



HiSVP

API 参考

文档版本 00B01
发布日期 2018-01-10

版权所有 © 深圳市海思半导体有限公司 2018。保留一切权利。

非经本公司书面许可，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深圳市海思半导体有限公司的商标。

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。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前 言

概述

本文档为使用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 与 Hi3559AV100 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


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	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。



符号	说明
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

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
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
1.3 API 参考	1
1.4 数据类型和数据结构.....	6
1.5 错误码.....	11
1.6 Proc 调试信息	12
1.6.1 概述	12
1.6.2 Proc 信息说明	12
2 NNIE.....	16
2.1 概述.....	16
2.2 功能描述.....	16
2.2.1 重要概念	16
2.2.2 使用示意	23
2.3 API 参考	24
2.4 数据类型和数据结构.....	34
2.5 错误码.....	50
2.6 Proc 调试信息	51
2.6.1 概述	51
2.6.2 Proc 信息说明	52



插图目录

图 2-1 跨度 (stride) 示意图.....	17
图 2-2 SVP_BLOB_TYPE_S32 类型 SVP_BLOB_S (2 通道 2 帧示意图)	19
图 2-3 SVP_BLOB_TYPE_U8 类型 SVP_BLOB_S (3 通道 2 帧示意图)	20
图 2-4 SVP_BLOB_TYPE_YVU420SP 类型 SVP_BLOB_S (2 帧 YVU420SP 示意图)	21
图 2-5 SVP_BLOB_TYPE_YVU422SP 类型 SVP_BLOB_S (2 帧 YVU422SP 示意图)	22
图 2-6 SVP_BLOB_TYPE_VEC_S32 类型 SVP_BLOB_S (2 帧示意图)	22
图 2-7 SVP_BLOB_TYPE_SEQ_S32 类型 SVP_BLOB_S(Num=N 帧示意图)	23
图 2-8 SVP_MEM_INFO_S 类型的数据内存示意	23
图 2-9 CNN_Forward 支持的多节点输入输出网络示意图	28
图 2-10 CNN_ForwardWithBbox 支持的输入输出网络示意图	30
图 2-11 CNN_ForwardWithBbox astBbox[i]输入示意图	31
图 2-12 CNN_ForwardWithBbox Score 输出示意图	31
图 2-13 CNN_ForwardWithBbox Bbox 调整值输出示意图 1	31
图 2-14 CNN_ForwardWithBbox Bbox 调整值输出示意图 2	32



表目录

表 1-1 DSP 模块 API 错误码.....	11
表 2-1 BLOB 内存排布类型表.....	18
表 2-2 NNIE 引擎 API 错误码.....	50



1 DSP

1.1 概述

SVP (Smart Vision Processing) 平台是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。

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。

1.3 API 参考

海思 DSP ARM 端模块 API 接口操作。

提供以下 API:

- [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_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】

无。

【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 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_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, SVP_DSP_MESSAGE_S  
*pstMsg, SVP_DSP_ID_E enDspId, SVP_DSP_ID_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
enDspId	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 enDspId, SVP_DSP_HANDLE  
hHandle, HI_BOOL *pbFinish, HI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
enDspId	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。



返回值	描述
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。

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_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_ID_E

【说明】

定义 DSP ID。

【定义】

```
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;
```



【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】

```
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  = 0x10,
    SVP_DSP_MEM_TYPE_DRAM_1_DSP_2  = 0x11,

    SVP_DSP_MEM_TYPE_SYS_DDR_DSP_3 = 0x12,
    SVP_DSP_MEM_TYPE_IRAM_DSP_3    = 0x13,
    SVP_DSP_MEM_TYPE_DRAM_0_DSP_3  = 0x14,
    SVP_DSP_MEM_TYPE_DRAM_1_DSP_3  = 0x15,

    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。
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。



成员名称	描述
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。



成员名称	描述
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。



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
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	配置错误



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

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[Nov 29 2017, 17:45:26]
```

```
-----MODULE PARAM-----
```

```
max_node_num
          30
```

```
-----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 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。
	RunTm	DSP 运行总时间。 单位：s。
DSP INVOKE	CoreId	DSP 核 ID。



参数		描述
INFO 调用信息	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。

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

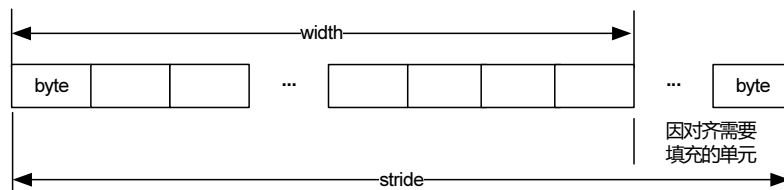
《HiMPP Vx.y 媒体处理软件开发参考》), 将数据从 cache 刷到 DDR, 以供 IVE 使用。

- 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



注意

1. Hi3559AV100 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 字节的奇数倍对齐。
2. 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 2-4~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。 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 2-6 所示。



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

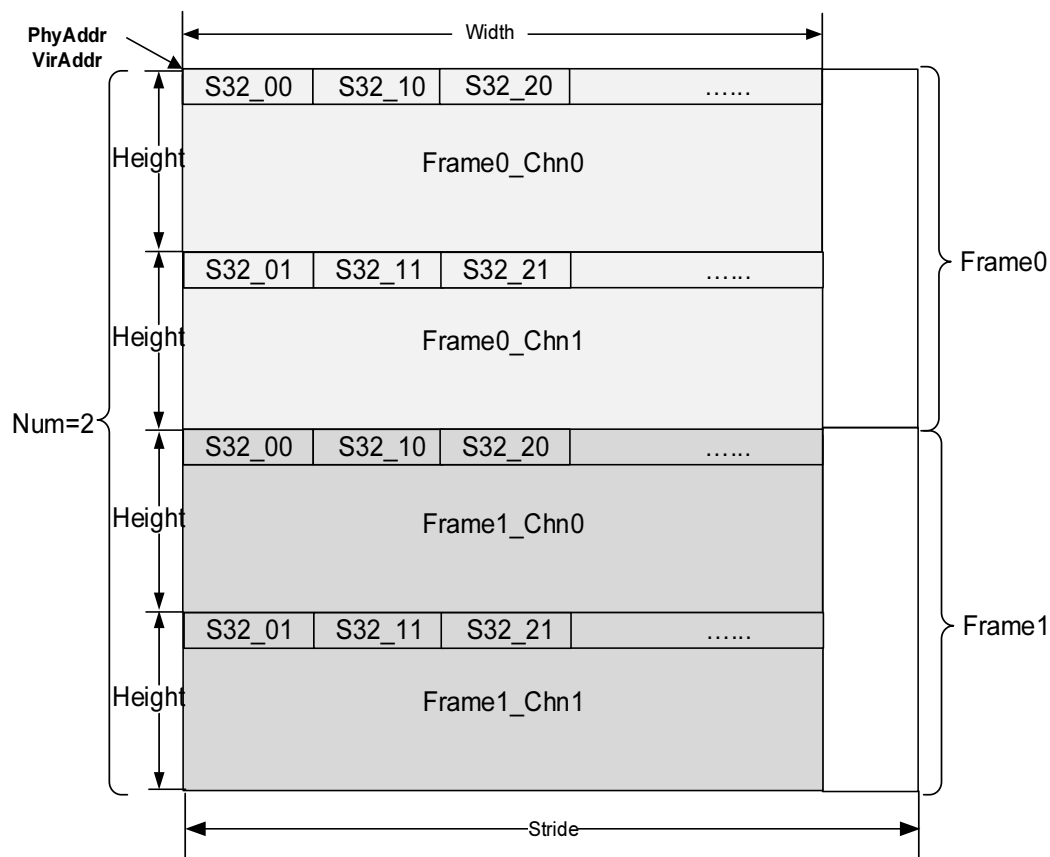
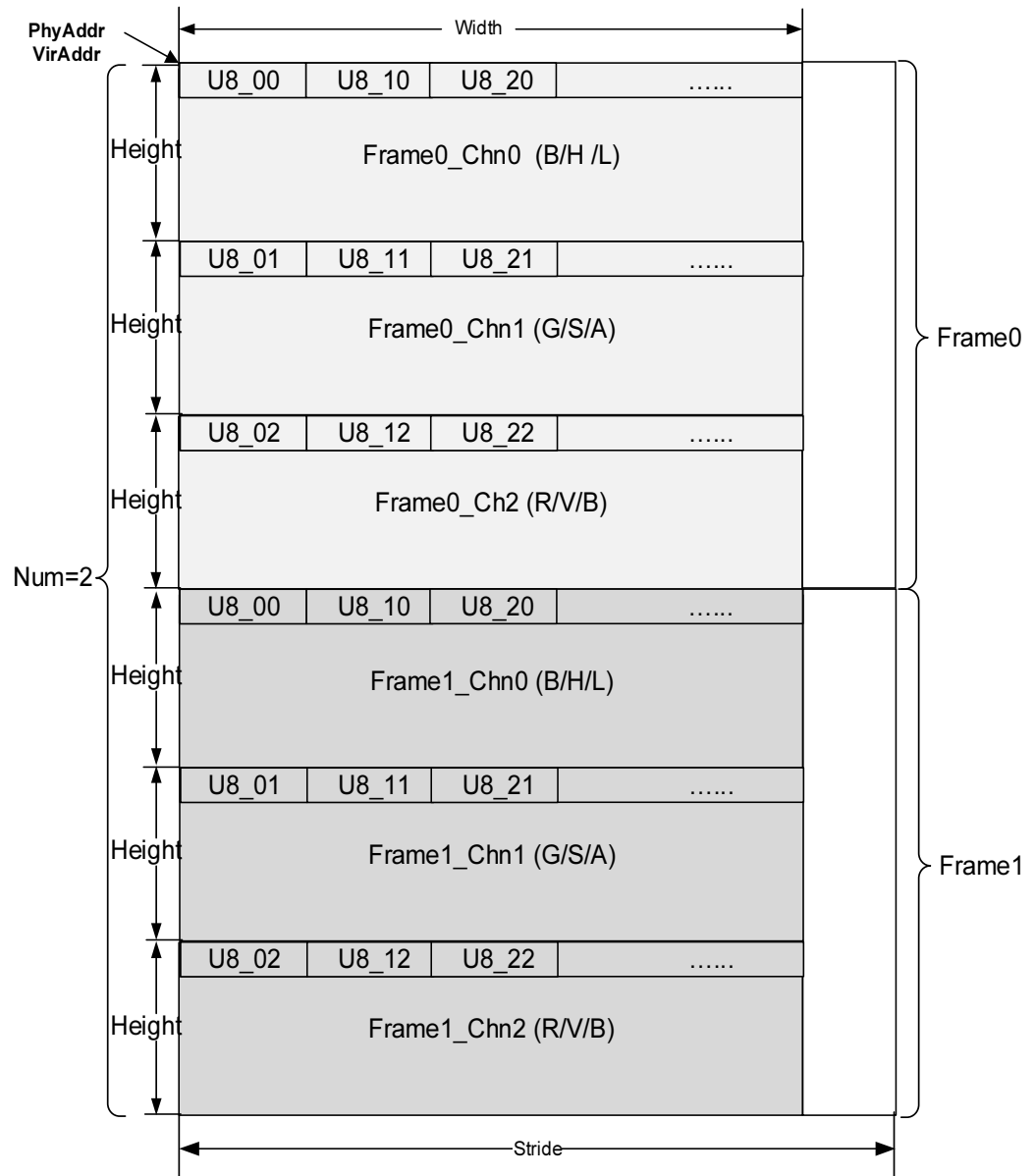


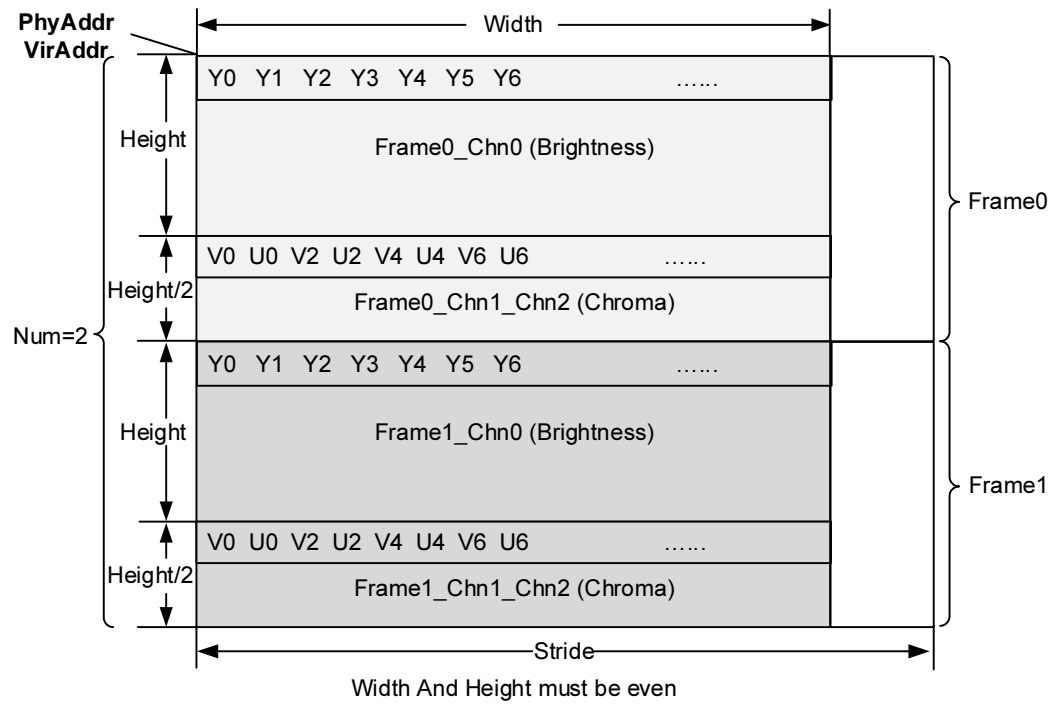
图2-3 SVP_BLOB_TYPE_U8 类型 SVP_BLOB_S (3 通道 2 帧示意图)



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。



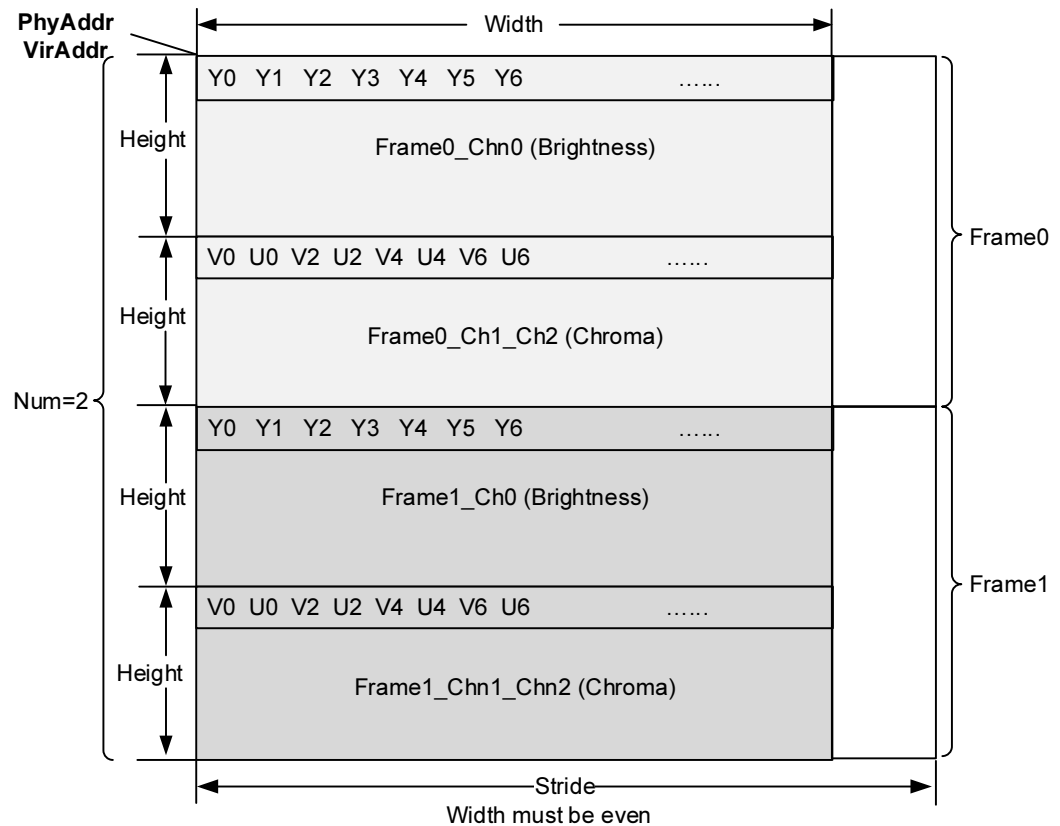
图2-4 SVP_BLOB_TYPE_YVU420SP 类型 SVP_BLOB_S (2 帧 YVU420SP 示意图)



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)

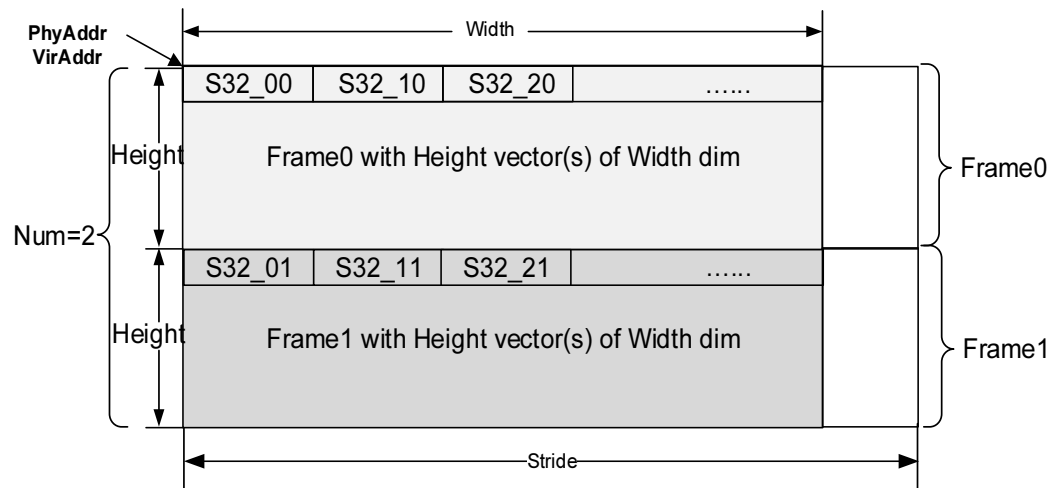




图2-7 SVP_BLOB_TYPE_SEQ_S32 类型 SVP_BLOB_S(Num=N 帧示意图)

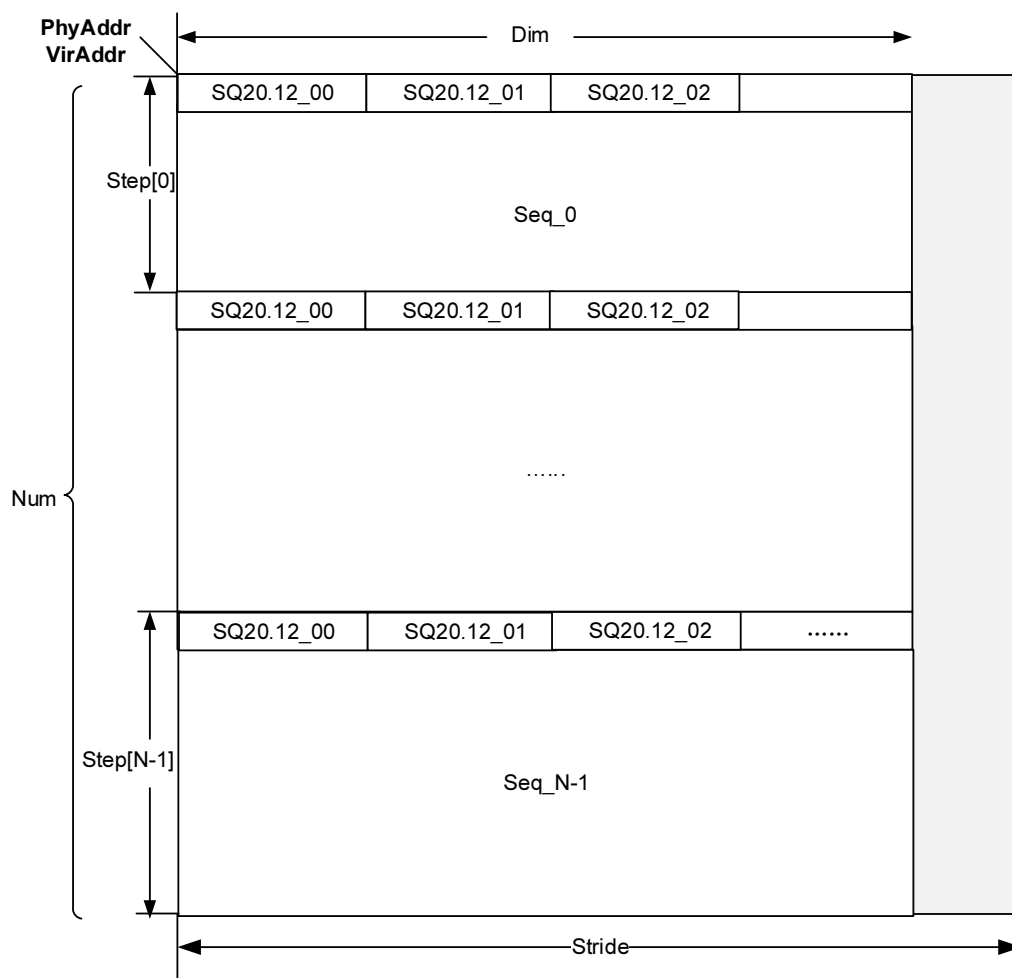
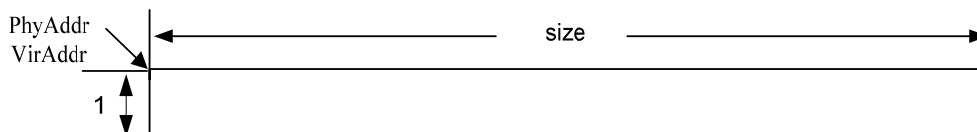


图2-8 SVP_MEM_INFO_S 类型的数据内存示意



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 `bInstant` 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_CNN_Forward](#): 多节点输入输出的 CNN 网络预测。
- [HI_MPI_SVP_NNIE_CNN_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_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 (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 hi_nnie1.1.lib）

【注意】



无。

【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, 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。	输入
u32MaxBboxNum	网络中有 RPN 层时输出的最大候选 Bounding box 个数。	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。	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 hi_nnie1.1.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_CNN_Forward

【描述】

多节点输入输出的 CNN 网络预测。

【语法】

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

【参数】

参数名称	描述	输入/输出
<code>phSvpNnieHandle</code>	handle 指针。 不能为空。	输出
<code>astSrc[]</code>	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
<code>pstModel</code>	网络模型结构体。	输入
<code>astDst[]</code>	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
<code>pstForwardCtrl</code>	控制结构体。	输入
<code>bInstant</code>	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	8x8~4096x4096 向量维度: 1~0xFFFFFFFF
pstModel	网络段类型支持 CNN\DNN\Curren t	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	无
astDst[]	S32\ VER_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 。

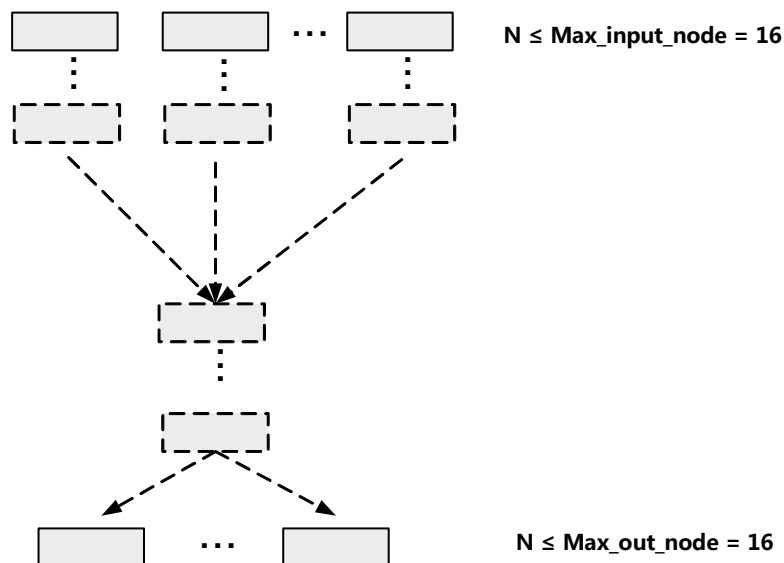
【需求】

- 头文件: hi_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件: libnnie.a (PC 上模拟用 hi_nnie1.1.lib)

【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2-9 CNN_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_CNN_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，网络对输入的目标框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,
SVP_SRC_BLOB_S astSrc[], SVP_SRC_BLOB_S astBbox[], SVP_NNIE_MODEL_S
*pstModel, SVP_DST_BLOB_S astDst[], 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 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。	输出
pstForwardCtrl	控制结构体。	输入
bInstant	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	8x8~4096x4096,
astBbox[]	S32	DDR3 16 byte DDR4 32 byte 追求高性能建议 256 byte	Width=4, Height≤5000
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	32x32~4096x4096 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_comm_svp.h、hi_nnie.h、mpi_nnie.h
- 库文件：libnnie.a（PC 上模拟用 hi_nnie1.1.lib）

【注意】

- 当前 astBbox 数组的元素个数仅支持 1，即 pstForwardCtrl→u32ProposalNum=1；
- astBbox 中的坐标都采用 SQ20.12 的定点输入；
- 根据训练时的设定，输出的 Bbox 坐标调整值 Bbox_Delta，大致可分为 2 种情况：
 - 一种是每一类别单独享有自己的 Bbox_Delta，则每一个 Bbox_Delta 的输出维度为 class_num * 4，参考[图 2-7](#)；
 - 另一种是各类别共享一组 Bbox_Delta，则每个 Bbox_Delta 的输出维度为 4，参考[图 2-8](#)；

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

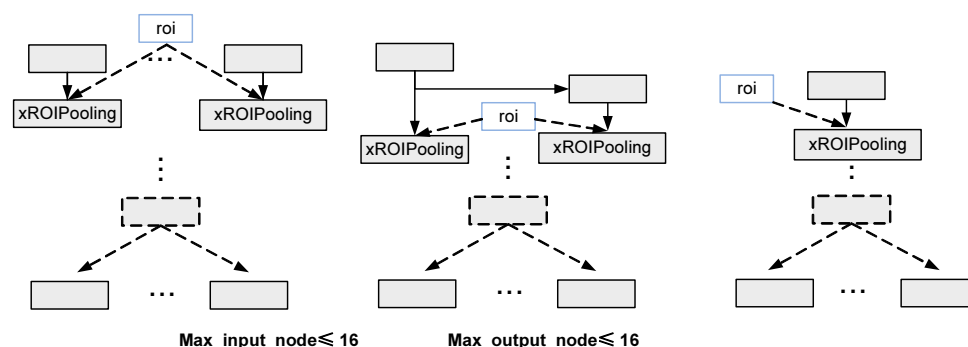


图2-11 CNN_ForwardWithBbox astBbox[i]输入示意图

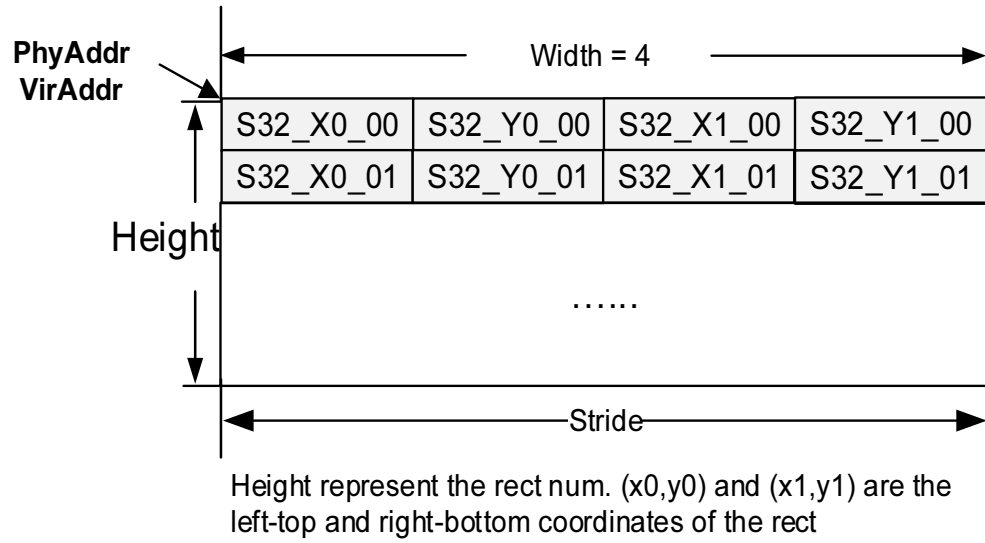


图2-12 CNN_ForwardWithBbox Score 输出示意图

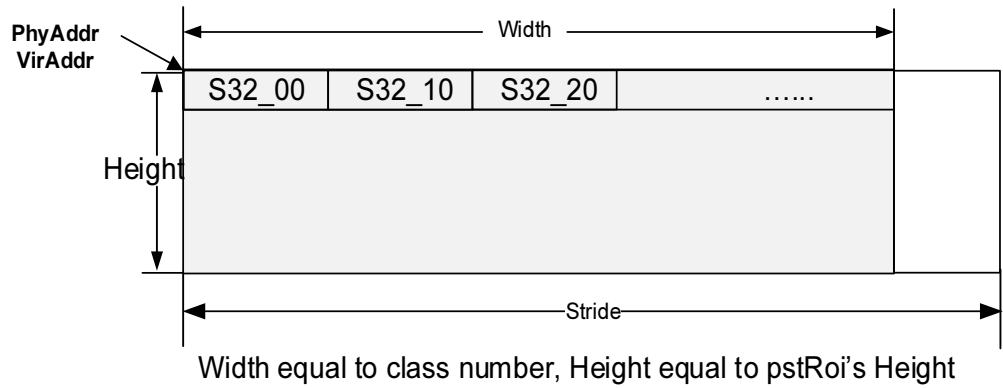


图2-13 CNN_ForwardWithBbox Bbox 调整值输出示意图 1

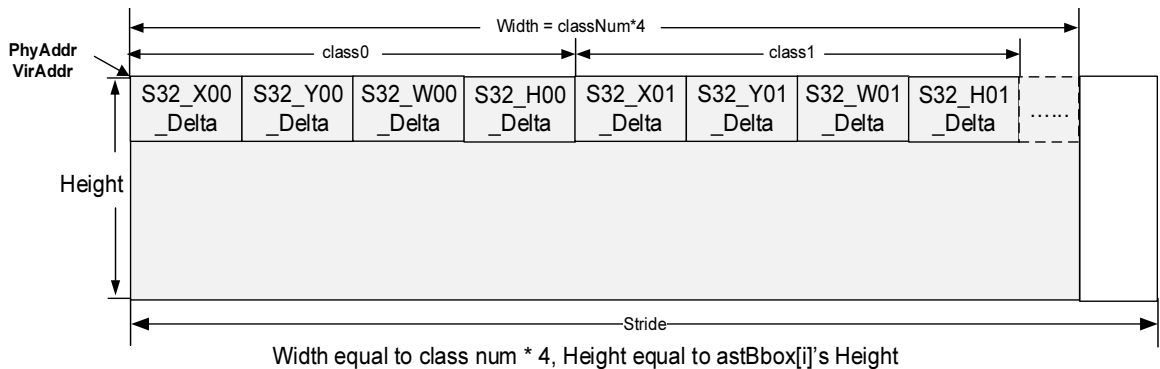
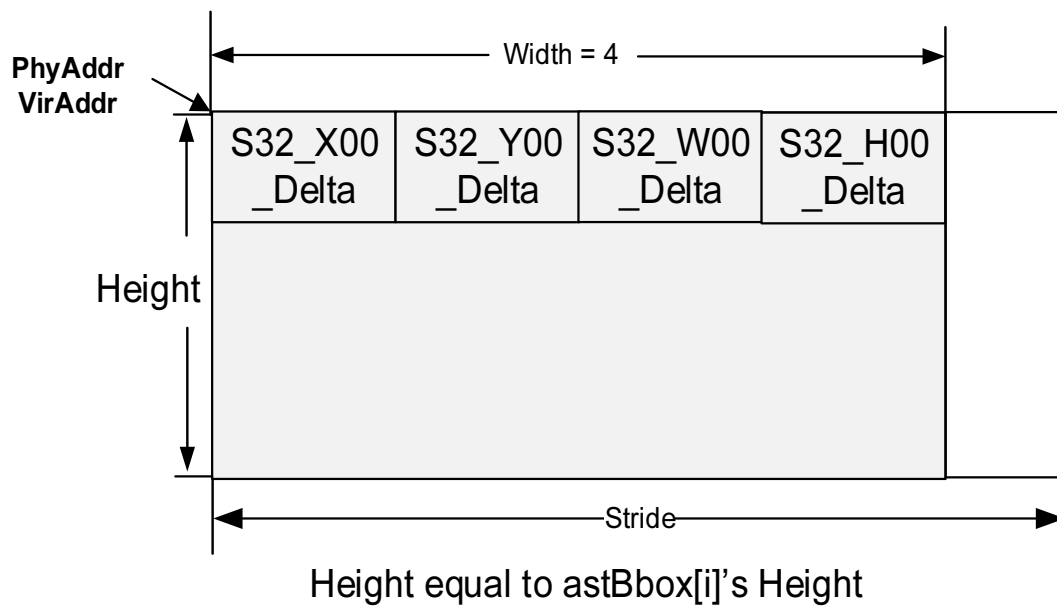




图2-14 CNN_ForwardWithBbox Bbox 调整值输出示意图 2



【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。



返回值	描述
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 hi_nnie1.1.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 hi_nnie1.1.lib）

【注意】

无。

【举例】

无。

【相关主题】

无。

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。
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。



成员名称	描述
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: vecotr 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。
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。



成员名称	描述
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】

[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】

```
typedef enum hiSVP_NNIE_ID_E
{
    SVP_NNIE_ID_0      = 0x0,
    SVP_NNIE_ID_1      = 0x1,
    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。



成员名称	描述
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 simultaion */
    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。
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 u32NodeId;
}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。
u32Height	节点内存形状的高。
u32Chn	节点内存形状的通道数。
u32Dim	节点内存的向量维度。
u32NodeId	节点在网络中的 Id。

【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                u16LayerNum;
    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                au32RoiIdx[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。
u16DstNum	网络段的输出节点数。
u16RoiPoolNum	网络段中包含的 RoiPooling 以及 PSRoiPooling layer 数。
u16LayerNum	网络段中的 layer 数。
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。
au32RoiIdx[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】

无。

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。
u32NetSegNum	网络模型中 NNIE 执行的网络分段数。
astRoiInfo	网络模型中 RoiPooling 以及 PsRoiPooling 的信息数组。
stBase	网路在 NNIE 引擎上执行的段数

【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   enNnieId;      /* 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。
u32DstNum	NNIE 执行网络段输出节点个数。
u32NetSegId	网络段的段序号。
enNnieId	执行网络段的 NNIE 引擎 ID。
stTmpBuf	辅助内存。
stTskBuf	辅助内存。

【注意事项】

无。

【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; /* elment num of roi array */
}
```




```

HI_U32          u32NetSegId;    /* net segment index running on NNIE
*/
SVP_NNIE_ID_E   enNnieId;      /* 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。
u32DstNum	NNIE 执行网络段输出节点个数。
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。
u32NetSegId	网络段的段序号。
enNnieId	执行网络段的 NNIE 引擎 ID。
stTmpBuf	辅助内存。
stTskBuf	辅助内存。

【注意事项】

无。

【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



错误代码	宏定义	描述
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
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	缓冲区中无图像
0xA033800F	HI_ERR_SVP_NNIE_BUF_FULL	缓冲区中图像满
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_OPEN_FILE	打开文件失败
0xA0338043	HI_ERR_SVP_NNIE_READ_FILE	读文件失败
0xA0338044	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 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

```
----- 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	RunTm
0	0	0	0
0	0	0	0



```

-----NNIE INVOKE INFO-----
CoreId  Forward  ForwardWithRoi
    0      0      0
    1      0      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 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。
	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。
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 bInstant 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。



参数		描述
	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。
	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。
	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。
	ForwardWithRoi	NNIE ForwardWithRoi 调用次数