP_NNIE_AddTskBuf 和 HI_MPI_SVP_NNIE_RemoveTskBuf 【注意】涉及修改 2.4 小节, SVP_NNIE_FORWARD_CTRL_S 和 SVP_NNIE_FORWARD_WITHBBOX_CTRL_S 【注意】涉及 修改 新增第 4 章
2018-09-04	00B07	新增第 3 章节 2.3 小节, 新增 HI_MPI_SVP_NNIE_AddTskBuf 和 HI_MPI_SVP_NNIE_RemoveTskBuf 2.6 小节, 新增 NNIE Proc 调试信息
2018-07-30	00B06	1.5 小节, 表 1-1 涉及修改 新增 Hi3556AV100 相关内容。
2018-05-20	00B05	2.3 小节, HI_MPI_SVP_NNIE_GetTskBufSize 【参数】涉 及修改 2.4 小节, SVP_NNIE_SEG_S、SVP_NNIE_MODEL_S 和 SVP_NNIE_FORWARD_CTRL_S 【成员】涉及修改
2018-04-13	00B04	添加 Hi3519AV100 的相关内容
2018-02-10	00B03	1.3 小节, 新增 HI_MPI_SVP_DSP_PowerOn 和 HI_MPI_SVP_DSP_PowerOff 2.3 小节, HI_MPI_SVP_NNIE_CNN_ForwardWithBbox 涉及修改
2018-01-29	00B02	1.4 小节, SVP_DSP_MEM_TYPE_E 【定义】涉及修改 2.6.2 小节涉及修改
2018-01-10	00B01	第一次临时版本发布



目 录

1 DSP	1
1.1 概述	1
1.2 功能描述	1
1.2.1 重要概念	1
1.2.2 模块参数	1
1.3 API 参考	2
1.4 数据类型和数据结构	13
1.5 错误码	21
1.6 Proc 调试信息	22
1.6.1 概述	22
1.6.2 Proc 信息说明	23
2 NNIE	27
2.1 概述	27
2.2 功能描述	27
2.2.1 重要概念	27
2.2.2 使用示意	35
2.3 API 参考	36
2.4 数据类型和数据结构	53
2.5 错误码	75
2.6 Proc 调试信息	76
2.6.1 概述	76
2.6.2 Proc 信息说明	77
3 Runtime	81
3.1 概述	81



3.2 功能描述	81
3.2.1 重要概念.....	81
3.3 API 参考	81
3.4 数据类型和数据结构.....	93
3.5 错误码.....	117
3.6 Proc 调试信息.....	118
3.6.1 概述.....	118
3.6.2 Proc 信息说明	118



插图目录

图 2-1 跨度 (stride) 示意图.....	28
图 2-2 SVP_BLOB_TYPE_S32 类型 SVP_BLOB_S (2 通道 2 帧示意图)	31
图 2-3 SVP_BLOB_TYPE_U8 类型 SVP_BLOB_S (3 通道 2 帧示意图)	32
图 2-4 SVP_BLOB_TYPE_YVU420SP 类型 SVP_BLOB_S (2 帧 YVU420SP 示意图)	33
图 2-5 SVP_BLOB_TYPE_YVU422SP 类型 SVP_BLOB_S (2 帧 YVU422SP 示意图)	34
图 2-6 SVP_BLOB_TYPE_VEC_S32 类型 SVP_BLOB_S (2 帧示意图)	34
图 2-7 SVP_BLOB_TYPE_SEQ_S32 类型 SVP_BLOB_S(Num=N 帧示意图).....	35
图 2-8 SVP_MEM_INFO_S 类型的数据内存示意	35
图 2-9 NNIE_Forward 支持的多节点输入输出网络示意图	41
图 2-10 NNIE_ForwardWithBbox 支持的输入输出网络示意图	45
图 2-11 NNIE_ForwardWithBbox astBbox[i]输入示意图	45
图 2-12 NNIE_ForwardWithBbox Score 输出示意图	46
图 2-13 NNIE_ForwardWithBbox Bbox 调整值输出示意图 1.....	46
图 2-14 NNIE_ForwardWithBbox Bbox 调整值输出示意图 2.....	46
图 2-15 NNIE_ForwardWithBbox Bbox 调整值输出示意图 3.....	47
图 3-1 模型组输入输出网络示意图	86



表目录

表 1-1 DSP 模块 API 错误码	21
表 2-1 BLOB 内存排布类型表	29
表 2-2 NNIE 引擎 API 错误码	75



1 DSP

1.1 概述

SVP (Smart Vision Processing) 平台是上海海思媒体处理芯片智能视觉异构加速平台。DSP (Digital Signal Process) 是 SVP 平台下的可编程硬件加速模块。用户基于 DSP 开发智能分析方案可以加速智能分析, 降低 CPU 占用。



说明

Hi3516CV500/Hi3516DV300/Hi3559V200 不支持 DSP。

1.2 功能描述

1.2.1 重要概念

- 句柄(handle)
用户在调用 DSP 处理任务时, 系统会为每个任务分配一个 handle, 用于标识不同的任务。
- 查询(query)
用户根据系统返回的 handle, 调用 HI_MPI_SVP_DSP_Query 可以查询对应算子任务是否完成。

1.2.2 模块参数

- max_node_num
DSP 模块各个优先级队列节点最大数目。用户可以依据实际使用场景配置合理的最大节点数, 以减少队列节点对系统内存的占用。取值范围[20, 51]。默认值为 30。



使用方法:

- Linux 系统通过加载 hi35xx_dsp.ko 时设置 max_node_num。
- Huawei LiteOS 系统通过 DSP_init 函数（在文件 sdk_init.c 中）设置参数 u16NodeNum 的值。
- dsp_init_mode
DSP 模块初始化模式。用户可以依据业务实际情况配置不同初始化模式，取值只有 0 或者 1 两个值。
 - 初始化模式取值为 0 时，只要整个媒体系统退出都需要调用 DSP 相关的下电，关闭 DSP 接口才能退出系统，重新使用 DSP 时，需要按照正常流程上电，加载 DSP 镜像，使能 DSP。
 - 初始化模式取值为 1 时，整个媒体系统退出时，不需要执行 DSP 的关闭和下电接口，保持 DSP 镜像系统是运行状态。重新使用 DSP 时，不需要再次加载 DSP 镜像。

使用方法:

- Linux 系统通过加载 hi35xx_dsp.ko 时设置 dsp_init_mode。
- Huawei LiteOS 系统通过 DSP_init 函数（在文件 sdk_init.c 中）设置参数 u16DspInitMode 的值。

1.3 API 参考

上海海思 DSP ARM 端模块 API 接口操作。

提供以下 API:

- [HI_MPI_SVP_DSP_PowerOn](#): DSP 上电。
- [HI_MPI_SVP_DSP_PowerOff](#): DSP 下电。
- [HI_MPI_SVP_DSP_LoadBin](#): 加载 DSP Bin。
- [HI_MPI_SVP_DSP_EnableCore](#): 使能 DSP 核，使其工作。
- [HI_MPI_SVP_DSP_DisableCore](#): 去使能 DSP 核，使其停止工作。
- [HI_MPI_SVP_DSP_RPC](#): 远程处理任务。
- [HI_MPI_SVP_DSP_Query](#): 查询任务完成情况。
- [HI_MPI_SVP_DSP_Query_Timeout](#): 查询任务完成情况，超时时间由上层用户控制。
- [HI_MPI_SVP_DSP_SetConfigAttr](#): 设置配置属性。



- [HI_MPI_SVP_DSP_GetConfigAttr](#): 获取配置属性。

HI_MPI_SVP_DSP_PowerOn

【描述】

DSP 上电。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_PowerOn(SVP\_DSP\_ID\_E enDspld);
```

【参数】

参数名称	描述	输入/输出
enDspld	DSP ID 号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

在加载 DSP 镜像前，必须先调用 [HI_MPI_SVP_DSP_PowerOn](#) 接口使 DSP 上电，否则加载镜像时会挂死。

【举例】

无。

【相关主题】

无。



HI_MPI_SVP_DSP_PowerOff

【描述】

DSP 下电。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_PowerOff(SVP\_DSP\_ID\_E enDspId);
```

【参数】

参数名称	描述	输入/输出
enDspId	DSP ID 号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

无。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_DSP_LoadBin

【描述】



加载 DSP Bin。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_LoadBin(const HI_CHAR *pszBinFileName, SVP_DSP_MEM_TYPE_E  
enMemType);
```

【参数】

参数名称	描述	输入/输出
pszBinFileName	Bin 文件名。 不能为空。	输入
enMemType	DSP 内存类型。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

用户要保证加载文件的完整性，否则可能出现 DSP 启动失败、系统挂死等问题。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_DSP_EnableCore

【描述】



使能 DSP 核, 使其工作。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_EnableCore(SVP_DSP_ID_E enDspld);
```

【参数】

参数名称	描述	输入/输出
enDspld	DSP ID 号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 参见 错误码 。

【需求】

- 头文件: hi_dsp.h、mpi_dsp.h
- 库文件: libdsp.a

【注意】

无。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_DSP_DisableCore

【描述】

去使能 DSP 核, 使其停止工作。

【语法】



```
HI_S32 HI_MPI_SVP_DSP_DisableCore(SVP\_DSP\_ID\_E enDspId);
```

【参数】

参数名称	描述	输入/输出
enDspId	DSP ID 号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

无。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_DSP_RPC

【描述】

远程处理任务。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_RPC(SVP_DSP_HANDLE *phHandle, const SVP\_DSP\_MESSAGE\_S  
*pstMsg, SVP\_DSP\_ID\_E enDspId, SVP\_DSP\_PRI\_E enPri);
```



【参数】

参数名称	描述	输入/输出
phHandle	handle 指针。 不能为空。	输出
pstMsg	处理消息体。 不能为空。	输入
enDspld	DSP Id 号。	输入
enPri	任务优先级。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

无。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_DSP_Query

【描述】



查询任务完成情况。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_Query(SVP\_DSP\_ID\_E enDspld, SVP\_DSP\_HANDLE hHandle, HI_BOOL  
*pbFinish, HI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
enDspld	DSP Id 号。	输入
hHandle	任务的 handle。	输入
pbFinish	任务完成状态指针。 不能为空。	输出
bBlock	是否阻塞查询标志。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

无。

【举例】

无。

【相关主题】

[HI_MPI_SVP_DSP_Query_Timeout](#)



HI_MPI_SVP_DSP_Query_Timeout

【描述】

查询任务完成情况，超时时间由上层用户控制。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_Query_Timeout(SVP_DSP_ID_E enDspld, SVP_DSP_HANDLE hHandle,  
HI_S32 s32MilliSec, HI_BOOL *pbFinish);
```

【参数】

参数名称	描述	输入/输出
enDspld	DSP Id 号。	输入
hHandle	任务的 handle。	输入
s32MilliSec	超时时间，单位毫秒。	输入
pbFinish	任务完成状态指针。 不能为空。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

当 s32MilliSec 设为-1 时，表示阻塞模式，程序一直等待，直到查询的任务完成或者系统出错才返回。

- 如果 s32MilliSec 等于 0 时，表示非阻塞模式。



- 如果 s32MilliSec 大于 0 时，表示超时等待模式，参数的单位是毫秒，指超时时间，在此时间内如果任务没有完成，则超时返回。

【举例】

无。

【相关主题】

[HI_MPI_SVP_DSP_Query](#)

HI_MPI_SVP_DSP_SetConfigAttr

【描述】

设置配置属性。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_SetConfigAttr(SVP\_DSP\_ID\_E enDspld, const SVP\_DSP\_CONFIG\_ATTR\_S *pstConfigAttr);
```

【参数】

参数名称	描述	输入/输出
enDspld	DSP Id 号。	输入
pstConfigAttr	配置属性指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【芯片差异】

芯片类型	是否支持该功能
Hi3559AV100/Hi3519AV100	支持



Hi3568V100/Hi3569V100	不支持
-----------------------	-----

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

- 该接口需要在 [HI_MPI_SVP_DSP_PowerOn](#) 被调用之前调用。接口不是必须要调用来设置配置属性的，只有当需要修改 memmap.xmm 文件（该文件在发布包 dsp/dspxx/liteos/dspxx_ldscripts 目录下，xx 代表数字）的 SRAM 起始地址或者修改 DSP 的 IDMA 访问 DDR 地址不依赖 MMZ 管理时才需要调用。
- 如果修改 memmap.xmm 文件的 SRAM 地址范围大小超过发布版本配置的地址范围，请同步修改头文件 svp_dsp_config.h(该头文件在发布包 dsp\dspxx\include 目录下，xx 代表数字)的宏定义 SVP_DSP_ICACHE_MPU_START_ADDR 和 SVP_DSP_ICACHE_MPU_END_ADDR 的值，默认值是整个 DSP 镜像空间的地址区域。

【举例】

无。

【相关主题】

[HI_MPI_SVP_DSP_GetConfigAttr](#)

HI_MPI_SVP_DSP_GetConfigAttr

【描述】

获取配置属性。

【语法】

```
HI_S32 HI_MPI_SVP_DSP_GetConfigAttr(SVP\_DSP\_ID\_E enDspId, SVP\_DSP\_CONFIG\_ATTR\_S
*pstConfigAttr);
```

【参数】

参数名称	描述	输入/输出
enDspId	DSP Id 号。	输入



参数名称	描述	输入/输出
pstConfigAttr	配置属性指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【芯片差异】

芯片类型	是否支持该功能
Hi3559AV100/Hi3519AV100	支持
Hi3568V100/Hi3569V100	不支持

【需求】

- 头文件：hi_dsp.h、mpi_dsp.h
- 库文件：libdsp.a

【注意】

无。

【举例】

无。

【相关主题】

[HI_MPI_SVP_DSP_SetConfigAttr](#)

1.4 数据类型和数据结构

DSP 相关数据类型、数据结构定义如下：



- [SVP_DSP_HANDLE](#): 定义 DSP 句柄。
- [SVP_DSP_ID_E](#): 定义 DSP ID。
- [SVP_DSP_PRI_E](#): 定义优先级。
- [SVP_DSP_MEM_TYPE_E](#): 定义内存类型。
- [SVP_DSP_CMD_E](#): 定义命令。
- [SVP_DSP_MESSAGE_S](#): 定义消息格式。
- [SVP_DSP_CONFIG_ATTR_S](#): 定义配置属性。

SVP_DSP_HANDLE

【说明】

定义 DSP 句柄。

【定义】

```
typedef HI_S32 SVP_DSP_HANDLE;
```

【成员】

成员名称	描述
SVP_DSP_HANDLE	DSP 句柄。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_DSP_ID_E

【说明】

定义 DSP ID。

【定义】

Hi3559AV100:

```
typedef enum hiSVP_DSP_ID_E
```



```
{  
    SVP_DSP_ID_0 = 0x0,  
    SVP_DSP_ID_1 = 0x1,  
    SVP_DSP_ID_2 = 0x2,  
    SVP_DSP_ID_3 = 0x3,  
  
    SVP_DSP_ID_BUTT  
}SVP_DSP_ID_E;
```

Hi3519A V100/Hi3556AV100:

```
typedef enum hiSVP_DSP_ID_E  
{  
    SVP_DSP_ID_0 = 0x0,  
    SVP_DSP_ID_BUTT  
}SVP_DSP_ID_E;
```

【成员】

成员名称	描述
SVP_DSP_ID_0	DSP ID 0。
SVP_DSP_ID_1	DSP ID 1。
SVP_DSP_ID_2	DSP ID 2。
SVP_DSP_ID_3	DSP ID 3。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_DSP_PRI_E

【说明】

定义优先级。

【定义】



```
typedef enum hiSVP_DSP_PRI_E
{
    SVP_DSP_PRI_0 = 0x0,
    SVP_DSP_PRI_1 = 0x1,
    SVP_DSP_PRI_2 = 0x2,
    SVP_DSP_PRI_BUTT
}SVP_DSP_PRI_E;
```

【成员】

成员名称	描述
SVP_DSP_PRI_0	优先级 0 最高。
SVP_DSP_PRI_1	优先级 1。
SVP_DSP_PRI_2	优先级 2。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_DSP_MEM_TYPE_E

【说明】

定义内存类型。

【定义】

Hi3559AV100:

```
typedef enum hiSVP_DSP_MEM_TYPE_E
{
    SVP_DSP_MEM_TYPE_SYS_DDR_DSP_0 = 0x0,
    SVP_DSP_MEM_TYPE_IRAM_DSP_0    = 0x1,
    SVP_DSP_MEM_TYPE_DRAM_0_DSP_0  = 0x2,
    SVP_DSP_MEM_TYPE_DRAM_1_DSP_0  = 0x3,
```



```
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_1 = 0x4,  
SVP_DSP_MEM_TYPE_IRAM_DSP_1    = 0x5,  
SVP_DSP_MEM_TYPE_DRAM_0_DSP_1  = 0x6,  
SVP_DSP_MEM_TYPE_DRAM_1_DSP_1  = 0x7,  
  
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_2 = 0x8,  
SVP_DSP_MEM_TYPE_IRAM_DSP_2    = 0x9,  
SVP_DSP_MEM_TYPE_DRAM_0_DSP_2  = 0xA,  
SVP_DSP_MEM_TYPE_DRAM_1_DSP_2  = 0xB,  
  
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_3 = 0xC,  
SVP_DSP_MEM_TYPE_IRAM_DSP_3    = 0xD,  
SVP_DSP_MEM_TYPE_DRAM_0_DSP_3  = 0xE,  
SVP_DSP_MEM_TYPE_DRAM_1_DSP_3  = 0xF,  
  
SVP_DSP_MEM_TYPE_BUTT  
}SVP_DSP_MEM_TYPE_E;
```

Hi3519AV100/Hi3556AV100:

```
typedef enum hiSVP_DSP_MEM_TYPE_E  
{  
    SVP_DSP_MEM_TYPE_SYS_DDR_DSP_0 = 0x0,  
    SVP_DSP_MEM_TYPE_IRAM_DSP_0    = 0x1,  
    SVP_DSP_MEM_TYPE_DRAM_0_DSP_0  = 0x2,  
    SVP_DSP_MEM_TYPE_DRAM_1_DSP_0  = 0x3,  
    SVP_DSP_MEM_TYPE_BUTT  
}SVP_DSP_MEM_TYPE_E;
```

【成员】

成员名称	描述
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_0	DSP0 使用的系统 DDR 内存地址空间。
SVP_DSP_MEM_TYPE_IRAM_DSP_0	DSP0 内部 IRAM 地址空间。
SVP_DSP_MEM_TYPE_DRAM_0_DSP_0	DSP0 内部 DRAM0 地址空间。
SVP_DSP_MEM_TYPE_DRAM_1_DSP_0	DSP0 内部 DRAM1 地址空间。



成员名称	描述
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_1	DSP1 使用的系统 DDR 内存地址空间。
SVP_DSP_MEM_TYPE_IRAM_DSP_1	DSP1 内部 IRAM 地址空间。
SVP_DSP_MEM_TYPE_DRAM_0_DSP_1	DSP1 内部 DRAM0 地址空间。
SVP_DSP_MEM_TYPE_DRAM_1_DSP_1	DSP1 内部 DRAM1 地址空间。
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_2	DSP2 使用的系统 DDR 内存地址空间。
SVP_DSP_MEM_TYPE_IRAM_DSP_2	DSP2 内部 IRAM 地址空间。
SVP_DSP_MEM_TYPE_DRAM_0_DSP_2	DSP2 内部 DRAM0 地址空间。
SVP_DSP_MEM_TYPE_DRAM_1_DSP_2	DSP2 内部 DRAM1 地址空间。
SVP_DSP_MEM_TYPE_SYS_DDR_DSP_3	DSP3 使用的系统 DDR 内存地址空间。
SVP_DSP_MEM_TYPE_IRAM_DSP_3	DSP3 内部 IRAM 地址空间。
SVP_DSP_MEM_TYPE_DRAM_0_DSP_3	DSP3 内部 DRAM0 地址空间。
SVP_DSP_MEM_TYPE_DRAM_1_DSP_3	DSP3 内部 DRAM1 地址空间。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_DSP_CMD_E

【说明】

定义命令。

【定义】

```
typedef enum hiSVP_DSP_CMD_E
{
    SVP_DSP_CMD_INIT        = 0x0,
    SVP_DSP_CMD_EXIT        = 0x1,
```



```
SVP_DSP_CMD_ERODE_3X3 = 0x2,  
SVP_DSP_CMD_DILATE_3X3 = 0x3,  
SVP_DSP_CMD_BUTT  
}SVP_DSP_CMD_E;
```

【成员】

成员名称	描述
SVP_DSP_CMD_INIT	初始化，系统命令，用户无需关心。
SVP_DSP_CMD_EXIT	去初始化，系统命令，用户无需关心。
SVP_DSP_CMD_ERODE_3X3	Erode 3x3 命令。
SVP_DSP_CMD_DILATE_3X3	Dilate 3x3 命令。

【注意事项】

用户增加的命令必须 SVP_DSP_CMD_BUTT + 1 以后，不能与上海海思的重合。

【相关数据类型及接口】

无。

SVP_DSP_MESSAGE_S

【说明】

定义消息格式。

【定义】

```
typedef struct hiSVP_DSP_MESSAGE_S  
{  
    HI_U32      u32CMD;      /**<CMD ID, user-defined*/  
    HI_U32      u32MsgId;    /**<Message ID*/  
    HI_U64      u64Body;     /**<Message body*/  
    HI_U32      u32BodyLen;  /**<Length of pBody*/  
} SVP_DSP_MESSAGE_S;
```

【成员】



成员名称	描述
u32CMD	消息命令。
u32MsgId	消息 ID。
u64Body	消息体。
u32BodyLen	消息体大小。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_DSP_CONFIG_ATTR_S

【说明】

定义配置属性。

【定义】

```
typedef struct hiSVP_DSP_CONFIG_ATTR_S {  
    HI_U64 u64SramFirstAddr; /* address value defined in memmap.xmm */  
    HI_BOOL bldmaOffsetUseMmz;  
} SVP_DSP_CONFIG_ATTR_S;
```

【成员】

成员名称	描述
u64SramFirstAddr	DSP 的 SRAM 首地址。
bldmaOffsetUseMmz	DSP IDMA 偏移是否使用 MMZ 区域首地址计算。

【注意事项】

无。

【相关数据类型及接口】



无。

1.5 错误码

DSP 模块 API 错误码如表 1-1 所示。

表1-1 DSP 模块 API 错误码

错误代码	宏定义	描述
0xA0348001	HI_ERR_SVP_DSP_INVALID_DEVID	设备 ID 超出合法范围
0xA0348002	HI_ERR_SVP_DSP_INVALID_CHNID	通道组号错误或无效区域句柄
0xA0348003	HI_ERR_SVP_DSP_ILLEGAL_PARAM	参数超出合法范围
00xA0348004	HI_ERR_SVP_DSP_EXIST	重复创建已存在的设备、通道或资源
0xA0348005	HI_ERR_SVP_DSP_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
0xA0348006	HI_ERR_SVP_DSP_NULL_PTR	函数参数中有空指针
0xA0348007	HI_ERR_SVP_DSP_NOT_CONFIG	模块没有配置
0xA0348008	HI_ERR_SVP_DSP_NOT_SUPPORT	不支持的参数或者功能
0xA0348009	HI_ERR_SVP_DSP_NOT_PERM	该操作不允许，如试图修改静态配置参数
0xA034800C	HI_ERR_SVP_DSP_NOMEM	分配内存失败，如系统内存不足
0xA034800D	HI_ERR_SVP_DSP_NOBUF	分配缓存失败，如申请的图像缓冲区太大
0xA034800E	HI_ERR_SVP_DSP_BUF_EMPTY	缓冲区中无图像
0xA034800F	HI_ERR_SVP_DSP_BUF_FULL	缓冲区中图像满
0xA0348010	HI_ERR_SVP_DSP_NOTREADY	系统没有初始化或没有加载相应模块



错误代码	宏定义	描述
0xA0348011	HI_ERR_SVP_DSP_BADADDR	地址非法
0xA0348012	HI_ERR_SVP_DSP_BUSY	系统忙
0xA0348040	HI_ERR_SVP_DSP_SYS_TIMEOUT	系统超时
0xA0348041	HI_ERR_SVP_DSP_QUERY_TIMEOUT	Query 查询超时
0xA0348042	HI_ERR_SVP_DSP_OPEN_FILE	打开文件失败
0xA0348043	HI_ERR_SVP_DSP_READ_FILE	读文件失败

1.6 Proc 调试信息

1.6.1 概述

调试信息采用了 Linux 下的 proc 文件系统，可实时反映当前系统的运行状态，所记录的信息可供问题定位及分析时使用。

【文件目录】

/proc/umap

【信息查看方法】

- 在控制台上可以使用 cat 命令查看信息，cat /proc/umap/dsp；也可以使用其他常用的文件操作命令，例如 cp /proc/umap/dsp ./，将文件拷贝到当前目录。
- 在应用程序中可以将上述文件当作普通只读文件进行读操作，例如 fopen、fread 等。



说明

参数在描述时有以下 2 种情况需要注意：



- 取值为{0, 1}的参数，如未列出具体的取值和含义的对应关系，则参数为 1 时表示肯定，为 0 时表示否定。
- 取值为{aaa, bbb, ccc}的参数，未列出具体的取值和含义的对应关系，但可直接根据取值 aaa、bbb 或 ccc 判断参数含义

1.6.2 Proc 信息说明

【调试信息】

```
# cat /proc/umap/dsp
```

```
[DSP] Version: [Hi3559AV100_MPP_V1.0.0.0 B010 Release], Build Time[Apr 8 2019, 09:27:28]
```

-----MODULE PARAM-----

max_node_num	dsp_init_mode
30	0

-----DSP CORE STATUS-----

CoreId	Enable
0	Yes
1	No

-----DSP PRI STATUS-----

CoreId	Pri	Create
0	0	Yes
0	1	No
0	2	No
1	0	No
1	1	No
1	2	No

-----DSP TASK INFO-----

CoreId	Pri	Hnd	TaskFsh	HndWrap	FreeNum	WorkNum
0	0	3	3	0	29	1
0	1	0	0	0	30	0
0	2	0	0	0	30	0
1	0	0	0	0	30	0
1	1	0	0	0	30	0
1	2	0	0	0	30	0



-----DSP RUN-TIME INFO-----

CoreId	Pri	CntPerSec	MaxCntPerSec	TotalIntCntLastSec	TotalIntCnt	QTCnt
0	0	2	2	3	3	0
0	1	0	0	0	0	0
0	2	0	0	0	0	0
1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	2	0	0	0	0	0

CostTm	MCostTm	CostTmPerSec	MCostTmPerSec	TotalIntCostTm	RunTm
1	2	3	3	5	3
0	0	0	0	0	3
0	0	0	0	0	3
0	0	0	0	0	3
0	0	0	0	0	3
0	0	0	0	0	3

-----DSP INVOKE INFO-----

CoreId	Pri	RPC
0	0	3
0	1	0
0	2	0
1	0	0
1	1	0
1	2	0

【调试信息分析】

记录当前 DSP 工作状态资源信息，主要包括 DSP 队列状态信息，任务状态信息，运行时状态信息和调用信息。

【参数说明】

参数		描述
MODULE PARAM 模块 参数	max_node_num	最大节点数，也即可保存处理结果最大数量。
	dsp_init_mode	初始化模式，只有 0 或者 1。



参数		描述
DSP TASK INFO 任务信息	CoreId	DSP 核 ID。
	Pri	任务优先级。
	Hnd	当前可分配的任务 handle 号。
	TaskFsh	当前已完成任务 handle 号。
	HndWrap	用户 handle 号分配发生回写的次数。
	FreeNum	空闲节点数。
	WorkNum	已使用任务数。
DSP RUN- TIME INFO 运 行时相关信息	CoreId	DSP 核 ID。
	Pri	任务优先级。
	CntPerSec	最近一次的 1 秒内中断执行次数。
	MaxCntPerSec	历史上的 1 秒内最大的中断执行次数。
	TotalIntCntLastSec	上一秒上报中断总次数。
	TotalIntCnt	DSP 产生中断的总次数。
	QTCnt	DSP 查询超时中断次数。
	CostTm	最近一次执行中断的执行耗时。 单位：us。
	MCostTm	执行一次中断的最大耗时。 单位：us。
	CostTmPerSec	最近一秒执行中断的执行耗时。 单位：us。
	MCostTmPerSec	历史上一秒执行中断的最大执行耗时。 单位：us。
	TotalIntCostTm	中断处理总时间。 单位：us。



参数		描述
	RunTm	DSP 运行总时间。 单位：s。
DSP INVOKE INFO 调用信息	CoreId	DSP 核 ID。
	Pri	任务优先级。
	RPC	RPC 调用次数

【注意】

无



2 NNIE

2.1 概述

NNIE (Neuron Network Inference Engine) 是上海海思媒体处理芯片智能分析系统中的神经网络推断引擎。用户基于 NNIE 开发智能分析方案，降低 CPU 占用。



说明

Hi3556AV100 不支持 NNIE。对于 Hi3559V200，当 NNIE 部署在 linux 侧，GDC 部署在 Huawei LiteOS 侧时，如果加载了 GDC 模块，NNIE 模块不能使用。

2.2 功能描述

2.2.1 重要概念

- 网络分段

对于 NNIE 不支持的某些网络层节点，编译器支持用户对网络分段，不执行的部分编译器不去编译，由用户自己用 CPU 去实现。强烈建议用户尽量使用 NNIE 支持的层去实现网络模型；NNIE 不支持的段数越多，网络切分越碎，软硬件交互越频繁，效率越低。

- 句柄(handle)

用户在调用 NNIE API 创建任务时，系统会为每个任务分配一个 handle，用于标识不同的任务。

- 及时返回结果标志 bInstant

用户在创建某个任务后，希望及时得到该任务完成的信息，则需要在创建该任务时，将 bInstant 设置为 HI_TRUE。否则，如果用户不关心该任务是否完成，建议



将 bInstant 设置为 HI_FALSE，这样可以与后续任务组链执行，减少中断次数，提升性能。

- 查询(query)

用户根据系统返回的 handle，调用 [HI_MPI_SVP_NNIE_Query](#) 可以查询对应算子任务是否完成。

- 及时刷 cache

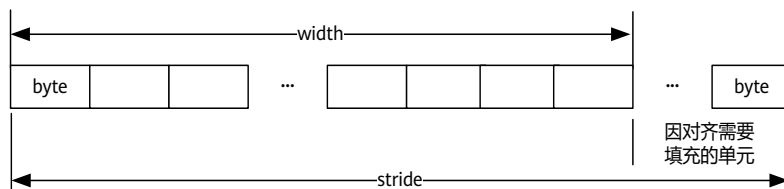
NNIE 硬件只能从 DDR 中获取数据。如果用户在调用 NNIE 任务时，访问空间可 cache 而且 CPU 曾经访问，为了保证 NNIE 输入输出数据不被 CPU cache 干扰，此时用户需要调用 HI_MPI_SYS_MmzFlushCache 接口刷 cache（详细信息请参见《HiMPP 媒体处理软件 Vx.y 开发参考》），将数据从 cache 刷到 DDR，以供 IVE 使用。

- 跨度 (stride)

一行的有效数据 byte 数目 + 为硬件快速跨越到下一行补齐的一些无效 byte 数目，如图 2-1 所示。注意不同的数据结构行存储表示的有效元素数目的变量不一样，且其度量跟 stride 不一定是一样的。

- SVP_BLOB_S 行存储方向表示的有效元素数目变量是 width。
- SVP_SEQ_S 行存储方向表示的有效元素数目变量是 dim。

图2-1 跨度 (stride) 示意图



- 对齐

硬件为了快速访问内存首地址或者跨行访问数据，要求内存地址或内存跨度必须为对齐系数的倍数。

- 数据内存首地址对齐
- 跨度对齐



须知

- Hi3559AV100/Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3559V200 在使用 DDR4 时，为提高访存效率，建议首地址使用 256 字节对齐，stride 使用 256 字节对齐。
- 区别于 IVE 模块中的 stride：NNIE 的 stride 是以 byte 为 stride 的计量单位；而 IVE 中的 stride 是与 width 具有相同的计量单位，是以“像素”为计量单位的。
- 输入、输出数据类型（具体结构定义请参见“1.4 数据类型和数据结构”）
 - 块数据类型
[SVP_BLOB_S](#)、[SVP_SRC_BLOB_S](#)、[SVP_DST_BLOB_S](#)，类型参考 [SVP_BLOB_TYPE_E](#)，具体的内存分配如图 2-2 ~ 图 2-7 所示。
 - 一维数据
[SVP_MEM_INFO_S](#)、[SVP_SRC_MEM_INFO_S](#)、[SVP_DST_MEM_INFO_S](#)，表示一维数据，如图 2-8。
- BLOB 内存排布类型

表2-1 BLOB 内存排布类型表

类型	BLOB 描述
SVP_BLOB_TYPE_S32	多帧有符号 32bit 多通道数据 Planar 格式存储顺序排布。此时 SVP_BLOB_S 结构体中，u32Num 表示帧数，u32Width 表示图像宽，u32Height 表示图像高，u32Chn 表示单帧图像通道数，如图 2-2 所示。
SVP_BLOB_TYPE_U8	多帧无符号 8bit 多通道数据 Planar 格式存储顺序排布。此时 SVP_BLOB_S 结构体中，u32Num 表示帧数，u32Width 表示图像宽，u32Height 表示图像高，u32Chn 表示单帧图像通道数，如图 2-3 所示。
SVP_BLOB_TYPE_YVU420SP	多帧 YCbCr420 SemiPlannar 数据格式图像顺序排布。此时 SVP_BLOB_S 结构体中，u32Num 表示帧数，u32Width 表示图像宽，u32Height 表示图像高，u32Chn=3，如图 2-4 所示。色度部分 V 在前，U 在后。



类型	BLOB 描述
	注意此时高、宽必须为偶数。
SVP_BLOB_TYPE_YVU422SP	多帧 YCbCr422 SemiPlannar 数据格式图像顺序排布。此时 SVP_BLOB_S 结构体中，u32Num 表示帧数，u32Width 表示图像宽，u32Height 表示图像高，u32Chn=3，如图 2-5 所示。色度部分 V 在前，U 在后。注意此时宽必须为偶数。
SVP_BLOB_TYPE_VEC_S32	多帧有符号 32bit 向量数据顺序排布。此时 SVP_BLOB_S 结构体中，u32Num 表示帧数，u32Width 表示向量数据维度，u32Height 表示单帧有多少个向量（一般 u32Height=1），u32Chn=1，如图 2-6 所示。
SVP_BLOB_TYPE_SEQ_S32	多段有符号 32bit 序列数据排布。此时 SVP_BLOB_S 结构体中，u32Num 表示段数，u32Dim 表示序列向量数据维度，u64VirAddrStep 是一个 u32Num 长度的数组地址，数组元素表示每段序列有多少个向量，如图 2-7 所示。



图2-2 SVP_BLOB_TYPE_S32 类型 SVP_BLOB_S (2 通道 2 帧示意图)

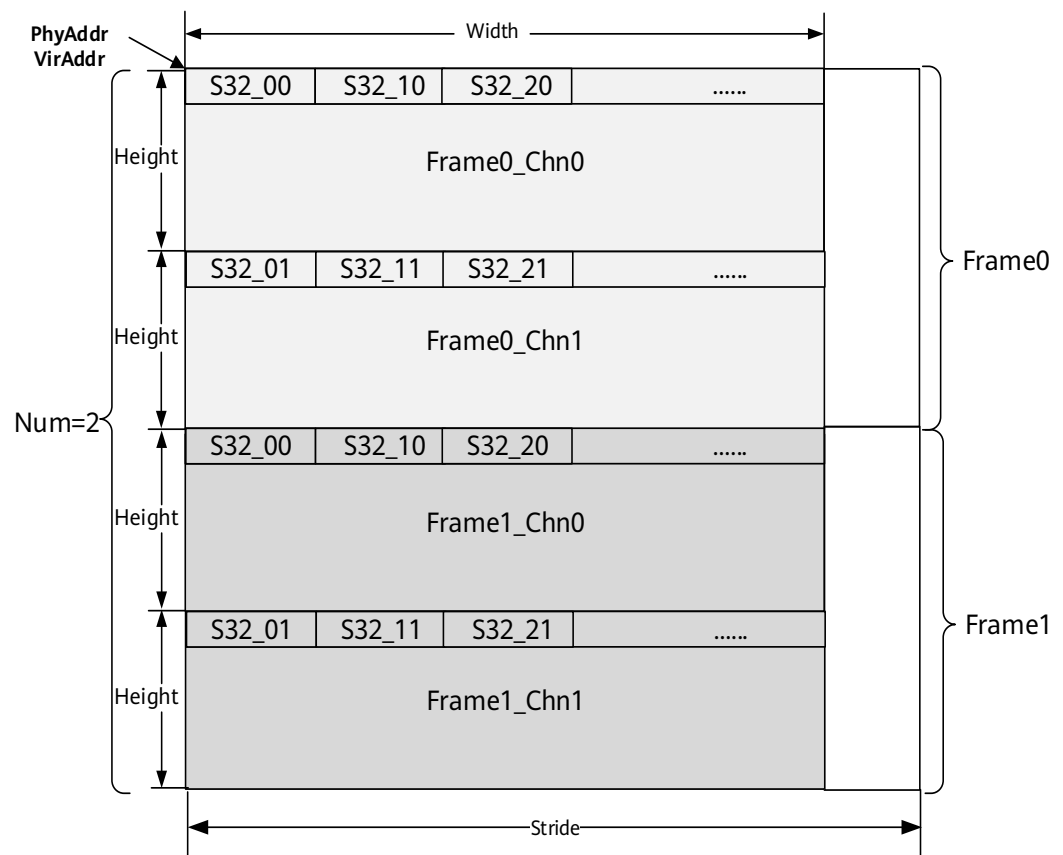
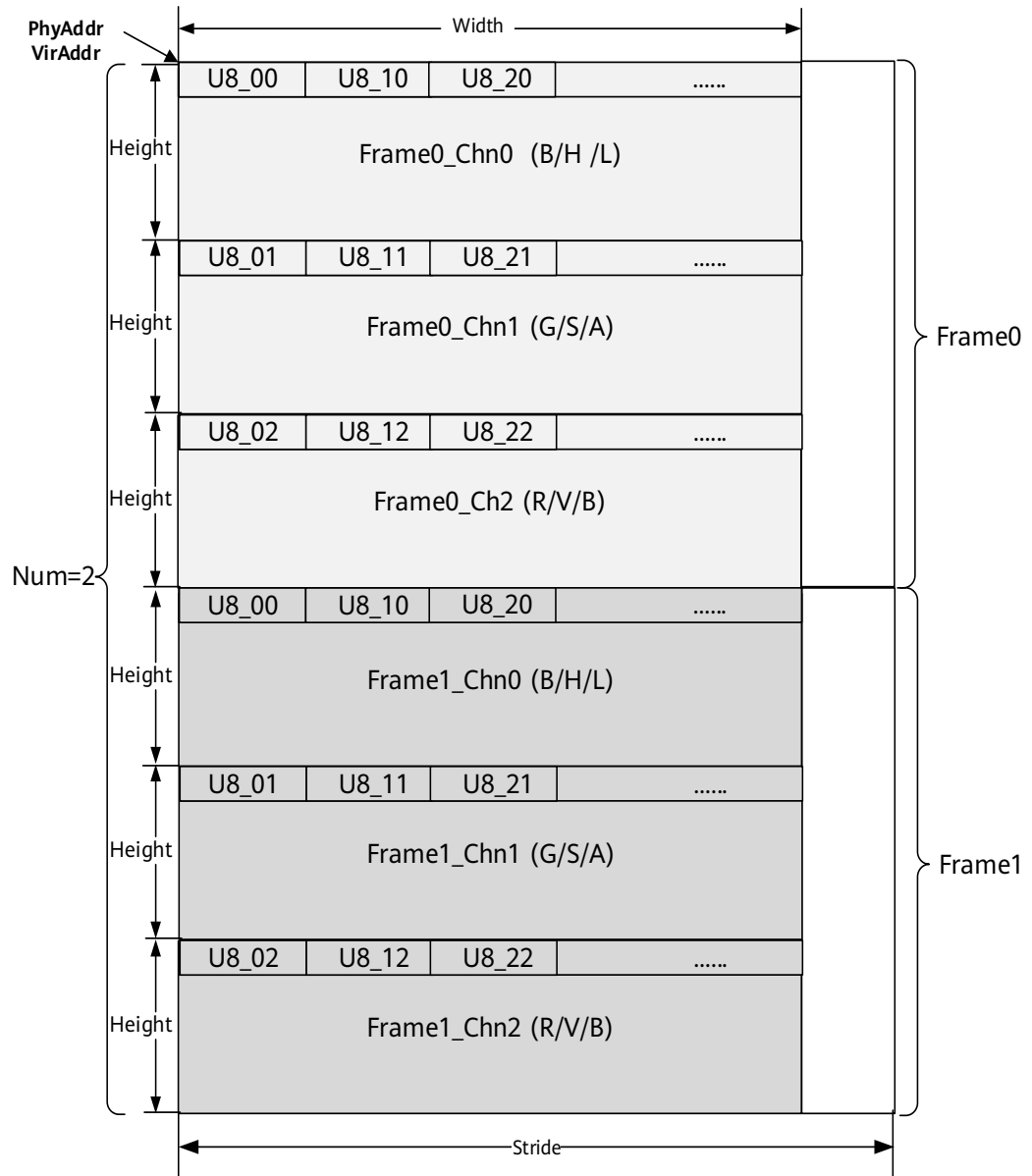


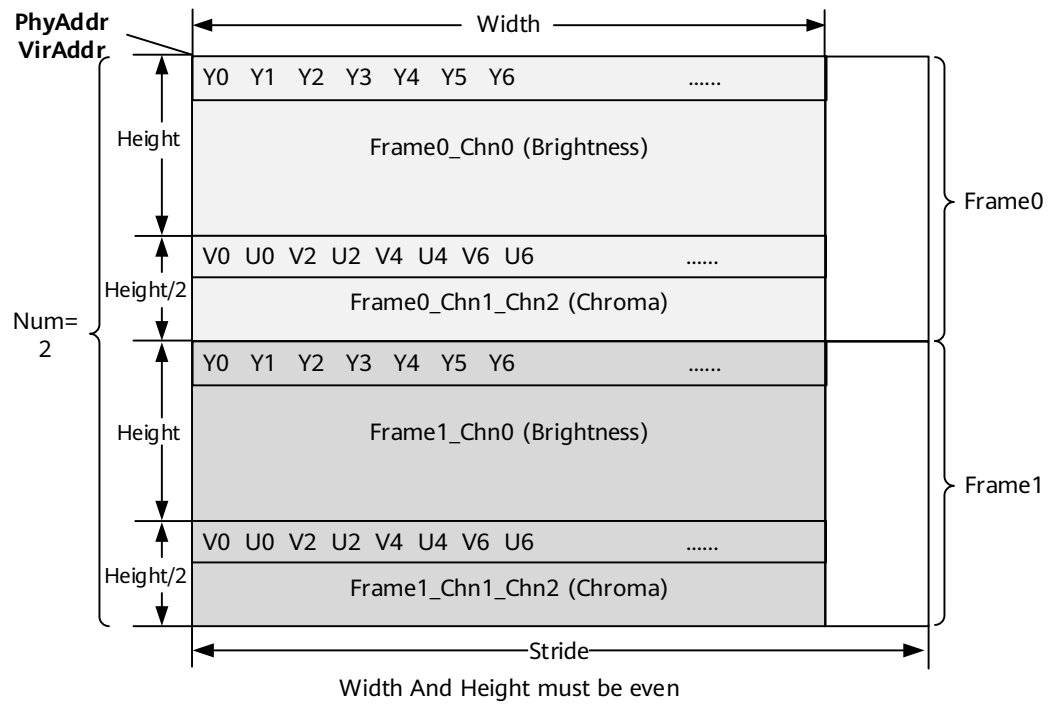
图2-3 SVP_BLOB_TYPE_U8 类型 SVP_BLOB_S (3 通道 2 帧示意图)



注：典型的 RGB\HSV\LAB 图像 Planar 格式存储，NNIE 默认以图示的 B\G\R、H\S\V、L\A\B 顺序存储。

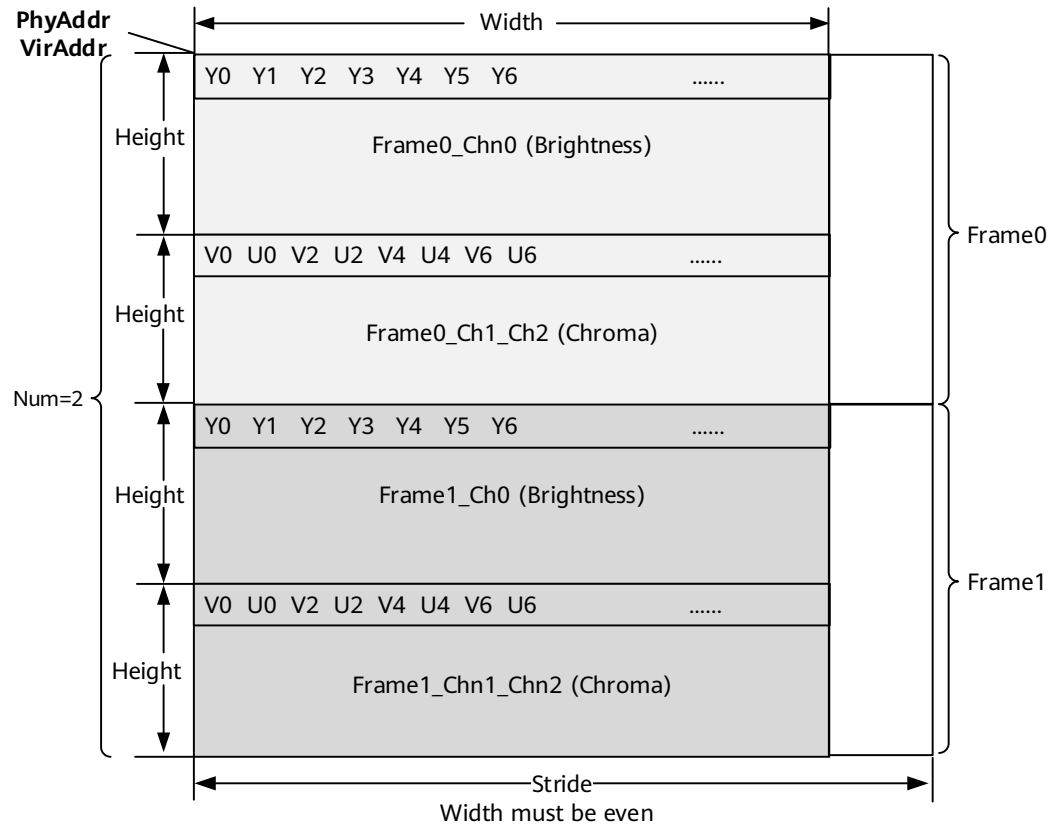


图2-4 SVP_BLOB_TYPE_YVU420SP 类型 SVP_BLOB_S (2 帧 YVU420SP 示意图)



注：这里 V 在前，U 在后。

图2-5 SVP_BLOB_TYPE_YVU422SP 类型 SVP_BLOB_S (2 帧 YVU422SP 示意图)



注：这里 V 在前，U 在后。

图2-6 SVP_BLOB_TYPE_VEC_S32 类型 SVP_BLOB_S (2 帧示意图)

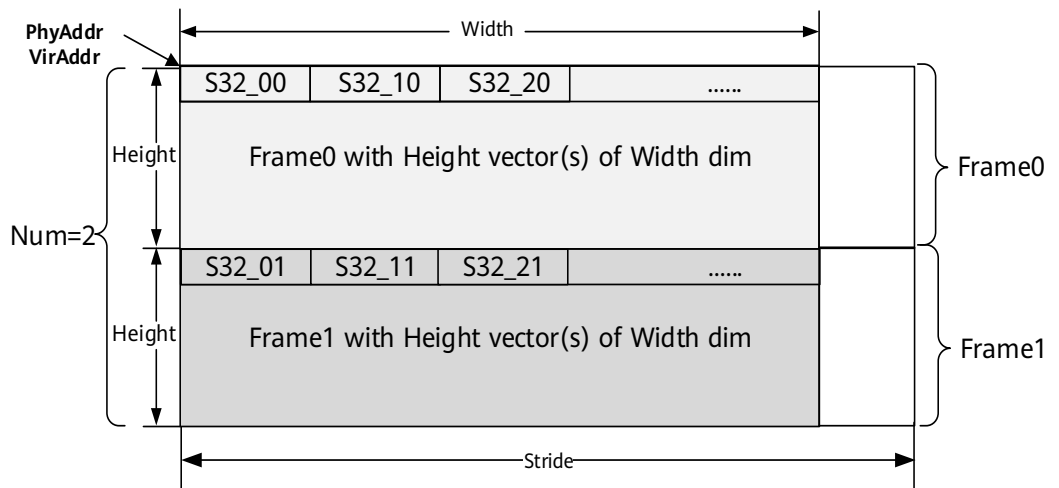


图2-7 SVP_BLOB_TYPE_SEQ_S32 类型 SVP_BLOB_S(Num=N 帧示意图)

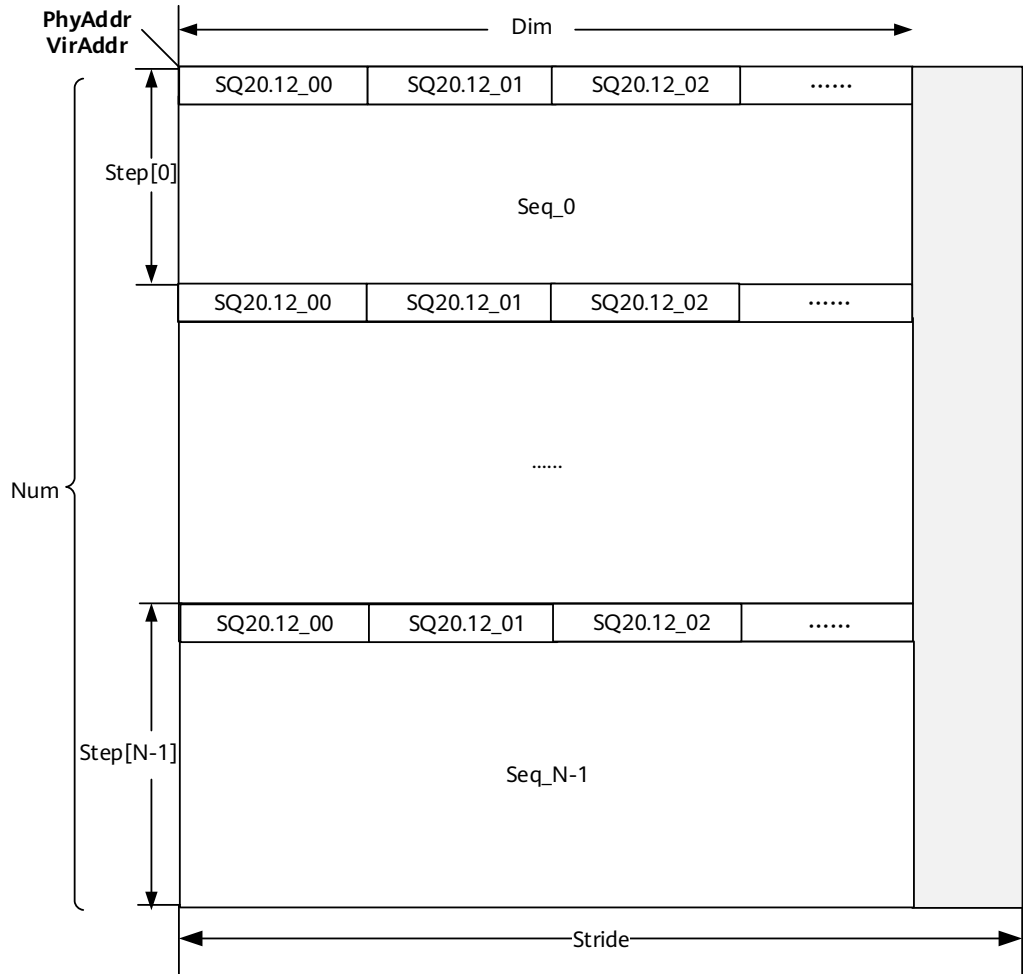
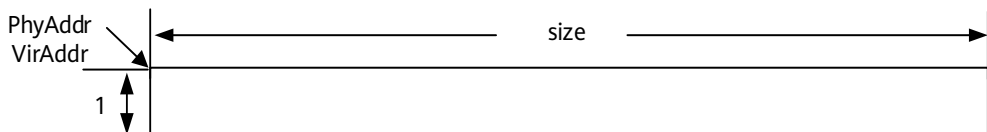


图2-8 SVP_MEM_INFO_S 类型的数据内存示意



2.2.2 使用示意

- 用户根据需求调用相应的算子接口创建任务，指定 blnstant 类型，并记录该任务返回的 handle 号。
- 根据返回的 handle 号，指定阻塞方式，可以查询到该任务的完成状态。



具体可参见 [HI_MPI_SVP_NNIE_Query](#) 中的【举例】。

2.3 API 参考

NNIE 模块提供了创建任务和查询任务的基本接口。

- [HI_MPI_SVP_NNIE_LoadModel](#): 从用户事先加载到 buf 中的模型中解析出网络模型。
- [HI_MPI_SVP_NNIE_GetTskBufSize](#): 获取给定网络任务各段辅助内存大小。
- [HI_MPI_SVP_NNIE_Forward](#): 多节点输入输出的 CNN 类型网络预测。
- [HI_MPI_SVP_NNIE_ForwardWithBbox](#): 多个节点 feature map 输入。
- [HI_MPI_SVP_NNIE_UnloadModel](#): 卸载模型。
- [HI_MPI_SVP_NNIE_Query](#): 查询任务是否完成。
- [HI_MPI_SVP_NNIE_Query_Timeout](#): 查询任务是否完成,超时时间由上层用户控制。
- [HI_MPI_SVP_NNIE_AddTskBuf](#): 记录 TskBuf 地址信息。
- [HI_MPI_SVP_NNIE_RemoveTskBuf](#): 移除 TskBuf 地址信息。

HI_MPI_SVP_NNIE_LoadModel

【描述】

从用户事先加载到 buf 中的模型中解析出网络模型。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_LoadModel (const SVP\_SRC\_MEM\_INFO\_S *pstModelBuf,  
SVP\_NNIE\_MODEL\_S *pstModel);
```

【参数】

参数名称	描述	输入/输出
pstModelBuf	存储模型的 buf, 用户需事先开辟好, 且将 NNIE 编译器得到的 wk 文件加载到该 buf 中。 不能为空。	输入
pstModel	网络模型结构体。	输出



【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib）

【注意】

- 用户需要保证 pstModelBuf 中的地址所指向的内存中存储的模型数据的完整性和正确性。
- 用户需要保证 pstModelBuf 中的地址所指向的内存只有当所存储的模型不再使用后才能被释放，并且在释放之前内存中的数据不能被修改。
- 用户需要保证解析获得的 pstModel 里的内容不能被修改。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_NNIE_GetTskBufSize

【描述】

获取给定网络任务各段辅助内存大小。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_GetTskBufSize(HI_U32 u32MaxBatchNum, HI_U32 u32MaxBboxNum,  
const SVP\_NNIE\_MODEL\_S *pstModel, HI_U32 au32TskBufSize[], HI_U32 u32NetSegNum);
```

【参数】



参数名称	描述	输入/输出
u32MaxBatchNum	输入到当前网络最大图像帧数。要求 HI_MPI_SVP_NNIE_Forward 和 HI_MPI_SVP_NNIE_ForwardWithBbox 输入的 astSrc[] 中的 u32Num 小于等于该值。 取值范围：[1, 256]	输入
u32MaxBboxNum	网络中有 RPN 层时输出的最大候选 Bounding box 个数。网络有 RPN 层时，nnie_mapper 中有配置 max_roi_frame_cnt 值，u32MaxBboxNum 要求不大于 max_roi_frame_cnt，建议配置相等。	输入
pstModel	网络模型结构体。	输入
au32TaskBufSize[]	网络任务各段辅助内存。	输出
u32NetSegNum	网络任务的段数，需跟 pstModel 中的段数匹配。 取值范围：[1, 8]	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib）

【注意】

- 针对单线程运行一个网络模型，用户开辟 tskBuf 可以根据网络段的运行关系来选择以下两种方案：



- NNIE→非 NNIE→NNIE→非 NNIE，类似这种 NNIE、非 NNIE（CPU 或者 DSP 等）间隔的网络，用户可以选择开辟一个分段 au32TskBufSize[]中的最大值，每个段可以复用这段内存；
- NNIE→NNIE→非 NNIE→NNIE→非 NNIE，类似这种存在 N 个 NNIE 连续顺序执行段的网络，连续的 NNIE 段不能复用 tskBuf，按照最省内存原则可以选择开辟满足这 N 个连续 NNIE 段的其中 N-1 个 size 和最小的 tskBuf 以及剩余所有段中最大的一片 tskBuf，具体按文中示例，可以选择开辟 “NNIE→NNIE” 中较小 size 的 tskBuf，后面 “非 NNIE→NNIE→非 NNIE” 中可以复用最大 size 这片 taskBuf；
- 多线程运行同一个网络模型，每个线程需要各自独立的 tskBuf，开辟的方式可以参考“针对单线程运行一个网络模型”的情况。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_NNIE_Forward

【描述】

多节点输入输出的 CNN 类型网络预测。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_Forward(SVP_NNIE_HANDLE *phSvpNnieHandle, const  
SVP_SRC_BLOB_S astSrc[],const SVP_NNIE_MODEL_S *pstModel, const SVP_DST_BLOB_S  
astDst[],const SVP_NNIE_FORWARD_CTRL_S *pstForwardCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
phSvpNnieHandle	handle 指针。 不能为空。	输出
astSrc[]	多个节点输入，节点的顺序跟网络描述中的顺序要求一致，支持多帧同时输入。	输入
pstModel	网络模型结构体。	输入



参数名称	描述	输入/输出
astDst[]	网络段的多个节点输出，包含用户标记需要上报输出的中间层结果，以及网络段的最终结果。	输出
pstForwardCtrl	控制结构体。	输入
blInstant	及时返回结果标志。	输入

参数名称	支持类型	地址对齐	分辨率
astSrc[]	YVU420SP/ YVU422SP/ U8/S32/ VEC_S32/ SEQ_S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	分辨率要求： 必须与 pstModel-> astSeg[pstForwardCtrl-> u32NetSegId]中 astSrcNode[]的维度信息 一致。
pstModel	网络段类型支持 CNN/Current	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	无
astDst[]	S32/VEC_S32/SEQ_S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	分辨率要求： 必须与 pstModel-> astSeg[pstForwardCtrl-> u32NetSegId]中 astDstNode[]的维度信息 一致。

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

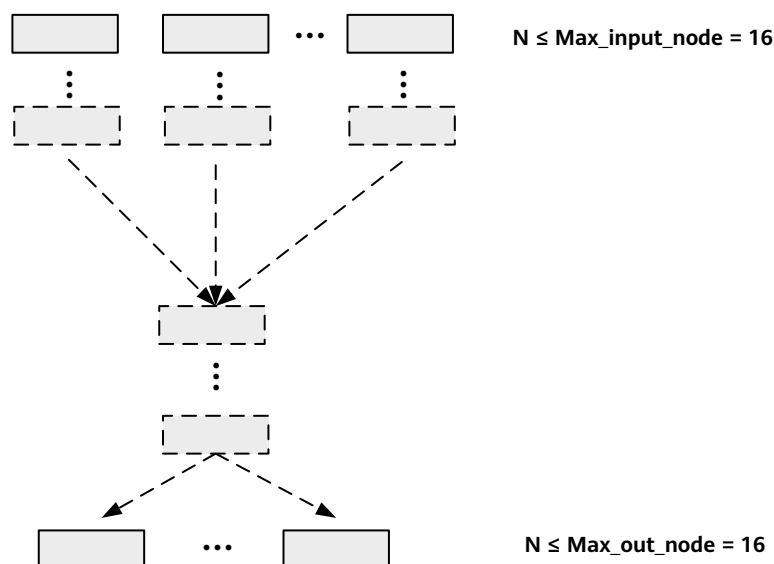
【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a (PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib)

【注意】

- 用户需要保证 pstModel-> stBase 中的地址所指向的内存中数据的完整性和正确性。
- 用户需要保证 pstModel 结构体中的内容与 pstModel-> stBase 中的地址所指向的内存中的数据是同一个模型文件解析获得的。
- 网络段类型为 SVP_NNIE_NET_TYPE_RECURRENT 类型时，用户需要保证类型为 SVP_BLOB_TYPE_SEQ_S32 的输入输出 blob 中虚拟地址 virt_addr_step 及其指向内存大小的正确性。
- U8 图像输入只支持 1 通道（灰度图）和 3 通道(RGB 图)；
- 多个 Blob 输入输出时，配合编译器输出的 dot 描述文件生成的 dot 图示，可以看到 Blob 跟层的对应关系。

图2-9 NNIE_Forward 支持的多节点输入输出网络示意图



【举例】



无。

【相关主题】

无。

HI_MPI_SVP_NNIE_ForwardWithBbox

【描述】

多个节点 feature map 输入，astBbox[]是经过 RPN 层得到的该网络段若干 ROI Pooling\ PSROI Pooling 层的 Bounding Box 输入，网络对输入的 astBbox[]位置进行评分以及重新调整。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_ForwardWithBbox(SVP_NNIE_HANDLE *phSvpNnieHandle, const  
SVP_SRC_BLOB_S astSrc[],const SVP_SRC_BLOB_S astBbox[],const SVP_NNIE_MODEL_S  
*pstModel, const SVP_DST_BLOB_S astDst[],const SVP_NNIE_FORWARD_WITHBBOX_CTRL_S  
*pstForwardCtrl, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
phSvpNnieHandle	handle 指针。 不能为空。	输出
astSrc[]	网络多节点输入，节点的顺序跟网络描述中的顺序要求一致，每个节点只支持单帧输入。	输入
astBbox[]	网络段的 Bounding Box 输入，Blob 中的 Height 表示 Bbox 的个数，Width=4，参考图 2-3。	输入
pstModel	网络模型结构体。	输入
astDst[]	网络段的多个节点输出，包含 Score、Bbox 调整值、中间层输出。 表示 Score 的 Blob 中，Width 跟 pstModel 中的类别数一致，Height 跟 astBbox 中的 Height 一致，参考图 2-4； 表示 Bbox 调整值的 Blob 中，Height 跟	输出



参数名称	描述	输入/输出
	astBbox 中的 Height 一致, 参考图 2-5、图 2-6 和注意中的说明。 若是有其它中间层上报的情况, 输出的 feature map 需要跟 pstModel 中记录的情况一致, 参考图 2-2。	
pstForwardCtrl	控制结构体。	输入
blInstant	及时返回结果标志。	输入

参数名称	支持类型	地址对齐	分辨率
astSrc[]	S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	分辨率要求: 必须与 pstModel->astSeg[pstForwardCtrl->u32NetSegId]中 astSrcNode[]的维度信息一致。
astBbox[]	S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	Width=4, Height≤5000, 并且 Height 不能大于调用 HI_MPI_SVP_NNIE_GetTskBufSize 计算 TskBufSize 时传入的 u32MaxBboxNum 的值。
pstModel	网络段类型只支持 ROI\PSROI	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	-



参数名称	支持类型	地址对齐	分辨率
astDst[]	VEC_S32/ S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	分辨率要求: 必须与 pstModel-> astSeg[pstForwardCtrl -> u32NetSegId]中 astDstNode[]的维度信 息一致。

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib）

【注意】

- 用户需要保证 pstModel-> stBase 中的地址所指向的内存中数据的完整性和正确性。
- 用户需要保证 pstModel 结构体中的内容与 pstModel-> stBase 中的地址所指向的内存中的数据是同一个模型文件解析获得的。
- 当前 astBbox[]数组的元素个数仅支持 1，即 pstForwardCtrl-> u32ProposalNum=1，参考[图 2-10](#)；
- astBbox 中的坐标都采用 SQ20.12 的定点输入，参考[图 2-11](#)；
- 输出的 Score 示意图参考[图 2-12](#)；
- 根据训练时的设定，输出的 Bbox 坐标调整值 Bbox_Delta，大致可分为 3 种情况：
 - 每一类别单独享有自己的 Bbox_Delta，则对应每一个框 Bbox_Delta 的输出维度为 class_num * 4，参考[图 2-13](#)；

- 各类别共享一组 Bbox_Delta, 则对应每一个框 Bbox_Delta 的输出维度为 4, 参考图 2-14;
- 背景类有一组 Bbox_Delta, 前景类别共用一组 Bbox_Delta, 则对应每一个框 Bbox_Delta 的输出维度为 8, 参考图 2-15;

图2-10 NNIE_ForwardWithBbox 支持的输入输出网络示意图

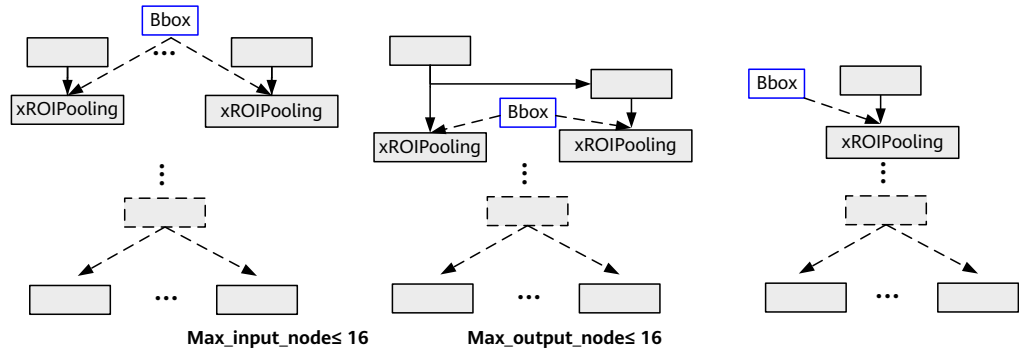


图2-11 NNIE_ForwardWithBbox astBbox[i]输入示意图

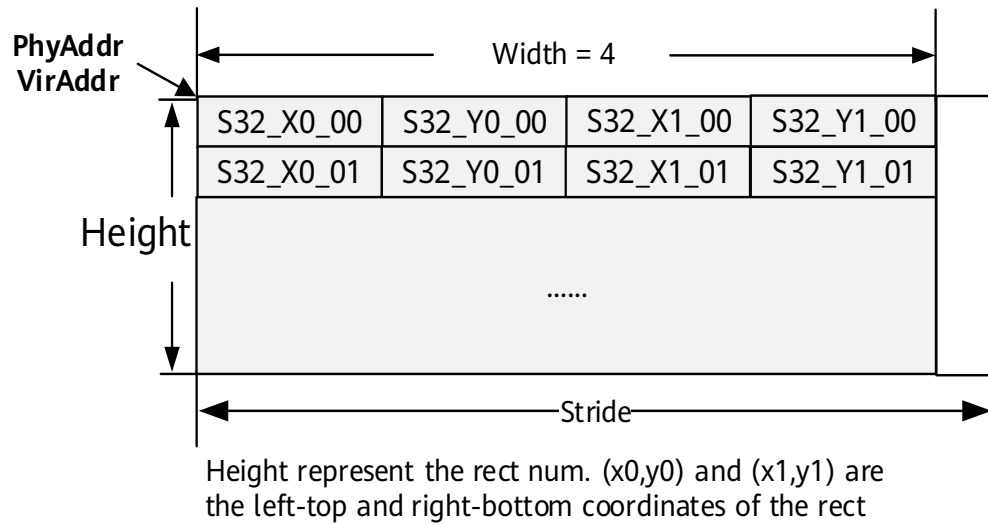


图2-12 NNIE_ForwardWithBbox Score 输出示意图

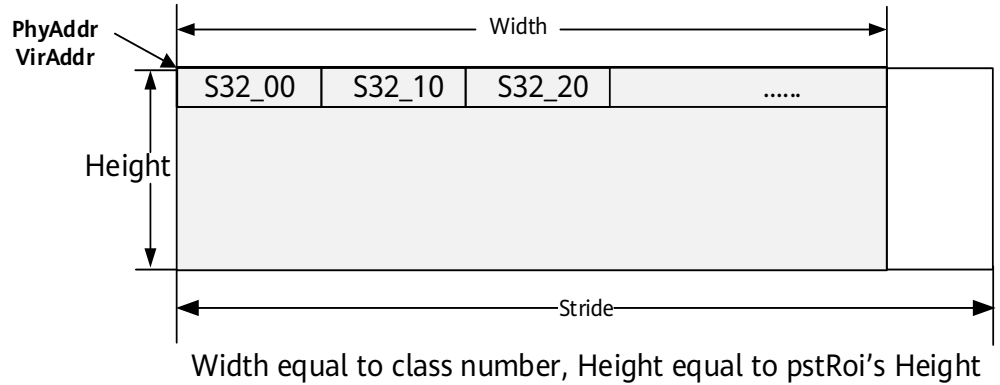


图2-13 NNIE_ForwardWithBbox Bbox 调整值输出示意图 1

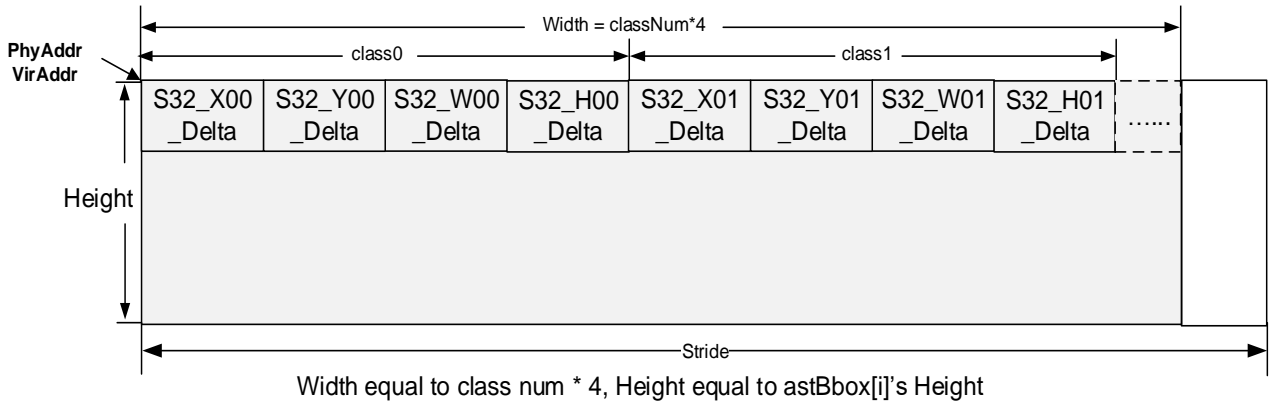


图2-14 NNIE_ForwardWithBbox Bbox 调整值输出示意图 2

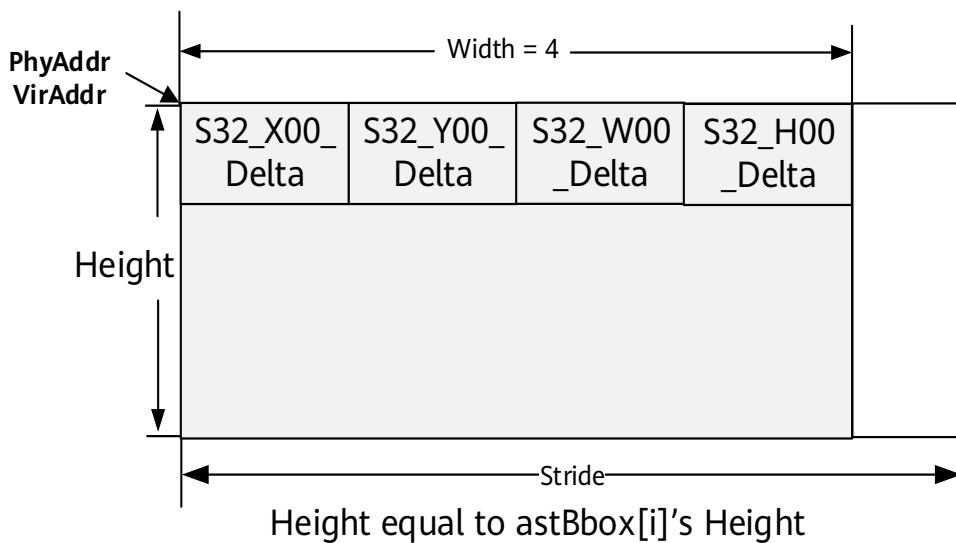
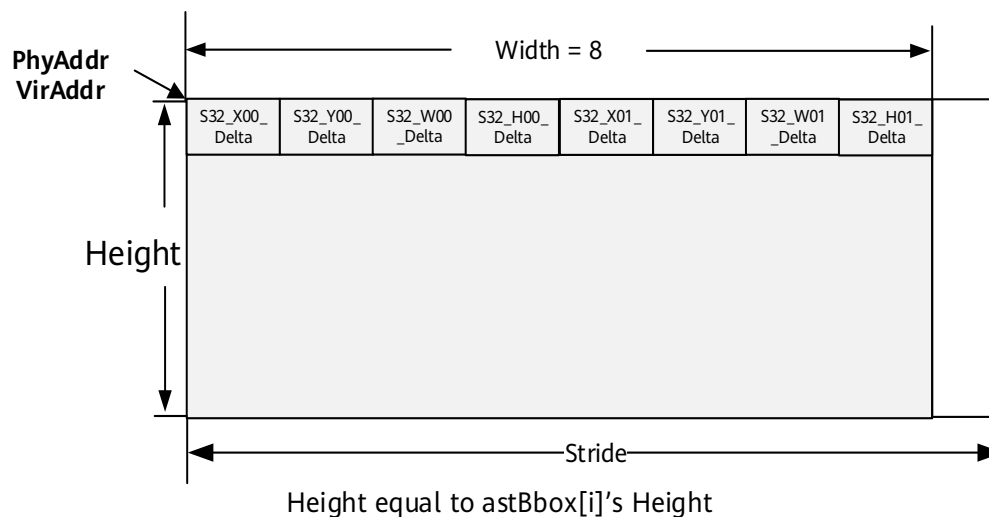




图2-15 NNIE_ForwardWithBbox Bbox 调整值输出示意图 3



【举例】

无。

【相关主题】

无。

HI_MPI_SVP_NNIE_UnloadModel

【描述】

卸载模型。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_UnloadModel(SVP\_NNIE\_MODEL\_S *pstModel);
```

【参数】

参数名称	描述	输入/输出
pstModel	网络模型结构体。	输入

【返回值】



返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a (PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib)

【注意】

无。

【举例】

无。

【相关主题】

无。

HI_MPI_SVP_NNIE_Query

【描述】

查询任务是否完成。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_Query(SVP\_NNIE\_ID\_E enNnieId, SVP\_NNIE\_HANDLE svpNnieHandle,  
HI_BOOL *pbFinish, HI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
enNnieId	任务所运行的 NNIE 核指示标志	输入
svpNnieHandle	handle。	输入
pbFinish	是否完成标志。	输出
bBlock	是否阻塞查询。	输入



【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib）

【注意】

无。

【举例】

无。

【相关主题】

[HI_MPI_SVP_NNIE_Query_Timeout](#)

HI_MPI_SVP_NNIE_Query_Timeout

【描述】

查询任务是否完成，超时时间由上层用户控制。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_Query_Timeout(SVP\_NNIE\_ID\_E enNnield, SVP\_NNIE\_HANDLE
svpNnieHandle, HI_S32 s32MilliSec, HI_BOOL *pbFinish);
```

【参数】

参数名称	描述	输入/输出
enNnield	任务所运行的 NNIE 核指示标志	输入
svpNnieHandle	handle。	输入



参数名称	描述	输入/输出
s32MilliSec	超时时间，单位毫秒。	输入
pbFinish	是否完成标志。 不能为空。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib）

【注意】

当 s32MilliSec 设为-1 时，表示阻塞模式，程序一直等待，直到查询的任务完成或者系统出错才返回。

- 如果 s32MilliSec 等于 0 时，表示非阻塞模式。
- 如果 s32MilliSec 大于 0 时，表示超时等待模式，参数的单位是毫秒，指超时时间，在此时间内如果任务没有完成，则超时返回。

【举例】

无。

【相关主题】

[HI_MPI_SVP_NNIE_Query](#)

HI_MPI_SVP_NNIE_AddTskBuf

【描述】

记录 TskBuf 地址信息。



【语法】

```
HI_S32 HI_MPI_SVP_NNIE_AddTskBuf(const SVP_MEM_INFO_S* pstTskBuf);
```

【参数】

参数名称	描述	输入/输出
pstTskBuf	TskBuf 指针。 不能为空。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib）

【注意】

- 记录 TskBuf 地址信息，用于减少内核态内存映射次数，提升效率。
- TskBuf 地址信息的记录是通过链表进行管理，链表长度默认值为 32，链表长度可通过模块参数 nnie_max_tskbuf_num 进行配置。
- 若没有调用 [HI_MPI_SVP_NNIE_AddTskBuf](#) 预先把 TskBuf 地址信息记录到系统，那么之后调用 Forward/ForwardWithBbox 每次都会 Map/Unmap 操作 TskBuf 内核态虚拟地址，效率会比较低。
- 必须与 [HI_MPI_SVP_NNIE_RemoveTskBuf](#) 成对匹配使用。
- 建议先把 Forward/ForwardWithBbox 用到的 TskBuf 地址信息调用此接口记录到系统。当不再使用时调用 [HI_MPI_SVP_NNIE_RemoveTskBuf](#) 把 TskBuf 地址信息移除。只需要在初始化时把 TskBuf 地址信息记录，后续可以直接使用，直到不再使用时才移除。
- pstTskBuf ->u64VirAddr 不使用，不做参数异常检查。



- pstTskBuf ->u32Size 不能为 0。
- TskBuf 内存由用户释放，记录的 TskBuf 要在移除后才能被释放。

【举例】

无。

【相关主题】

- [HI_MPI_SVP_NNIE_GetTskBufSize](#)
- [HI_MPI_SVP_NNIE_Forward](#)
- [HI_MPI_SVP_NNIE_ForwardWithBbox](#)
- [HI_MPI_SVP_NNIE_RemoveTskBuf](#)

HI_MPI_SVP_NNIE_RemoveTskBuf

【描述】

移除 TskBuf 地址信息。

【语法】

```
HI_S32 HI_MPI_SVP_NNIE_RemoveTskBuf(const SVP\_MEM\_INFO\_S\* pstTskBuf);
```

【参数】

参数名称	描述	输入/输出
pstTskBuf	TskBuf 指针。 不能为空。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_comm_svp.h、hi_nnie.h、mpi_nnie.h



- 库文件: libnnie.a (PC 上模拟用 nniefc1.x.lib\nnieit1.x.lib)

【注意】

- 如果 TskBuf 不再使用, 需要将记录的 TskBuf 地址信息从链表中移除。
- 必须与 [HI_MPI_SVP_NNIE_AddTskBuf](#) 成对匹配使用。
- pstTskBuf ->u64VirAddr 不使用, 不做参数异常检查。
- pstTskBuf ->u32Size 不能为 0。
- TskBuf 内存由用户释放, 记录的 TskBuf 要在移除后才能被释放。

【举例】

无。

【相关主题】

- [HI_MPI_SVP_NNIE_GetTskBufSize](#)
- [HI_MPI_SVP_NNIE_Forward](#)
- [HI_MPI_SVP_NNIE_ForwardWithBbox](#)
- [HI_MPI_SVP_NNIE_AddTskBuf](#)

2.4 数据类型和数据结构

NNIE 相关数据类型、数据结构定义如下:

定点数据类型

【说明】

定义定点化的数据类型。

【定义】

```
typedef unsigned char    HI_U0Q8;
typedef unsigned char    HI_U1Q7;
typedef unsigned char    HI_U5Q3;
typedef unsigned short   HI_U0Q16;
typedef unsigned short   HI_U4Q12;
typedef unsigned short   HI_U6Q10;
typedef unsigned short   HI_U8Q8;
typedef unsigned short   HI_U14Q2;
```




```
typedef unsigned short    HI_U12Q4;
typedef short             HI_S14Q2;
typedef short             HI_S9Q7;
typedef unsigned int      HI_U22Q10;
typedef unsigned int      HI_U25Q7;
typedef int                HI_S25Q7;
typedef int                HI_S20Q12;
typedef unsigned short     HI_U8Q4F4;    /*8bits unsigned integer, 4bits decimal
fraction, 4bits flag bits*/
```

【成员】

成员名称	描述
HI_U0Q8	用 0bit 表示整数部分，8bit 表示小数部分。文档中用 UQ0.8 来表示。
HI_U1Q7	用高 1bit 无符号数据表示整数部分，低 7bit 表示小数部分。文档中用 UQ1.7 来表示。
HI_U5Q3	用高 5bit 无符号数据表示整数部分，低 3bit 表示小数部分。文档中用 UQ5.3 来表示。
HI_U0Q16	用 0bit 表示整数部分，16bit 表示小数部分。文档中用 UQ0.16 来表示。
HI_U4Q12	用高 4bit 无符号数据表示整数部分，低 12bit 表示小数部分。文档中用 UQ4.12 来表示。
HI_U6Q10	用高 6bit 无符号数据表示整数部分，低 10bit 表示小数部分。文档中用 UQ6.10 来表示。
HI_U8Q8	用高 8bit 无符号数据表示整数部分，低 8bit 表示小数部分。文档中用 UQ8.8 来表示。
HI_U14Q2	用高 14bit 无符号数据表示整数部分，低 2bit 表示小数部分。文档中用 UQ14.2 来表示。
HI_U12Q4	用高 12bit 无符号数据表示整数部分，低 4bit 表示小数部分。文档中用 UQ12.4 来表示。



成员名称	描述
HI_S14Q2	用高 14bit 有符号数据表示整数部分，低 2bit 表示小数部分。文档中用 SQ14.2 来表示。
HI_S9Q7	用高 9bit 有符号数据表示整数部分，低 7bit 表示小数部分。文档中用 SQ9.7 来表示。
HI_U22Q10	用高 22bit 无符号数据表示整数部分，低 10bit 表示小数部分。文档中用 UQ22.10 来表示。
HI_U25Q7	用高 25bit 无符号数据表示整数部分，低 7bit 表示小数部分。文档中用 UQ25.7 来表示。
HI_S25Q7	用高 25bit 有符号数据表示整数部分，低 7bit 表示小数部分。文档中用 SQ25.7 来表示。
HI_S20Q12	用高 20bit 有符号数据表示整数部分，低 12bit 表示小数部分。文档中用 SQ20.12 来表示。
HI_U8Q4F4	用高 8bit 无符号数据表示整数部分，中间 4bit 表示小数部分，低 4bit 表示标志位。文档中用 UQF8.4.4 来表示。

【注意事项】

HI_UxQyFz\HI_SxQy:

- U 后面的数字 x 表示是用 x bit 无符号数据表示整数部分；
- S 后面的数字 x 表示用 x bit 有符号数据表示整数部分；
- Q 后面的数字 y 表示用 y bit 数据表示小数部分；
- F 后面的数字 z 表示用 z bit 来表示标志位；
- 从左到右依次表示高 bit 位到低 bit 位。

【相关数据类型及接口】

无。

SVP_NNIE_HANDLE

【说明】

定义 NNIE 的句柄。



【定义】

```
typedef HI_S32 SVP_NNIE_HANDLE;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_MEM_INFO_S

【说明】

定义一维内存信息。

【定义】

```
typedef struct hiSVP_MEM_INFO_S
{
    HI_U64  u64PhyAddr; /* RW;The physical address of the memory */
    HI_U64  u64VirAddr; /* RW;The virtual address of the memory */
    HI_U32  u32Size;    /* RW;The size of memory */
}SVP_MEM_INFO_S;
```

【成员】

成员名称	描述
u64VirAddr	内存块虚拟地址。
u64PhyAddr	内存块物理地址。
u32Size	内存块字节数。见 图 2-8 。

【注意事项】

无。

【相关数据类型及接口】



- [SVP_SRC_MEM_INFO_S](#)
- [SVP_DST_MEM_INFO_S](#)

SVP_SRC_MEM_INFO_S

【说明】

定义源序列。

【定义】

```
typedef SVP\_MEM\_INFO\_S SVP_SRC_MEM_INFO_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

- [SVP_MEM_INFO_S](#)
- [SVP_DST_MEM_INFO_S](#)

SVP_DST_MEM_INFO_S

【说明】

定义输出序列。

【定义】

```
typedef SVP\_MEM\_INFO\_S SVP_DST_MEM_INFO_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

- [SVP_MEM_INFO_S](#)
- [SVP_SRC_MEM_INFO_S](#)



SVP_BLOB_TYPE_E

【说明】

定义 blob 的数据内存排布。

【定义】

```
typedef enum hiSVP_BLOB_TYPE_E
{
    SVP_BLOB_TYPE_S32      = 0x0,
    SVP_BLOB_TYPE_U8       = 0x1,
    /*channel = 3*/
    SVP_BLOB_TYPE_YVU420SP = 0x2,
    /*channel = 3*/
    SVP_BLOB_TYPE_YVU422SP = 0x3,
    SVP_BLOB_TYPE_VEC_S32  = 0x4,
    SVP_BLOB_TYPE_SEQ_S32  = 0x5,
    SVP_BLOB_TYPE_BUTT
}SVP_BLOB_TYPE_E;
```

【成员】

成员名称	描述
SVP_BLOB_TYPE_S32	Blob 数据元素为 S32 类型，参考图 2-2
SVP_BLOB_TYPE_U8	Blob 数据元素为 U8 类型，参考图 2-3
SVP_BLOB_TYPE_YVU420SP	Blob 数据内存排布为 YVU420SP，参考图 2-4。
SVP_BLOB_TYPE_YVU422SP	Blob 数据内存排布为 YVU422SP，参考图 2-5。
SVP_BLOB_TYPE_VEC_S32	Blob 中存储向量，每个元素为 S32 类型，参考图 2-6。
SVP_BLOB_TYPE_SEQ_S32	Blob 中存储序列，数据元素为 S32 类型，排布见图 2-7。

【注意事项】

无。



【相关数据类型及接口】

SVP_BLOB_S

SVP_BLOB_S

【说明】

定义多个连续存放的 blob 信息。

【定义】

```
typedef struct hiSVP_BLOB_S
{
    SVP_BLOB_TYPE_E enType;    /*Blob type*/
    HI_U32 u32Stride;          /*Stride, a line bytes num*/
    HI_U64 u64VirAddr;         /*virtual addr*/
    HI_U64 u64PhyAddr;         /*physical addr*/
    HI_U32 u32Num;             /*N: frame num or sequence num,correspond to caffe
blob's n*/
    union
    {
        struct
        {
            HI_U32 u32Width;    /*W: frame width, correspond to caffe blob's w*/
            HI_U32 u32Height;   /*H: frame height, correspond to caffe blob's h*/
            HI_U32 u32Chn;      /*C: frame channel,correspond to caffe blob's c*/
        }stWhc;
        struct
        {
            HI_U32 u32Dim;       /*D: vector dimension*/
            HI_U64 u64VirAddrStep; /*T: virtual adress of time steps array in each
sequence*/
        }stSeq;
    }unShape;
}SVP_BLOB_S;
```

【成员】



成员名称	描述
enType	Blob 类型。 enType 取值范围[SVP_BLOB_TYPE_S32, SVP_BLOB_TYPE_BUTT)。
u32Stride	Blob 中单行数据的对齐后的字节数。
u64VirAddr	Blob 首虚拟地址。
u64PhyAddr	Blob 首物理地址。
u32Num	表示连续内存块的数目，若一帧数据对应一个块，则表示 blob 中有 u32Num 帧。 enType 为 YVU420SP\YVU422SP\U8\SEQ_S32 时，取值范围[1, 256]。 enType 为 S32/ VEC_S32 时，取值范围[1, 5000]。
u32Width	Blob 的宽度。 enType 为 YVU420SP\YVU422SP\U8 时，取值范围[8, 4096]。 enType 为 S32 时，取值范围[1, 0xFFFFFFFF]。 enType 为 VEC_S32 时，取值范围[1, 0xFFFFFFFF]。
u32Height	Blob 的高度。 enType 为 YVU420SP\YVU422SP\U8 时，取值范围[8, 4096]。 enType 为 S32 时，取值范围[1, 0xFFFFFFFF]。 enType 为 VEC_S32 时，取值范围[1, 1]。
u32Chn	Blob 中的通道数。 enType 为 YVU420SP\YVU422SP 时，取值为 3 enType 为 U8 时，取值为 1 或 3 enType 为 S32 时，取值范围[1, 0xFFFFFFFF] enType 为 VEC_S32 时，取值范围[1, 1]
u32Dim	向量的长度，仅用作 RNN\LSTM 数据的表示。 enType 为 SEQ_S32 时，取值范围[1, 0xFFFFFFFF]



成员名称	描述
u64VirAddrStep	数组地址，数组元素表示每段序列有多少个向量。

【注意事项】

- Caffe 中不同数据内存块用来表示内存形状的数据如下表：

数据块	Dim0	Dim1	Dim2	Dim3
Image\Feature Map	N	C	H	W
FC(normal) vector	N	C	-	-
RNN\LSTM vector	T	N	D	-

- 对应于本文中的 blob 表示如下表：

数据块	Dim0	Dim1	Dim2	Dim3
Image\Feature Map	u32Num	32Chn	u32Height	u32Width
FC(normal) vector	u32Num	u32Width	-	-
RNN\LSTM vector	u64VirAddrStep[i]]	u32Num	u32Dim	-

- u32Stride 表示的是在 u32Width 方向和 u32Dim 方向上一行数据对齐后的字节数。
- enType 为 S32 时，(u32Chn * u32Height * u32Stride)的乘积结果取值范围必须为[1, 0xFFFFFFFF]。
- enType 为 SEQ_S32/ VEC_S32 时，u32Stride 的取值范围必须为[1, 0xFFFFFFFF]。

【相关数据类型及接口】

[SVP_BLOB_TYPE_E](#)

SVP_SRC_BLOB_S

【说明】



定义源序列。

【定义】

```
typedef SVP_BLOB_S SVP_SRC_BLOB_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

- SVP_BLOB_S
- SVP_DST_BLOB_S

SVP_DST_BLOB_S

【说明】

定义输出序列。

【定义】

```
typedef SVP_BLOB_S SVP_DST_BLOB_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

- SVP_BLOB_S
- SVP_SRC_BLOB_S

SVP_NNIE_ID_E

【说明】

定义 NNIE 硬件的 ID 枚举类型。

【定义】



Hi3559AV100:

```
typedef enum hiSVP_NNIE_ID_E
{
    SVP_NNIE_ID_0      = 0x0,
    SVP_NNIE_ID_1      = 0x1,
    SVP_NNIE_ID_BUTT
}SVP_NNIE_ID_E;
```

Hi3519AV100/Hi3516CV500/Hi3516DV300/Hi3559V200:

```
typedef enum hiSVP_NNIE_ID_E
{
    SVP_NNIE_ID_0      = 0x0,
    SVP_NNIE_ID_BUTT
}SVP_NNIE_ID_E;
```

【成员】

成员名称	描述
SVP_NNIE_ID_0	表示下标为 0 的 NNIE 引擎。
SVP_NNIE_ID_1	表示下标为 1 的 NNIE 引擎。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_RUN_MODE_E

【说明】

定义运行模式。

【定义】

```
typedef enum hiSVP_NNIE_RUN_MODE_E
{
    SVP_NNIE_RUN_MODE_CHIP      = 0x0, /* on SOC chip running */

```



```
SVP_NNIE_RUN_MODE_FUNC_SIM = 0x1, /* functional simulaion */  
SVP_NNIE_RUN_MODE_BUTT  
}SVP_NNIE_RUN_MODE_E;
```

【成员】

成员名称	描述
SVP_NNIE_RUN_MODE_CHIP	表示只能用于在 Chip 上运行。
SVP_NNIE_RUN_MODE_FUNC_SIM	表示只能用于 PC 端的功能仿真。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_NET_TYPE_E

【说明】

定义网络类型。

【定义】

```
typedef enum hiSVP_NNIE_NET_TYPE_E  
{  
    SVP_NNIE_NET_TYPE_CNN      = 0x0, /* Normal cnn net */  
    SVP_NNIE_NET_TYPE_ROI      = 0x1, /* With ROI input cnn net*/  
    SVP_NNIE_NET_TYPE_RECURRENT = 0x2, /* RNN or LSTM net */  
    SVP_NNIE_NET_TYPE_BUTT  
}SVP_NNIE_NET_TYPE_E;
```

【成员】

成员名称	描述
SVP_NNIE_NET_TYPE_CNN	普通的 CNN\DNN 网络类型。
SVP_NNIE_NET_TYPE_ROI	有 RPN 层输出框信息进行框信息以及置信度回归的网络类型。



成员名称	描述
SVP_NNIE_NET_TYPE_RECURRENT	循环神经网络类型。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_ROIPOOL_TYPE_E

【说明】

定义 RoiPooling 的类型。

【定义】

```
typedef enum hiSVP_NNIE_ROIPOOL_TYPE_E
{
    SVP_NNIE_ROIPOOL_TYPE_NORMAL = 0x0,    /* Roipooling*/
    SVP_NNIE_ROIPOOL_TYPE_PS      = 0x1,    /* Position-Sensitive roipooling */
    SVP_NNIE_ROIPOOL_TYPE_BUTT
}SVP_NNIE_ROIPOOL_TYPE_E;
```

【成员】

成员名称	描述
SVP_NNIE_ROIPOOL_TYPE_NORMAL	普通模式下的 RoiPooling。
SVP_NNIE_ROIPOOL_TYPE_PS	Position-Sensitive RoiPooling。

【注意事项】

无。

【相关数据类型及接口】

无。



SVP_NNIE_NODE_S

【说明】

定义网络输入输出节点类型。

【定义】

```
typedef struct hiSVP_NNIE_NODE_S
{
    SVP_BLOB_TYPE_E enType;
    union
    {
        struct
        {
            HI_U32 u32Width;
            HI_U32 u32Height;
            HI_U32 u32Chn;
        }stWhc;
        HI_U32 u32Dim;
    }unShape;
    HI_U32 u32NodId;
    HI_CHAR szName[SVP_NNIE_NODE_NAME_LEN];
}SVP_NNIE_NODE_S;
```

【成员】

成员名称	描述
enType	节点的类型。
u32Width	节点内存形状的宽。 enType 为 YVU420SP\YVU422SP\U8 时，取值范围[8, 4096] enType 为 S32 时，取值范围[1, 0xFFFFFFFF] enType 为 VEC_S32 时，取值范围[1, 0xFFFFFFFF]



成员名称	描述
u32Height	节点内存形状的高。 enType 为 YVU420SP\YVU422SP\U8 时, 取值范围[8, 4096] enType 为 S32 时, 取值范围[1, 0xFFFFFFFF] enType 为 VEC_S32 时, 取值范围[1, 1]
u32Chn	节点内存形状的通道数。 enType 为 YVU420SP\YVU422SP 时, 取值为 3 enType 为 U8 时, 取值为 1 或 3 enType 为 S32 时, 取值范围[1, 0xFFFFFFFF] enType 为 VEC_S32 时, 取值范围[1, 1]
u32Dim	节点内存的向量维度。 enType 为 SEQ_S32 时, 取值范围[1, 0xFFFFFFFF]
u32NodeId	节点在网络中的 Id。
szName	节点名称。

【注意事项】

无。

【相关数据类型及接口】

[SVP_NNIE_SEG_S](#)

SVP_NNIE_NODE_NAME_LEN

【说明】

定义节点名称长度。

【定义】

```
#define SVP_NNIE_NODE_NAME_LEN    32 /*NNIE node name length*/
```

【成员】

无。



【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_MAX_NET_SEG_NUM

【说明】

定义最大网络分段数。

【定义】

```
#define SVP_NNIE_MAX_NET_SEG_NUM      8 /*NNIE max segment num that the net being  
cut into*/
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_MAX_INPUT_NUM

【说明】

定义最大网络输入节点数。

【定义】

```
#define SVP_NNIE_MAX_INPUT_NUM        16 /*NNIE max input num in each seg*/
```

【成员】

无。

【注意事项】

无。



【相关数据类型及接口】

无。

SVP_NNIE_MAX_OUTPUT_NUM

【说明】

定义最大网络输出节点数。

【定义】

```
#define SVP_NNIE_MAX_OUTPUT_NUM 16 /*NNIE max output num in each seg*/
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_MAX_ROI_LAYER_NUM_OF_SEG

【说明】

定义单个网络段中最大包含 RoiPooling 以及 PSRoiPooling 的 layer 数。

【定义】

```
#define SVP_NNIE_MAX_ROI_LAYER_NUM_OF_SEG 2 /*NNIE max roi layer num in each seg*/
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。



SVP_NNIE_MAX_ROI_LAYER_NUM

【说明】

定义网络中最多包含的 RoiPooling 以及 PSRoiPooling layer 数。

【定义】

```
#define SVP_NNIE_MAX_ROI_LAYER_NUM      4  /*NNIE max roi layer num*/
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_SEG_S

【说明】

定义网络段结构体。

【定义】

```
typedef struct hiSVP_NNIE_SEG_S
{
    SVP_NNIE_NET_TYPE_E enNetType;
    HI_U16                u16SrcNum;
    HI_U16                u16DstNum;
    HI_U16                u16RoiPoolNum;
    HI_U16                u16MaxStep;
    HI_U32                u32InstOffset;
    HI_U32                u32InstLen;
    SVP_NNIE_NODE_S       astSrcNode[SVP_NNIE_MAX_INPUT_NUM];
    SVP_NNIE_NODE_S       astDstNode[SVP_NNIE_MAX_OUTPUT_NUM];
    HI_U32                au32RoIdx[SVP_NNIE_MAX_ROI_LAYER_NUM_OF_SEG]; /*Roipooling
info index*/
}SVP_NNIE_SEG_S;
```

【成员】



成员名称	描述
enNetType	网络段的类型。
u16SrcNum	网络段的输入节点数。 取值范围：[1, 16]
u16DstNum	网络段的输出节点数。 取值范围：[1, 16]
u16RoiPoolNum	网络段中包含的 RoiPooling 以及 PSRoiPooling layer 数。 单段网络中可包含 xRoiPooling 层个数为[0, SVP_NNIE_MAX_ROI_LAYER_NUM_OF_SEG]，整个网络中可包含 xRoiPooling 层个数为[0, SVP_NNIE_MAX_ROI_LAYER_NUM]。
u16MaxStep	RNN/LSTM 网络中序列的最大“帧数”。 取值范围：[1, 1024]
astSrcNode[i]	网络段的第 i 个输入节点信息。
astDstNode[i]	网络段的第 i 个输出节点信息。
au32RoIdx[i]	网络段的第 i 个 RoiPooling 或者 PsRoiPooling 在 SVP_NNIE_MODEL_S 中 SVP_NNIE_ROIPOOL_INFO_S 数组的下标。

【注意事项】

无。

【相关数据类型及接口】

[SVP_NNIE_MODEL_S](#)

SVP_NNIE_MODEL_S

【说明】

定义 NNIE 模型结构体。

【定义】

```
typedef struct hiSVP_NNIE_MODEL_S
```



```
{  
    SVP_NNIE_RUN_MODE_E    enRunMode;  
    HI_U32                  u32TmpBufSize; /*temp buffer size*/  
    HI_U32                  u32NetSegNum;  
    SVP_NNIE_SEG_S          astSeg[SVP_NNIE_MAX_NET_SEG_NUM];  
    SVP_NNIE_ROIPOOL_INFO_S astRoiInfo[SVP_NNIE_MAX_ROI_LAYER_NUM]; /*ROI Pooling  
info*/  
    SVP_MEM_INFO_S          stBase;  
}SVP_NNIE_MODEL_S;
```

【成员】

成员名称	描述
enRunMode	网络模型运行模式。
u32TmpBufSize	辅助内存 tmpBuf 大小。 取值范围：(0, 4294967295]
u32NetSegNum	网络模型中 NNIE 执行的网络分段数。 取值范围：[1, 8]
astSeg	网络在 NNIE 引擎上执行的段信息。
astRoiInfo	网络模型中 RoiPooling 以及 PsRoiPooling 的信息数组。
stBase	网络其他信息。

【注意事项】

无。

【相关数据类型及接口】

无。

SVP_NNIE_FORWARD_CTRL_S

【说明】

CNN/DNN/RNN 网络预测控制参数。



【定义】

```
typedef struct hiSVP_NNIE_FORWARD_CTRL_S
{
    HI_U32          u32SrcNum;      /* input node num, [1, 16] */
    HI_U32          u32DstNum;      /* output node num, [1, 16]*/
    HI_U32          u32NetSegId;    /* net segment index running on NNIE */
    SVP_NNIE_ID_E   enNniel;       /* device target which running the seg*/
    SVP_MEM_INFO_S  stTmpBuf;       /* auxiliary temp mem */
    SVP_MEM_INFO_S  stTskBuf;       /* auxiliary task mem */
}SVP_NNIE_FORWARD_CTRL_S;
```

【成员】

成员名称	描述
u32SrcNum	NNIE 执行网络段输入节点个数。 取值范围：[1, 16]。
u32DstNum	NNIE 执行网络段输出节点个数。 取值范围：[1, 16]。
u32NetSegId	网络段的段序号。 取值范围：[0, 8)，并且需要小于执行网络的分段数。
enNniel	执行网络段的 NNIE 引擎 ID。 取值范围：[SVP_NNIE_ID_0, SVP_NNIE_ID_BUTT)。
stTmpBuf	辅助内存。
stTskBuf	辅助内存。

【注意事项】

- stTmpBuf 和 stTskBuf 当不再有任务使用时才能被释放。
- 调用 Forward 开始执行任务后，在任务完成之前，stTmpBuf 和 stTskBuf 所指向的内存，不能被其他任务使用。

【相关数据类型及接口】

无。



SVP_NNIE_FORWARD_WITHBBOX_CTRL_S

【说明】

有 Bbox 输入的目标检测网络预测控制参数。

【定义】

```
typedef struct hiSVP_NNIE_FORWARD_WITHBBOX_CTRL_S
{
    HI_U32          u32SrcNum;      /* input node num, [1, 16] */
    HI_U32          u32DstNum;      /* output node num, [1, 16]*/
    HI_U32          u32ProposalNum; /* element num of roi array */
    HI_U32          u32NetSegId;    /* net segment index running on NNIE */
    SVP_NNIE_ID_E   enNnield;      /* device target which running the seg*/
    SVP_MEM_INFO_S  stTmpBuf;       /* auxiliary temp mem */
    SVP_MEM_INFO_S  stTskBuf;       /* auxiliary task mem */
}SVP_NNIE_FORWARD_WITHBBOX_CTRL_S;
```

【成员】

成员名称	描述
u32SrcNum	NNIE 执行网络段输入节点个数。 取值范围：[1, 16]。
u32DstNum	NNIE 执行网络段输出节点个数。 取值范围：[1, 16]。
u32ProposalNum	生成 Bbox 网络层的 Proposal 层数目，对应 HI_MPI_SVP_NNIE_ForwardWithBbox 接口中 astBbox[]数组的元素个数。 取值范围：[1, SVP_NNIE_MAX_ROI_LAYER_NUM_OF_SEG]。
u32NetSegId	网络段的段序号。 取值范围：[0, 8)，并且需要小于执行网络的分段数
enNnield	执行网络段的 NNIE 引擎 ID。 取值范围：[SVP_NNIE_ID_0, SVP_NNIE_ID_BUTT)。



成员名称	描述
stTmpBuf	辅助内存。
stTskBuf	辅助内存。

【注意事项】

- stTmpBuf 和 stTskBuf 当不再有任务使用时才能被释放。
- 调用 ForwardWithBbox 开始执行任务后，在任务完成之前，stTmpBuf 和 stTskBuf 所指向的内存，不能被其他任务使用。

【相关数据类型及接口】

无。

2.5 错误码

NNIE 引擎 API 错误码如表 2-2 所示。

表2-2 NNIE 引擎 API 错误码

错误代码	宏定义	描述
0xA0338001	HI_ERR_SVP_NNIE_INVALID_DEVID	设备 ID 超出合法范围
0xA0338002	HI_ERR_SVP_NNIE_INVALID_CHNID	通道组号错误或无效区域句柄
0xA0338003	HI_ERR_SVP_NNIE_ILLEGAL_PARAM	参数超出合法范围
0xA0338004	HI_ERR_SVP_NNIE_EXIST	重复创建已存在的设备、通道或资源
0xA0338005	HI_ERR_SVP_NNIE_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
0xA0338006	HI_ERR_SVP_NNIE_NULL_PTR	函数参数中有空指针
0xA0338007	HI_ERR_SVP_NNIE_NOT_CONFIG	模块没有配置
0xA0338008	HI_ERR_SVP_NNIE_NOT_SUPPORT	不支持的参数或者功能



错误代码	宏定义	描述
0xA0338009	HI_ERR_SVP_NNIE_NOT_PERM	该操作不允许，如试图修改静态配置参数
0xA033800C	HI_ERR_SVP_NNIE_NOMEM	分配内存失败，如系统内存不足
0xA033800D	HI_ERR_SVP_NNIE_NOBUF	分配缓存失败，如申请的图像缓冲区太大
0xA033800E	HI_ERR_SVP_NNIE_BUF_EMPTY	缓冲区中无数据
0xA033800F	HI_ERR_SVP_NNIE_BUF_FULL	缓冲区中数据满
0xA0338010	HI_ERR_SVP_NNIE_NOTREADY	系统没有初始化或没有加载相应模块
0xA0338011	HI_ERR_SVP_NNIE_BADADDR	地址非法
0xA0338012	HI_ERR_SVP_NNIE_BUSY	系统忙
0xA0338040	HI_ERR_SVP_NNIE_SYS_TIMEOUT	系统超时
0xA0338041	HI_ERR_SVP_NNIE_QUERY_TIMEOUT	Query 查询超时
0xA0338042	HI_ERR_SVP_NNIE_CFG_ERR	配置错误
0xA0338043	HI_ERR_SVP_NNIE_OPEN_FILE	打开文件失败
0xA0338044	HI_ERR_SVP_NNIE_READ_FILE	读文件失败
0xA0338045	HI_ERR_SVP_NNIE_WRITE_FILE	写文件失败

2.6 Proc 调试信息

2.6.1 概述

调试信息采用了 Linux 下的 proc 文件系统，可实时反映当前系统的运行状态，所记录的信息可供问题定位及分析时使用。

【文件目录】



/proc/umap

【信息查看方法】

- 在控制台上可以使用 cat 命令查看信息，cat /proc/umap/nnie；也可以使用其他常用的文件操作命令，例如 cp /proc/umap/nnie ./，将文件拷贝到当前目录。
- 在应用程序中可以将上述文件当作普通只读文件进行读操作，例如 fopen、fread 等。

说明

参数在描述时有以下 2 种情况需要注意：

- 取值为{0, 1}的参数，如未列出具体取值和含义的对应关系，则参数为 1 时表示肯定，为 0 时表示否定。
- 取值为{aaa, bbb, ccc}的参数，未列出具体取值和含义的对应关系，但可直接根据取值 aaa、bbb 或 ccc 判断参数含义

2.6.2 Proc 信息说明

【调试信息】

```
# cat /proc/umap/nnie

[NNIE] Version: [Hi3559AV100_MPP_Vx.x.x.x B0xx Release], Build Time[mm dd yyyy, hh:mm:ss]
-----NNIE MODULE PARAM-----
nnie_save_power      nnie_max_tskbuf_num
          1              32

-----NNIE QUEUE INFO-----
CoreId    Wait    Busy WaitCurId WaitEndId BusyCurId BusyEndId
    0         0      -1         0         0         0         0
    1         0      -1         0         0         0         0

----- NNIE TASK INFO-----
CoreId    Hnd    TaskFsh    LastId    TaskId    HndWrap    FshWrap
    0         0         0         0         0         0         0
    1         0         0         0         0         0         0

FreeTskBufNum    UseTskBufNum
    32              0
    32              0

----- NNIE RUN-TIME INFO-----
```




CoreId	LastInst	CntPerSec	MaxCntPerSec	TotalIntCntLastSec			
	0	0	0	0	0		
	1	0	0	0	0		
TotalIntCnt	QTCnt	STCnt	CfgErrCnt	CostTm	MCostTm		
	0	0	0	0	0	0	
	0	0	0	0	0	0	
CostTmPerSec	MCostTmPerSec	TotalIntCostTm		HwUtilization		RunTm	
0	0	0		0		0%	0
0	0	0		0		0%	0
-----NNIE INVOKE INFO-----							
CoreId	Forward	ForwardWithBbox					
	0	0					
	1	0					

【调试信息分析】

记录当前 NNIE 工作状态资源信息，主要包括 NNIE 队列状态信息，任务状态信息，运行时状态信息和调用信息。

【参数说明】

参数		描述
NNIE MODULE PARAM NNIE 模块参数	nnie_save_power	低功耗标志。 0: 关闭低功耗; 1: 打开低功耗。
	nnie_max_tskbuf_num	最大能记录的 TskBuf 个数
NNIE QUEUE INFO NNIE 队列信息	CoreId	NNIE 核 ID。
	Wait	等待队列编号 (0 或 1) 。
	Busy	正在调度队列编号 (0, 1 或-1) , -1 表示 NNIE 硬件空闲。
	WaitCurId	等待队列的首个有效任务 ID。



参数		描述
	WaitEndId	等待队列的末尾有效任务 ID + 1。
	BusyCurId	调度队列的首个有效任务 ID。
	BusyEndId	调度队列的末尾有效任务 ID + 1。
NNIE TASK INFO NNIE TASK 相关信息	CoreId	NNIE 核 ID。
	Hnd	当前可分配的任务 handle 号。
	TaskFsh	当前已完成任务的个数。
	LastId	上一次中断完成的任务 ID。
	TaskId	本次中断完成的任务 ID。
	HndWrap	用户 handle 号分配发生回写的次数。
	FshWrap	完成任务数发生回写的次数。
	FreeTskBufNum	空闲 TskBuf 链表节点数。
	UseTskBufNum	已使用 TskBuf 链表节点数。
NNIE RUN-TIME INFO NNIE 运行时相关信息	CoreId	NNIE 核 ID。
	LastInst	用户最后一次提交任务时传入的 blntant 值。
	CntPerSec	最近一次的 1 秒内中断执行次数。
	MaxCntPerSec	历史上的 1 秒内最大的中断执行次数。
	TotalIntCntLastSec	上一秒上报中断总次数。
	TotalIntCnt	NNIE 产生中断的总次数。
	QTCnt	NNIE 查询超时中断次数。
	STCnt	NNIE 系统超时次数。
	CfgErrCnt	NNIE 配置错误中断次数。
	CostTm	最近一次执行中断的执行耗时。 单位：us。



参数		描述
	MCostTm	执行一次中断的最大耗时。 单位：us。
	CostTmPerSec	最近一秒执行中断的执行耗时。 单位：us。
	MCostTmPerSec	历史上一秒执行中断的最大执行耗时。 单位：us。
	TotalIntCostTm	中断处理总时间。 单位：us。
	HwUtilization	NNIE 利用率（一秒内 NNIE 执行时间所占比例）。
	RunTm	NNIE 运行总时间。 单位：s。
NNIE INVOKE INFO NNIE 调用信息	CoreId	NNIE 核 ID。
	Forward	NNIE Forward 调用次数。
	ForwardWithBbox	NNIE ForwardWithBbox 调用次数



3 Runtime

3.1 概述

Runtime 是基于神经网络推断引擎 NNIE 开发的一套软件系统。用户基于 Runtime 开发智能分析方案，不需要关注调度等逻辑，最大化复用 nnie 硬件。

3.2 功能描述

3.2.1 重要概念

- 网络模型组
现网的实际使用场景在很多情况下是一个模型下接另外一个模型，Runtime 模式下，客户可以将有级联关系的模型构成一个模型组。系统会为每个模型组分配一个 handle，不同的 handle 表示不同的模型组。
- 连接器 Connector
上述模型组构建过程中，模型与模型之间需要有连接器对象，通过连接器对象，可以使两个网络连接起来，且用户可以在内部定制业务逻辑。
- 优先级
模型组和模型组之间，用户可以根据业务需要设置模型组的优先级，Runtime 根据优先级选择模型组运行。

3.3 API 参考

Runtime 模块提供了创建任务和查询任务的基本接口。



- [HI_SVPRT_RUNTIME_Init](#): 初始化 Runtime 运行环境。
- [HI_SVPRT_RUNTIME_LoadModelGroup](#): 从用户加载到 buf 中的模型中解析出网络模型组，同步操作。
- [HI_SVPRT_RUNTIME_ForwardGroupSync](#): 多节点输入输出的网络预测，同步操作。
- [HI_SVPRT_RUNTIME_ForwardGroupASync](#): 多节点输入输出的网络预测，异步操作。
- [HI_SVPRT_RUNTIME_UnloadModelGroup](#): 卸除已加载的模型组。
- [HI_SVPRT_RUNTIME_DeInit](#): 去初始化 Runtime 环境。
- [HI_NodePlugin_getNodeType](#): 获取插件类型，当前插件类型为 Proposal。
- [HI_NodePlugin_Compute](#): 计算插件的函数。

HI_SVPRT_RUNTIME_Init

【描述】

初始化 Runtime 运行环境。

【语法】

```
HI_S32 HI_SVPRT_RUNTIME_Init(IN const HI_CHAR* pszGlobalSetting, IN  
HI\_RUNTIME\_MEM\_CTRL\_S *pstMemCtrl);
```

【参数】

参数名称	描述	输入/输出
pszGlobalSetting	初始化全局配置，cpu 线程的亲和度，全局插件库等	输入
pstMemCtrl	初始化内存管理函数，填入 HI_NULL 时，默认使用 mmz 内存	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。



【需求】

- 头文件：hi_runtime_comm.h、hi_runtime_api.h
- 库文件：libsvpruntime.a\libsvpruntime.so（PC 上模拟用 libsvpruntime.a\svpruntime.lib\svpruntime.dll）

【举例】

请参考发布包中的 sample 目录下 hirt/src。

【相关主题】

无。

HI_SVPRT_RUNTIME_LoadModelGroup

【描述】

从用户事先加载到 buf 中的模型组中解析出网络模型。

【语法】

```
HI_S32 HI_SVPRT_RUNTIME_LoadModelGroup(IN const HI_CHAR* pstModelGroupConfig, IN HI_RUNTIME_GROUP_INFO_S *pstModelGroupAttr, OUT HI_RUNTIME_GROUP_HANDLE* phGroupHandle);
```

【参数】

参数名称	描述	输入/输出
pstModelGroupConfig	网络模型组的网络结构，prototxt 格式的网络拓扑结构	输入
pstModelGroupAttr	网络模型组内部信息结构体	输入
phGroupHandle	Group 句柄	输出

【返回值】

返回值	描述
0	成功。



返回值	描述
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_runtime_comm.h、hi_runtime_api.h
- 库文件：libsvpruntime.a\libsvpruntime.so（PC 上模拟用 libsvpruntime.a\svpruntime.lib\svpruntime.dll）

【注意】

无。

【举例】

请参考发布包中的 sample 目录下 hirt/src。

【相关主题】

无。

HI_SVPRT_RUNTIME_ForwardGroupSync

【描述】

多节点输入输出的网络预测，同步接口。

【语法】

```
HI_S32 HI_SVPRT_RUNTIME_ForwardGroupSync(IN const HI_RUNTIME_GROUP_HANDLE  
hGroupHandle, IN const HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_PTR pstSrc, OUT  
HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR pstDst, IN HI_U64 u64SrcId);
```

【参数】

参数名称	描述	输入/输出
hGroupHandle	Group 句柄。 不能为空。	输入
pstSrc	模型组中各模型的多个节点输入构成的 BLOB 对象，支持多帧同时输入。	输入



参数名称	描述	输入/输出
pstDst	模型组中各模型的网络段的多个节点输出，包含用户标记需要上报输出的中间层结果，以及模型组的最终结果。	输出
u64SrcId	传入的 Src ID。	输入

参数名称	支持类型	地址对齐	分辨率
pstSrc	YVU420SP/ YVU422SP/ U8/S32/ VEC_S32/ SEQ_S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	8x8 ~ 4096x4096 向量维度: 1~0xFFFFFFFF
pstDst	S32/VEC_S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	8x8 ~ 4096x4096 向量维度: 1~0xFFFFFFFF

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

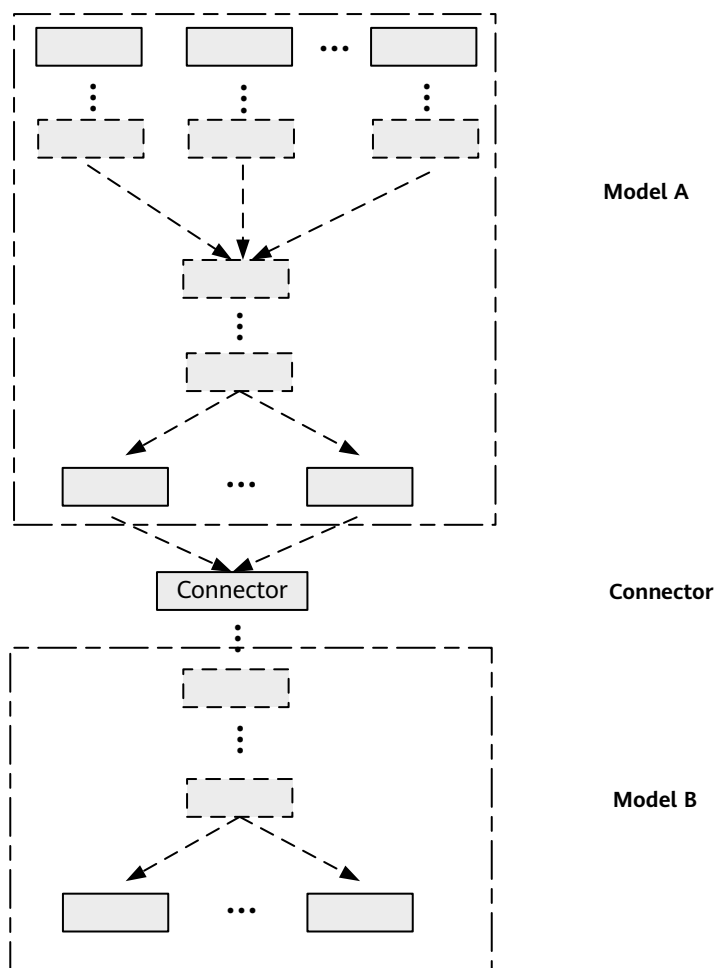
- 头文件：hi_runtime_comm.h、hi_runtime_api.h
- 库文件：libsvpruntime.a\libsvpruntime.so（PC 上模拟用
libsvpruntime.a\svpruntime.lib\svpruntime.dll）

【注意】

- U8 图像输入只支持 1 通道（灰度图）和 3 通道(RGB 图)；

- 多个 Blob 输入输出时，配合编译器输出的 dot 描述文件生成的 dot 图示，可以看到 Blob 跟层的对应关系。

图3-1 模型组输入输出网络示意图

**【举例】**

请参考发布包中的 sample 目录下 hirt/src。

【相关主题】

无。



HI_SVPRT_RUNTIME_ForwardGroupASync

【描述】

多节点输入输出的网络预测，异步接口。

【语法】

```
HI_S32 HI_SVPRT_RUNTIME_ForwardGroupASync(IN const HI_RUNTIME_GROUP_HANDLE  
hGroupHandle, IN const HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_PTR pstSrc, OUT  
HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR pstDst, IN HI_U64 u64SrcId, IN  
HI_RUNTIME_Forward_Callback pCbFun);
```

【参数】

参数名称	描述	输入/输出
hGroupHandle	Group 句柄。 不能为空。	输入
pstSrc	模型组中各模型的多个节点输入构成的 BLOB 对象，支持多帧同时输入。	输入
pstDst	模型组中各模型的网络段的多个节点输出，包含用户标记需要上报输出的中间层结果，以及模型组的最终结果。	输出
u64SrcId	传入的 Src ID。	输入
pCbFun	结果回调函数。	输入

参数名称	支持类型	地址对齐	分辨率
pstSrc	YVU420SP/ YVU422SP/ U8/S32/ VEC_S32/ SEQ_S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	8x8 ~ 4096x4096 向量维度: 1~0xFFFFFFFF



参数名称	支持类型	地址对齐	分辨率
pstDst	S32/VEC_S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	8x8 ~ 4096x4096 向量维度: 1~0xFFFFFFFF

【返回值】

返回值	描述
0	成功。
非 0	失败, 参见 错误码 。

【需求】

- 头文件: hi_runtime_comm.h、hi_runtime_api.h
- 库文件: libsvpruntime.a\libsvpruntime.so (PC 上模拟用
libsvpruntime.a\svpruntime.lib\svpruntime.dll)

【注意】

- U8 图像输入只支持 1 通道 (灰度图) 和 3 通道(RGB 图);
- 多个 Blob 输入输出时, 配合编译器输出的 dot 描述文件生成的 dot 图示, 可以看到 Blob 跟层的对应关系;
- 示意图请参考 [HI_SVPRT_RUNTIME_ForwardGroupSync](#) 接口部分。

【举例】

请参考发布包中的 sample 目录下 hirt/src。

【相关主题】

无。

HI_SVPRT_RUNTIME_UnloadModelGroup

【描述】

卸载模型组。



【语法】

```
HI_S32 HI_SVPRT_RUNTIME_UnloadModelGroup(IN const HI_RUNTIME_GROUP_HANDLE  
hGroupHandle);
```

【参数】

参数名称	描述	输入/输出
hGroupHandle	Group 句柄。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_runtime_comm.h、hi_runtime_api.h
- 库文件：libsvpruntime.a\libsvpruntime.so（PC 上模拟用
libsvpruntime.a\svpruntime.lib\svpruntime.dll）

【注意】

无。

【举例】

请参考发布包中的 sample 目录下 hirt/src。

【相关主题】

无。

HI_SVPRT_RUNTIME_DeInit

【描述】

去初始化 Runtime 运行环境

【语法】



```
HI_S32 HI_SVPRT_RUNTIME_Delinit();
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_runtime_comm.h、hi_runtime_api.h
- 库文件：libsvpruntime.a\libsvpruntime.so（PC 上模拟用 libsvpruntime.a\svpruntime.lib\svpruntime.dll）

【注意】

无。

【举例】

请参考发布包中的 sample 目录下 hirt/src。

【相关主题】

无。

HI_NodePlugin_getNodeType

【描述】

获取插件类型，当前插件类型为 Proposal。

【语法】

```
HI_S32 HI_NodePlugin_getNodeType(HI_CHAR pszNodeType[])
```

【参数】



参数名称	描述	输入/输出
pszNodeType	获取插件节点类型的字符数组	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_plugin.h、hi_runtime_comm.h、proposal_common.h
- 库文件：libruntime_plugin_proposal.a\libruntime_plugin_proposal.so（PC 上模拟用 libruntime_plugin_proposal.dll）

【注意】

- 该函数为自定义的一个 Proposal 插件的实现。由于当前 nnie 不支持 Proposal 层实现，所以将其实现为了一个插件。
- 如果用户有 nnie 不支持的层，可以照着该插件进行插件实现。
- 该函数为 runtime 库调用的函数，用户不需要直接调用。

【举例】

请参考发布包中的 sample 目录下 hirt/plugins。

【相关主题】

无。

HI_NodePlugin_Compute

【描述】

计算插件的函数。

【语法】

```
HI_S32 HI_NodePlugin_Compute(const HI_NodePlugin_Operand_S *pstInputs, HI_U32  
u32InputNum, HI_NodePlugin_Operand_S *pstOutputs,
```



```
HI_U32 u32Outputs, HI_NodePlugin_NodeParam_S* pstHyperParam,  
HI_NodePlugin_NodeParam_S* pstTrainingParam)
```

【参数】

参数名称	描述	输入/输出
pstInputs	输入给插件的 Blobs	输入
u32InputNum	输入给插件的 Blobs 的数量	输入
pstOutputs	插件输出的 Blobs	输出
u32Outputs	输出的 Blobs 数量	输出
pstHyperParam	插件配置的超参数	输入
pstTrainingParam	插件配置的训练参数	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，参见 错误码 。

【需求】

- 头文件：hi_plugin.h、hi_runtime_comm.h、proposal_common.h
- 库文件：libruntime_plugin_proposal.a\libruntime_plugin_proposal.so（PC 上模拟用 libruntime_plugin_proposal.dll）

【注意】

- 该函数为自定义的一个 Proposal 插件的实现。由于当前 nnie 不支持 Proposal 层实现，所以将其实现为了一个插件。
- 如果用户有 nnie 不支持的层，可以照着该插件进行插件实现。
- 该函数为 runtime 库调用的函数，用户不需要直接调用。
- pstTrainingParam 为本插件的保留字段。用户在实现自己的插件库的时候，可以根据自身情况使用该字段。



【举例】

请参考发布包中的 sample 目录下 hirt/plugins。

【相关主题】

无。

3.4 数据类型和数据结构

RUNTIME 相关数据类型、数据结构定义如下：

- [MAX_NAME_LEN](#)：模型和 BLOB 名称最大长度。
- [HI_RUNTIME_MEM_S](#)：定义一维内存信息。
- [HI_RUNTIME_BLOB_TYPE_E](#)：定义 blob 的数据内存排布。
- [HI_RUNTIME_GROUP_PRIORITY_E](#)：定义模型组的优先级。
- [HI_RUNTIME_BLOB_S](#)：定义多个连续存放的 blob 信息。
- [HI_RUNTIME_BLOB_ARRAY_S](#)：定义源序列。
- [HI_RUNTIME_SRC_BLOB_ARRAY_S](#)：定义输入序列。
- [HI_RUNTIME_DST_BLOB_ARRAY_S](#)：定义输出序列。
- [HI_RUNTIME_SRC_BLOB_ARRAY_PTR](#)：定义输入序列指针。
- [HI_RUNTIME_DST_BLOB_ARRAY_PTR](#)：定义输出序列指针。
- [HI_RUNTIME_GROUP_HANDLE](#)：定义模型组句柄。
- [HI_RUNTIME_GROUP_BLOB_S](#)：定义组对应的 BLOB 结构。
- [HI_RUNTIME_GROUP_BLOB_ARRAY_S](#)：定义组对应的 BLOB 数组信息。
- [HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_S](#)：定义组对应的输入 BLOB 数组信息。
- [HI_RUNTIME_GROUP_DST_BLOB_ARRAY_S](#)：定义组对应的输出 BLOB 数组信息。
- [HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_PTR](#)：定义组对应的输入 BLOB 数组信息指针。
- [HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR](#)：定义组对应的输出 BLOB 数组信息指针。
- [HI_RUNTIME_GROUP_SRC_BLOB_S](#)：组输入 BLOB。



- [HI_RUNTIME_GROUP_DST_BLOB_S](#): 组输出 BLOB。
- [HI_RUNTIME_MEM_CTRL_S](#): 内存管理结构体。
- [HI_RUNTIME_WK_INFO_S](#): WK 模型数据结构体。
- [HI_RUNTIME_WK_INFO_ARRAY_S](#): 定义 WK 模型数组信息。
- [HI_RUNTIME_COP_ATTR_S](#): 定义用户自定义层属性信息。
- [HI_RUNTIME_COP_ATTR_ARRAY_S](#): 定义用户自定义层数组信息。
- [HI_RUNTIME_CONNECTOR_ATTR_S](#): 定义 Connector 对象的属性信息。
- [HI_RUNTIME_CONNECTOR_ATTR_ARRAY_S](#): 定义 Connector 对象数组信息。
- [HI_RUNTIME_GROUP_INFO_S](#): 定义 Runtime 组信息。
- [HI_RUNTIME_FORWARD_STATUS_CALLBACK_E](#): 定义异步 Forward 的状态信息。
- [HI_RUNTIME_Forward_Callback](#): 定义异步 Forward 的回调函数。
- [HI_RUNTIME_Connector_Compute](#): 定义 Connector 的回调函数。
- [MAX_OPERAND_NAME_LEN](#): 定义对插件进行的操作的描述字符串的最大长度。
- [HI_NodePlugin_Shape_S](#): 定义插件输入输出的 HWC 维度的结构体。
- [HI_NodePlugin_ElemType_E](#): 定义插件数据对齐方式的枚举。
- [HI_NodePlugin_Operand_S](#): 定义插件输入输出的数据和属性的结构体。
- [HI_NodePlugin_NodeParam_S](#): 定义插件超参数和训练参数的结构体。

MAX_NAME_LEN

【说明】

模型和 BLOB 名称最大长度。

【定义】

```
#define MAX_NAME_LEN 64
```

【成员】

无

【注意事项】

无



【相关数据类型及接口】

无

HI_RUNTIME_MEM_S

【说明】

定义一维内存信息。

【定义】

```
typedef struct hiRUNTIME_MEM_S
{
    HI_U64  u64PhyAddr; /* RW;The physical address of the memory */
    HI_U64  u64VirAddr; /* RW;The virtual address of the memory */
    HI_U32  u32Size;    /* RW;The size of memory */
}HI_RUNTIME_MEM_S;
```

【成员】

成员名称	描述
u64VirAddr	内存块虚拟地址。
u64PhyAddr	内存块物理地址。
u32Size	内存块字节数。见 图 2-8 。

【注意事项】

无。

【相关数据类型及接口】

无

HI_RUNTIME_BLOB_TYPE_E

【说明】

定义 blob 的数据内存排布。

【定义】

```
typedef enum hiRUNTIME_BLOB_TYPE_E
```



```
{  
    HI_RUNTIME_BLOB_TYPE_S32      = 0x0,  
    HI_RUNTIME_BLOB_TYPE_U8       = 0x1,  
    HI_RUNTIME_BLOB_TYPE_YVU420SP = 0x2,  
    HI_RUNTIME_BLOB_TYPE_YVU422SP = 0x3,  
    HI_RUNTIME_BLOB_TYPE_VEC_S32  = 0x4,  
    HI_RUNTIME_BLOB_TYPE_SEQ_S32  = 0x5,  
    HI_RUNTIME_BLOB_TYPE_BUTT  
}HI_RUNTIME_BLOB_TYPE_E;
```

【成员】

成员名称	描述
HI_RUNTIME_BLOB_TYPE_S32	Blob 数据元素为 S32 类型
HI_RUNTIME_BLOB_TYPE_U8	Blob 数据元素为 U8 类型
HI_RUNTIME_BLOB_TYPE_YVU420SP	Blob 数据内存排布为 YVU420SP
HI_RUNTIME_BLOB_TYPE_YVU422SP	Blob 数据内存排布为 YVU422SP
HI_RUNTIME_BLOB_TYPE_VEC_S32	Blob 中存储向量，每个元素为 S32 类型
HI_RUNTIME_BLOB_TYPE_SEQ_S32	Blob 中存储序列，数据元素为 S32 类型

【注意事项】

内存布局方式请参考 NNIE 的 [SVP_BLOB_TYPE_E](#)。

【相关数据类型及接口】

无

HI_RUNTIME_GROUP_PRIORITY_E

【说明】

定义模型组的优先级。

【定义】

```
typedef enum hiRUNTIME_GROUP_PRIORITY_E  
{
```



```
HI_RUNTIME_GROUP_PRIORITY_HIGHEST = 0x0,  
HI_RUNTIME_GROUP_PRIORITY_HIGH,  
HI_RUNTIME_GROUP_PRIORITY_MEDIUM,  
HI_RUNTIME_GROUP_PRIORITY_LOW,  
HI_RUNTIME_GROUP_PRIORITY_LOWEST,  
HI_RUNTIME_GROUP_PRIORITY_BUTT  
} HI_RUNTIME_GROUP_PRIORITY_E;
```

【成员】

成员名称	描述
HI_RUNTIME_GROUP_PRIORITY_HIGHEST	最高优先级
HI_RUNTIME_GROUP_PRIORITY_HIGH	高优先级
HI_RUNTIME_GROUP_PRIORITY_MEDIUM	中等优先级
HI_RUNTIME_GROUP_PRIORITY_LOW	低优先级
HI_RUNTIME_GROUP_PRIORITY_LOWEST	最低优先级

【注意事项】

无

【相关数据类型及接口】

无

HI_RUNTIME_BLOB_S

【说明】

定义多个连续存放的 blob 信息。

【定义】

```
typedef struct hiRUNTIME_BLOB_S  
{  
    SVP_BLOB_TYPE_E enBlobType;    /*Blob type*/  
    HI_U32 u32Stride;                /*Stride, a line bytes num*/  
    HI_U64 u64VirAddr;               /*virtual addr*/  
    HI_U64 u64PhyAddr;               /*physical addr*/  
}
```



```
HI_U32 u32Num;          /*N: frame num or sequence num,correspond to caffe blob's n*/
union
{
    struct
    {
        HI_U32 u32Width;    /*W: frame width, correspond to caffe blob's w*/
        HI_U32 u32Height;   /*H: frame height, correspond to caffe blob's h*/
        HI_U32 u32Chn;      /*C: frame channel,correspond to caffe blob's c*/
    }stWhc;
    struct
    {
        HI_U32 u32Dim;       /*D: vector dimension*/
        HI_U64 u64VirAddrStep; /*T: virtual adress of    time steps array in each
sequence*/
    }stSeq;
    }unShape;
}HI_RUNTIME_BLOB_S,*HI_RUNTIME_BLOB_PTR;
```

【成员】

成员名称	描述
enBlobType	Blob 类型。
u32Stride	Blob 中单行数据的对齐后的字节数。
u64VirAddr	Blob 首虚拟地址。
u64PhyAddr	Blob 首物理地址。
u32Num	表示连续内存块的数目，若一帧数据对应一个块，则表示 blob 中有 u32Num 帧。
u32Width	Blob 的宽度。
u32Height	Blob 的高度。
u32Chn	Blob 中的通道数。
u32Dim	向量的长度，仅用作 RNN\LSTM 数据的表示。
u64VirAddrStep	数组地址，数组元素表示每段序列有多少个向量。



【注意事项】

- Caffe 中不同数据内存块用来表示内存形状的数据如下表：

数据块	Dim0	Dim1	Dim2	Dim3
Image\Feature Map	N	C	H	W
FC(normal) vector	N	C	-	-
RNN\LSTM vector	T	N	D	-

- 对应于本文中的 blob 表示如下表：

数据块	Dim0	Dim1	Dim2	Dim3
Image\Feature Map	u32Num	32Chn	u32Height	u32Width
FC(normal) vector	u32Num	u32Width	-	-
RNN\LSTM vector	u64VirAddrStep[i]]	u32Num	u32Dim	-

- u32Stride 表示的是在 u32Width 方向和 u32Dim 方向上一行数据对齐后的字节数。

【相关数据类型及接口】

[HI_RUNTIME_BLOB_TYPE_E](#)

HI_RUNTIME_BLOB_ARRAY_S

【说明】

定义源序列。

【定义】

```
typedef struct hiRUNTIME_SRC_BLOB_ARRAY_S
{
    HI_U32 u32BlobNum;
    HI\_RUNTIME\_BLOB\_S *pstBlobs;
} HI_RUNTIME_BLOB_ARRAY_S, *HI_RUNTIME_BLOB_ARRAY_PTR;
```



【成员】

成员名称	描述
u32BlobNum	输入 Blob 数量
pstBlobs	输入 Blob 指针, 指向 Blob 数组

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_BLOB_S](#)

HI_RUNTIME_SRC_BLOB_ARRAY_S

【说明】

定义输入序列。

【定义】

```
typedef HI\_RUNTIME\_BLOB\_ARRAY\_S HI_RUNTIME_SRC_BLOB_ARRAY_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_BLOB_ARRAY_S](#)

HI_RUNTIME_DST_BLOB_ARRAY_S

【说明】

定义输出序列。

【定义】

```
typedef HI\_RUNTIME\_BLOB\_ARRAY\_S HI_RUNTIME_DST_BLOB_ARRAY_S;
```



【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_BLOB_ARRAY_S](#)

HI_RUNTIME_SRC_BLOB_ARRAY_PTR

【说明】

定义输入序列指针。

【定义】

```
typedef HI_RUNTIME_BLOB_ARRAY_PTR HI_RUNTIME_SRC_BLOB_ARRAY_PTR;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_BLOB_ARRAY_S](#)

HI_RUNTIME_DST_BLOB_ARRAY_PTR

【说明】

定义输出序列指针。

【定义】

```
typedef HI_RUNTIME_BLOB_ARRAY_PTR HI_RUNTIME_DST_BLOB_ARRAY_PTR;
```

【成员】

无。

【注意事项】



无。

【相关数据类型及接口】

[HI_RUNTIME_BLOB_ARRAY_S](#)

HI_RUNTIME_GROUP_HANDLE

【说明】

定义模型组句柄。

【定义】

```
typedef HI_VOID* HI_RUNTIME_GROUP_HANDLE;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

HI_RUNTIME_GROUP_BLOB_S

【说明】

定义组对应的 BLOB 结构。

【定义】

```
typedef struct hiRUNTIME_GROUP_BLOB_S
{
    HI_RUNTIME_BLOB_PTR pstBlob;
    HI_CHAR acOwnerName[MAX_NAME_LEN+1];
    HI_CHAR acBlobName[MAX_NAME_LEN+1];
} HI_RUNTIME_GROUP_BLOB_S;
```

【成员】

成员名称	描述
pstBlob	Blob 指针



成员名称	描述
acOwnerName	Blob 的属主名称
acBlobName	Blob 的名称

【注意事项】

无。

【相关数据类型及接口】

- [HI_RUNTIME_BLOB_S](#)
- [MAX_NAME_LEN](#)

HI_RUNTIME_GROUP_BLOB_ARRAY_S

【说明】

定义组对应的 BLOB 数组信息。

【定义】

```
typedef struct hiRUNTIME_GROUP_BLOB_ARRAY_S
{
    HI_U32 u32BlobNum;
    HI\_RUNTIME\_GROUP\_BLOB\_S* pstBlobs;
} HI_RUNTIME_GROUP_BLOB_ARRAY_S, *HI_RUNTIME_GROUP_BLOB_ARRAY_PTR;
```

【成员】

成员名称	描述
u32BlobNum	Blob 数量
pstBlobs	Blob 指针，指向 Blob 数组

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_S](#)



HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_S

【说明】

定义组对应的输入 BLOB 数组信息。

【定义】

```
typedef HI\_RUNTIME\_GROUP\_BLOB\_ARRAY\_S HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_S;
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_ARRAY_S](#)

HI_RUNTIME_GROUP_DST_BLOB_ARRAY_S

【说明】

定义组对应的输出 BLOB 数组信息。

【定义】

```
typedef HI\_RUNTIME\_GROUP\_BLOB\_ARRAY\_S HI_RUNTIME_GROUP_DST_BLOB_ARRAY_S;
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_ARRAY_S](#)

HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_PTR

【说明】

定义组对应的输入 BLOB 数组信息指针。

【定义】



```
typedef HI\_RUNTIME\_GROUP\_BLOB\_ARRAY\_S* HI_RUNTIME_GROUP_SRC_BLOB_ARRAY_PTR;
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_ARRAY_S](#)

HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR

【说明】

定义组对应的输出 BLOB 数组信息指针。

【定义】

```
typedef HI\_RUNTIME\_GROUP\_BLOB\_ARRAY\_S* HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR;
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_ARRAY_S](#)

HI_RUNTIME_GROUP_SRC_BLOB_S

【说明】

组输入 BLOB。

【定义】

```
typedef HI\_RUNTIME\_GROUP\_BLOB\_S HI_RUNTIME_GROUP_SRC_BLOB_S;
```

【成员】

无。

【注意事项】



无。

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_S](#)

HI_RUNTIME_GROUP_DST_BLOB_S

【说明】

组输出 BLOB。

【定义】

```
typedef HI\_RUNTIME\_GROUP\_BLOB\_S HI_RUNTIME_GROUP_DST_BLOB_S;
```

【成员】

无。

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_GROUP_BLOB_S](#)

HI_RUNTIME_MEM_CTRL_S

【说明】

内存管理结构体。

【定义】

```
typedef struct hiRUNTIME_MEM_CTRL_S  
{  
    HI_RUNTIME_AllocMem allocMem;  
    HI_RUNTIME_FlushCache flushMem;  
    HI_RUNTIME_FreeMem freeMem;  
} HI_RUNTIME_MEM_CTRL_S;
```

【成员】

成员名称	描述
allocMem	内存分配回调函数



成员名称	描述
flushMem	内存刷新回调函数
freeMem	内存释放回调函数

【注意事项】

无。

【相关数据类型及接口】

HI_RUNTIME_AllocMem、HI_RUNTIME_FlushCache 和 HI_RUNTIME_FreeMem 三个回调函数请参考 hi_runtime_comm.h

HI_RUNTIME_WK_INFO_S

【说明】

WK 模型数据结构体。

【定义】

```
typedef struct hiRUNTIME_WK_MEM_S
{
    HI_CHAR acModelName[MAX_NAME_LEN+1];
    HI_RUNTIME_MEM_S stWKMemory;
} HI_RUNTIME_WK_INFO_S;
```

【成员】

成员名称	描述
acModelName	模型名称
stWKMemory	模型存放的内存信息

【注意事项】

无。

【相关数据类型及接口】

- [HI_RUNTIME_MEM_S](#)



- [MAX_NAME_LEN](#)

HI_RUNTIME_WK_INFO_ARRAY_S

【说明】

定义 WK 模型数组信息。

【定义】

```
typedef struct hiRUNTIME_WK_MEM_ARRAY_S
{
    HI_U32 u32WKNum;
    HI\_RUNTIME\_WK\_INFO\_S *pstAttrs;
} HI_RUNTIME_WK_INFO_ARRAY_S, *HI_RUNTIME_WK_INFO_ARRAY_PTR;
```

【成员】

成员名称	描述
u32WKNum	模型数量
pstAttrs	模型指针，指向模型数组

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_WK_INFO_S](#)

HI_RUNTIME_COP_ATTR_S

【说明】

定义用户自定义层属性信息。

【定义】

```
typedef struct hiRUNTIME_COP_ATTR_S
{
    HI_CHAR acModelName[MAX\_NAME\_LEN+1];
    HI_CHAR acCopName[MAX\_NAME\_LEN+1];
    HI_U32 u32ConstParamSize;
```



```
HI_VOID* pConstParam;  
} HI_RUNTIME_COP_ATTR_S;
```

【成员】

成员名称	描述
acModelName	自定义层所在的模型名称。
acCopName	自定义层层名。
u32ConstParamSize	超参数占用大小。
pConstParam	指向超参数内容的指针。

【注意事项】

无。

【相关数据类型及接口】

[MAX_NAME_LEN](#)

HI_RUNTIME_COP_ATTR_ARRAY_S

【说明】

定义用户自定义层数组信息。

【定义】

```
typedef struct hiRUNTIME_COP_ATTR_ARRAY_S  
{  
    HI_U32 u32CopNum;  
    HI\_RUNTIME\_COP\_ATTR\_S *pstAttrs;  
} HI_RUNTIME_COP_ATTR_ARRAY_S, *HI_RUNTIME_COP_ATTR_ARRAY_PTR;
```

【成员】

成员名称	描述
u32CopNum	自定义层的数量。
pstAttrs	自定义层对应的指针，指向 COP 数组。



【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_COP_ATTR_S](#)

HI_RUNTIME_CONNECTOR_ATTR_S

【说明】

定义 Connector 对象的属性信息。

【定义】

```
typedef struct hiRUNTIME_CONNECTOR_ATTR_S
{
    HI_CHAR acName[MAX_NAME_LEN+1];
    HI\_RUNTIME\_Connector\_Compute pConnectorFun;
    HI_VOID *pParam;
} HI_RUNTIME_CONNECTOR_ATTR_S;
```

【成员】

成员名称	描述
acName	Connector 对象名称
pConnectorFun	Connector 回调函数
pParam	Connector 参数内容指针

【注意事项】

[HI_RUNTIME_Connector_Compute](#) 回调函数请参考 [hi_runtime_comm.h](#)

【相关数据类型及接口】

[MAX_NAME_LEN](#)

HI_RUNTIME_CONNECTOR_ATTR_ARRAY_S

【说明】

定义 Connector 对象数组信息。



【定义】

```
typedef struct hiRUNTIME_CONNECTOR_ATTR_ARRAY_S
{
    HI_U32 u32ConnectorNum;
    HI\_RUNTIME\_CONNECTOR\_ATTR\_S *pstAttrs;
} HI_RUNTIME_CONNECTOR_ATTR_ARRAY_S; *HI_RUNTIME_CONNECTOR_ATTR_ARRAY_PTR;
```

【成员】

成员名称	描述
u32ConnectorNum	Connector 对象个数
pstAttrs	Connector 对象指针，指向 Connector 数组

【注意事项】

无。

【相关数据类型及接口】

[HI_RUNTIME_CONNECTOR_ATTR_S](#)

HI_RUNTIME_GROUP_INFO_S

【说明】

定义 Runtime 组信息。

【定义】

```
typedef struct hiRUNTIME_GROUP_INFO_S
{
    HI\_RUNTIME\_WK\_INFO\_ARRAY\_S stWksInfo;
    HI\_RUNTIME\_COP\_ATTR\_ARRAY\_S stCopsAttr;
    HI\_RUNTIME\_CONNECTOR\_ATTR\_ARRAY\_S stConnectorsAttr;
} HI_RUNTIME_GROUP_INFO_S;
```

【成员】

成员名称	描述
stWksInfo	Wk 对象数组结构体信息



成员名称	描述
stCopsAttr	Cop 对象数组结构体信息
stConnectorsAttr	Connector 对象数组结构体信息

【注意事项】

无。

【相关数据类型及接口】

- [HI_RUNTIME_WK_INFO_ARRAY_S](#)
- [HI_RUNTIME_COP_ATTR_ARRAY_S](#)
- [HI_RUNTIME_CONNECTOR_ATTR_S](#)

HI_RUNTIME_FORWARD_STATUS_CALLBACK_E

【说明】

定义异步 Forward 的状态信息。

【定义】

```
typedef enum hiRUNTIME_FORWARD_STATUS_CALLBACK_E
{
    HI_RUNTIME_FORWARD_STATUS_SUCC = 0x0,
    HI_RUNTIME_FORWARD_STATUS_FAIL,
    HI_RUNTIME_FORWARD_STATUS_ABORT,
    HI_RUNTIME_FORWARD_STATUS_BUTT
} HI_RUNTIME_FORWARD_STATUS_CALLBACK_E;
```

【成员】

成员名称	描述
HI_RUNTIME_FORWARD_STATUS_SUCC	FORWARD 结果返回成功
HI_RUNTIME_FORWARD_STATUS_FAIL	FORWARD 结果返回失败
HI_RUNTIME_FORWARD_STATUS_ABORT	FORWARD 异常终止

【注意事项】



无。

【相关数据类型及接口】

无。

HI_RUNTIME_Forward_Callback

【说明】

定义异步 Forward 的回调函数。

【定义】

```
typedef HI_S32  
(*HI_RUNTIME_Forward_Callback)(HI_RUNTIME_FORWARD_STATUS_CALLBACK_E enEvent,  
HI_RUNTIME_GROUP_HANDLE hGroupHandle, HI_U64 u64SrcId,  
HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR pstDst);
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

- [HI_RUNTIME_FORWARD_STATUS_CALLBACK_E](#)
- [HI_RUNTIME_GROUP_HANDLE](#)
- [HI_RUNTIME_GROUP_DST_BLOB_ARRAY_PTR](#)

HI_RUNTIME_Connector_Compute

【说明】

定义 Connector 的回调函数。

【定义】

```
typedef HI_S32 (*HI_RUNTIME_Connector_Compute)(HI_RUNTIME_SRC_BLOB_ARRAY_S*  
pstConnectorSrc, HI_RUNTIME_DST_BLOB_ARRAY_S* pstConnectorDst, HI_U64 u64SrcId,  
HI_VOID* pParam);
```

【成员】

无



【注意事项】

无

【相关数据类型及接口】

- [HI_RUNTIME_SRC_BLOB_ARRAY_S](#)
- [HI_RUNTIME_DST_BLOB_ARRAY_S](#)

MAX_OPERAND_NAME_LEN

【说明】

定义对插件进行的操作的描述字符串的最大长度。

【定义】

```
#define MAX_OPERAND_NAME_LEN 64
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

无

HI_NodePlugin_Shape_S

【说明】

定义插件输入输出的 HWC 维度的结构体。

【定义】

```
typedef struct hiNodePlugin_Shape_S {  
    HI_S32 s32H;  
    HI_S32 s32W;  
    HI_S32 s32C;  
} HI_NodePlugin_Shape_S;
```

【成员】



成员名称	描述
s32H	插件的 H 维度
s32W	插件的 W 维度
s32C	插件的 C 维度

【注意事项】

无

【相关数据类型及接口】

无

HI_NodePlugin_ElemType_E

【说明】

定义插件数据对齐方式的枚举。

【定义】

```
typedef enum hiNodePlugin_ElemType_E {  
    ELEM_TYPE_U8,  
    ELEM_TYPE_U16,  
    ELEM_TYPE_U32  
} HI_NodePlugin_ElemType_E;
```

【成员】

成员名称	描述
ELEM_TYPE_U8	8Bit 对齐
ELEM_TYPE_U16	16Bit 对齐
ELEM_TYPE_U32	32Bit 对齐

【注意事项】

无



【相关数据类型及接口】

无

HI_NodePlugin_Operand_S

【说明】

定义插件输入输出的数据和属性的结构体。

【定义】

```
typedef struct hiNodePlugin_Operand_S {  
    HI_U64 u64Offset; // addr  
    HI_CHAR mName[MAX_OPERAND_NAME_LEN + 1];  
    HI_NodePlugin_ElemType_E enElemType;  
    HI_U32 u32Num;  
    HI_U32 u32Stride;  
    HI_NodePlugin_Shape_S stShape;  
} HI_NodePlugin_Operand_S
```

【成员】

成员名称	描述
u64Offset	储存着输入和输出的数据的地址
mName	对齐操作的类型的字符串
enElemType	对齐类型
u32Num	表示连续内存块的数目，若一帧数据对应一个块，则表示 blob 中有 u32Num 帧。
u32Stride	位移
stShape	输入输出的 HWC

【注意事项】

无。

【相关数据类型及接口】

- [HI_NodePlugin_ElemType_E](#)



- [HI_NodePlugin_Shape_S](#)

HI_NodePlugin_NodeParam_S

【说明】

定义插件超参数和训练参数的结构体。

【定义】

```
typedef struct HiNodeParam {  
    HI_VOID *pParam;  
    HI_U32 u32Size;  
} HI_NodePlugin_NodeParam_S;
```

【成员】

成员名称	描述
pParam	插件参数的指针
u32Size	插件参数的大小，单位：byte

【注意事项】

pParam 为用自定义的结构体的指针。由于插件的参数是根据自己运行的网络决定的，所以这里实现为泛型指针。u32Size 为该数据类型的大小。

【相关数据类型及接口】

无

3.5 错误码

错误代码	宏定义	描述
0xFF000F01	HI_ERR_SVP_RUNTIME_ILLEGAL_STATE	非法状态
0xFF000F02	HI_ERR_SVP_RUNTIME_MODEL_NOLOAD	模型未加载
0xFF000F03	HI_ERR_SVP_RUNTIME_NULL_PTR	函数参数中有空指针
0xFF000F04	HI_ERR_SVP_RUNTIME_INVALID_PARAM	参数错误



错误代码	宏定义	描述
0xFF000F05	HI_ERR_SVP_RUNTIME_SDK_ERROR	SDK 接口执行错误
0xFF000F06	HI_ERR_SVP_RUNTIME_SDK_NOMEM	分配内存失败，如系统内存不足

3.6 Proc 调试信息

3.6.1 概述

调试信息采用了 Linux 下的 proc 文件系统，可实时反映当前系统 Runtime 的运行状态，所记录的信息可供问题定位及分析时使用。

【文件路径】

/proc/hisi/svp/rt/task

【信息查看方法】

- 在控制台上可以使用 cat 命令查看信息，cat /proc/hisi/svp/rt/task；也可以使用其他常用的文件操作命令，例如 cp /proc/hisi/svp/rt/task ./，将文件拷贝到当前目录。
- 在应用程序中可以将上述文件当作普通只读文件进行读操作，例如 fopen、fread 等。

3.6.2 Proc 信息说明

【调试信息】

```
cat /proc/hisi/svp/rt/task
```

```
Left Seg Num Info
```

```
=====
```

```
Left Unready Seg Num: 6
```

```
Left Ready Seg Num On NNIE: 0
```

```
Left Ready Seg Num On CPU: 0
```

```
NNIE_0 Cost Time / Total Cost Time / Use Rate
```

```
=====
```



```

1802111          /3654734          /49%

NNIE_1 Cost Time / Total Cost Time / Use Rate
=====
637106          /3693870          /17%
GroupName      Frameld ForwardTime  EnterPriQTime  EnterExecQTime  ExecStartTime
ExecEndTime TotalTime   SegInfo
=====rfcn
_alexnet      0(24)  364099995  585   25   10      355993  356614
rfcn(Vop0)(171377, NNIE_0)
rfcn(proposal)(73911, CPU_2)
rfcn(Vop2)(11764, NNIE_1)
rfcn(Vop3)(8016, NNIE_0)
rfcn_conn_alexnet(rfcn_conn_alexnet)(4315, CPU_2)
alexnet(Vop0)(47226, NNIE_1)

```

【调试信息分析】

记录当前 NNIE 工作状态资源信息，主要包括 Runtime 队列状态信息，任务状态信息，运行时状态信息等。

【参数说明】

参数		描述
Left Seg Num Info	Left Unready Seg Num	队列中数据未准备好的分段单元
	Left Ready Seg Num on NNIE	队列中数据已准备好的 NNIE 分段单元
	Left Ready Seg Num on CPU	队列中数据已准备好的 CPU 分段单元
NNIE INFO	Cost Time	NNIE 上执行数据总时间
	Total Cost Time	NNIE 总消耗时间，包含空闲时间和 NNIE 上执行数据总时间
	Use Rate	NNIE 的利用率
Group Info	Group Name	模型组名称
	Frameld	外部输入帧 ID，()内为内部记录的帧 ID



参数		描述
	ForwardTime	调用 Forward 的时间, utc 时间, 单位 μs
	EnterPriQTime	进入优先级队列的时间, 相对 ForwardTime 的时间, 单位 μs
	EnterExecQTime	进入执行队列的时间, 相对 EnterPriQTime 的时间, 单位 μs
	ExecStartTime	第一个分段执行的时间, 相对 EnterExecQTime 的时间, 单位 μs
	ExecEndTime	模型组执行完的时间, 相对 ExecStartTime 的时间, 单位 μs
	TotalTime	从 Forward 到结束总的耗时, 单位 μs
	SegInfo	模型组的各分段执行情况, 包含如下部分: 分段执行时间、在哪个设备上执行, 时间单位 μs