AX Joint 使用手册

文档版本：V1.5

发布日期：2022/01/11

目 录

权利声明

注意

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

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

前 言

适用产品

爱芯 AX620A

适读人群

- 终端用户
- 售前
- 售后
- 技术人员

符号与格式定义

符号/格式	说明
`xxx`	表示您可以执行的命令行。
斜体	表示变量。如，“*安装目录*/AX620A_SDK_Vx.x.x/build 目录”中的“安装目录”是一个变量，由您的实际环境决定。
☞ 说明/备注：	表示您在使用产品的过程中，我们向您说明的事项。
！ 注意：	表示您在使用产品的过程中，需要您特别注意的事项。

修订历史

文档版本	发布时间	修订说明
V1.0	2020/01/01	文档初版
V1.1	2021/04/30	更新错误码信息
V1.2	2021/05/06	更新接口函数，可获取当前 NPU 的模式；
V1.3	2021/07/07	更新文档模板
V1.4	2021/07/14	为 API 中体现的结构体增加跳转链接
V1.5	2022/01/11	更新接口函数级结构体函数

1 Functions 函数

AX_JOINT_Adv_Init

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

初始化 Run Joint 环境。

```
AX_S32 AX_JOINT_Adv_Init(AX_JOINT_SDK_ATTR_T* pAttr)
```

【Parameters】

pAttr：Run Joint 属性配置结构体。

【Return】

运行结果错误码。

AX_JOINT_Adv_Deinit

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

反初始化 Run Joint 环境。

```
AX_S32 AX_JOINT_Adv_Deinit(AX_VOID)
```

【Parameters】

无

【Return】

运行结果错误码。

AX_JOINT_CreateHandle

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

创建 Run Joint 句柄。

```
AX_S32 AX_JOINT_CreateHandle(AX_JOINT_HANDLE *pHandle, const AX_VOID *pData,
AX_U32 nDataSize)
```

【Parameters】

- pHandle：句柄指针。
- pData：Joint Model 数据。
- nDataSize：Joint Model 数据大小（Byte）。

【Return】

运行结果错误码。

AX_JOINT_DestroyHandle

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

销毁 Run Joint 句柄。

```
AX_S32 AX_JOINT_DestroyHandle(AX_JOINT_HANDLE handle)
```

【Parameters】

handle：待销毁的句柄。

【Return】

运行结果错误码。

AX_JOINT_GetIOInfo

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

获取 Joint Model IO 信息数组首地址。

```
const AX_JOINT_IO_INFO_T* AX_JOINT_GetIOInfo(AX_JOINT_HANDLE handle)
```

【Parameters】

handle：Joint 句柄。

【Return】

Joint Model IO 信息。

AX_JOINT_CreateExecutionContext

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

创建 Joint 运行上下文。

```
AX_S32 AX_JOINT_CreateExecutionContext(AX_JOINT_HANDLE handle,
AX_JOINT_EXECUTION_CONTEXT* pContext)
```

【Parameters】

- handle：Joint 句柄。
- pContext：上下文指针。

【Return】

运行结果错误码。

AX_JOINT_CreateExecutionContextV2

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

使用用户配置参数创建 Joint 运行上下文。

```
AX_S32 AX_JOINT_CreateExecutionContextV2(AX_JOINT_HANDLE handle,
        AX_JOINT_EXECUTION_CONTEXT* pContext,
        AX_JOINT_EXECUTION_CONTEXT_SETTING_T* pSetting)
```

【Parameters】

- handle：Joint 句柄。
- pContext：上下文指针。
- pSetting：上下文参数指针。

【Return】

运行结果错误码。

AX_JOINT_DestroyExecutionContext

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

销毁 Joint 运行上下文。

```
AX_S32 AX_JOINT_DestroyExecutionContext(AX_JOINT_EXECUTION_CONTEXT context)
```

【Parameters】

context：待销毁上下文指针。

【Return】

运行结果错误码。

AX_JOINT_RunSync

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

执行推理操作。

```
AX_S32 AX_JOINT_RunSync(AX_JOINT_HANDLE handle, AX_JOINT_EXECUTION_CONTEXT context, AX_JOINT_IO_T* pIO)
```

【Parameters】

- handle：Joint 句柄。
- context：执行上下文。
- pIO：IO Buffer 结构体。

【Return】

运行结果错误码。

AX_JOINT_AllocBuffer

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

通过 IO Meta 信息自动分配对应 IO Buffer，用于封装底层细节，用户也可自行调用 SYS 相关接口完成内存分配。

```
AX_S32 AX_JOINT_AllocBuffer(const AX_JOINT_IOMETA_T* pMeta,
AX_JOINT_IO_BUFFER_T* pBuf, AX_JOINT_ALLOC_BUFFER_STRATEGY_T eStrategy)
```

【Parameters】

- pMeta：IO Meta 信息指针。
- pBuf：IO Buffer 指针。
- eStrategy：分配策略。

【Return】

运行结果错误码。

！ 注意：

A. 用户自行调用 SYS 相关接口进行内存分配时，如启用 cache 策略，则在调用 AX_JOINT_RunSync 函数之前必须调用 AXSYSMemFlushCache 接口刷新缓冲区，在调用 AX_JOINT_RunSync 函数之后必须调用 AXSYSMemInvalidateCache 函数刷新缓冲区。

B. 用户自行调用 SYS 相关接口进行内存分配时，是否启用 cache 策略不影响推理性能。

AX_JOINT_FreeBuffer

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

释放 IO Buffer。

```
AX_S32 AX_JOINT_FreeBuffer(AX_JOINT_IO_BUFFER_T* pBuf)
```

【Parameters】

pBuf：IO Buffer 指针。

【Return】

运行结果错误码。

AX_JOINT_GetJointModelType

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

通过 Joint Model 数据获取 VNPU 模式。

```
AX_S32 AX_JOINT_GetJointModelType(const AX_S8* pJoint, AX_U32 nJointSize,
AX_NPU_SDK_EX_MODEL_TYPE_T* pModelType)
```

【Parameters】

- pJoint：Joint Model 数据指针。
- pJoint：Joint Model 数据大小。
- pJoint：VNPU 模式结构体指针。

【Return】

运行结果错误码。

AX_JOINT_GetVNPUMode

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

通过 Joint 句柄获取 VNPU 模式。

```
AX_S32 AX_JOINT_GetVNPUMode(AX_JOINT_HANDLE handle,
AX_NPU_SDK_EX_MODEL_TYPE_T *pModelType)
```

【Parameters】

- handle：Joint 句柄。
- pJoint：VNPU 模式结构体指针。

【Return】

运行结果错误码。

AX_JOINT_ADV_GetComponents

- Declared in File joint_adv.h
- Defined in File joint.cpp

【函数功能描述】

通过 Joint 执行上下文获取 Joint Model 子图信息。

```
AX_S32 AX_JOINT_ADV_GetComponents(AX_JOINT_EXECUTION_CONTEXT context,
AX_JOINT_COMPONENT_T **pComponents, AX_U32 *nSize)
```

【Parameters】

- context：Joint 执行上下文。
- pComponents：Joint Model 子图结构体数组首地址指针。
- nSize：Joint Model 子图结构体数组大小。

【Return】

运行结果错误码。

AX_JOINT_ShortcutRun

- Declared in File joint_shortcunt.h
- Defined in File joint_shortcut.cpp

【函数功能描述】

快捷创建 Joint Model 运行环境并执行模型推理。

```
AX_S32 AX_JOINT_ShortcutRun(const AX_VOID* pJoint, AX_U32 nJointSize,
    const AX_VOID* pInput, AX_U32 nInputSize, AX_JOINT_IOMETA_T* pOutputInfos,
AX_JOINT_IO_BUFFER_T** pOutputs, AX_U32* pOutputSize)
```

【Parameters】

- pJoint：Joint Model 数据指针。
- nJointSize：Joint Model 数据大小。
- pInput：输入数据指针。
- nInputSize：输入数据大小。
- pOutputInfos：输出 IO Meta 信息数组首地址指针。
- pOutputs：输出 IO Buffer 数组首地址指针。
- pOutputSize：输出数据大小指针。

【Return】

运行结果错误码。

AX_JOINT_DestroyShortcutRunOutput

- Declared in File joint_shortcunt.h
- Defined in File joint_shortcut.cpp

【函数功能描述】

销毁 AX_JOINT_ShortcutRun 函数输出 IO Meta 信息及 IO Buffer。

```
AX_VOID AX_JOINT_DestroyShortcutRunOutput(AX_JOINT_IOMETA_T* pOutputInfos,
AX_JOINT_IO_BUFFER_T* pOutput, AX_U32 nOutputSize)
```

【Parameters】

- pOutputInfos：输出 IO Meta 信息数组地址。
- pOutputs：输出 IO Buffer 数组地址。
- nOutputSize：输出数据大小，应等于 pOutputInfos 及 pOutputs 数组大小。

【Return】

运行结果错误码。

AX_JOINT_GetVNPUHardMode

- Declared in File joint.h
- Defined in File joint.cpp

【函数功能描述】

获取当前 NPU 的模式。

```
AX_S32 AX_JOINT_GetVNPUHardMode(AX_NPU_SDK_EX_HARD_MODE_T *pHardMode);
```

【Parameters】

pHardMode：输出 NPU 的模式。

【Return】

运行结果错误码（如果 NPU 初始化过则返回 0，未初始化则返回非 0）。

2 Enums 枚举类

AX_JOINT_TENSOR_LAYOUT_T

【定义 Tensor Layout 形式】

- JOINT_TENSOR_LAYOUT_UNKNOWN：未知 Layout。
- JOINT_TENSOR_LAYOUT_NHWC：NHWC 排布。
- JOINT_TENSOR_LAYOUT_NCHW：NCHW 排布。

【Values】

- OINT_TENSOR_LAYOUT_UNKNOWN = 0
- JOINT_TENSOR_LAYOUT_NHWC = 1
- JOINT_TENSOR_LAYOUT_NCHW = 2

AX_JOINT_MEMORY_TYPE_T

【定义 Joint Buffer 内存形式】

- AX_JOINT_MT_INVALID：无效形式。
- AX_JOINT_MT_PHYSICAL：物理连续内存，必须使用 AX_SYS_Mem 接口分配/释放内存空间。
- AX_JOINT_MT_VIRTUAL：虚拟内存。

【Values】

- AX_JOINT_MT_INVALID = 0
- AX_JOINT_MT_PHYSICAL = 1
- AX_JOINT_MT_VIRTUAL = 2

AX_JOINT_DATA_TYPE_T

【定义 Joint Buffer Data 数据类型】

- AX_JOINT_DT_UNKNOWN
- AX_JOINT_DT_UINT8
- AX_JOINT_DT_UINT16
- AX_JOINT_DT_FLOAT32
- AX_JOINT_DT_SINT16
- AX_JOINT_DT_SINT8
- AX_JOINT_DT_SINT32
- AX_JOINT_DT_UINT32
- AX_JOINT_DT_FLOAT64

【Values】

- AX_JOINT_DT_UNKNOWN = 0
- AX_JOINT_DT_UINT8 = 1
- AX_JOINT_DT_UINT16 = 2
- AX_JOINT_DT_FLOAT32 = 3
- AX_JOINT_DT_SINT16 = 4
- AX_JOINT_DT_SINT8 = 5
- AX_JOINT_DT_SINT32 = 6
- AX_JOINT_DT_UINT32 = 7
- AX_JOINT_DT_FLOAT64 = 8

AX_JOINT_COLOR_SPACE_T

【定义 Joint Model 输入数据色彩空间】

- AX_JOINT_CSFEATUREMAP：输入为 Tensor 数据。
- AX_JOINT_CS_RAW8
- AX_JOINT_CS_RAW10
- AX_JOINT_CS_RAW12
- AX_JOINT_CS_RAW14
- AX_JOINT_CS_RAW16
- AX_JOINT_CS_NV12
- AX_JOINT_CS_NV21
- AX_JOINT_CS_RGB
- AX_JOINT_CS_BGR
- AX_JOINT_CS_RGBA
- AX_JOINT_CS_GRAY
- AX_JOINT_CS_YUV444

【Values】

- AX_JOINT_CS_FEATUREMAP = 0
- AX_JOINT_CS_RAW8 = 12
- AX_JOINT_CS_RAW10 = 1
- AX_JOINT_CS_RAW12 = 2
- AX_JOINT_CS_RAW14 = 11
- AX_JOINT_CS_RAW16 = 3

- AX_JOINT_CS_NV12 = 4
- AX_JOINT_CS_NV21 = 5
- AX_JOINT_CS_RGB = 6
- AX_JOINT_CS_BGR = 7
- AX_JOINT_CS_RGBA = 8
- AX_JOINT_CS_GRAY = 9
- AX_JOINT_CS_YUV444 = 10

AX_JOINT_COMPONENT_TYPE_T

【定义 Joint Model 内部子图类型】

- AX_JOINT_COMPONENT_TYPE_UNKNOWN：未知类型。
- AX_JOINT_COMPONENT_TYPE_NEU：neu 子图。
- AX_JOINT_COMPONENT_TYPE_ONNX：onnx 子图。
- AX_JOINT_COMPONENT_TYPE_MAGMA：magma 子图。
- AX_JOINT_COMPONENT_TYPE_AXE：AXEngine 子图。

【Values】

- AX_JOINT_COMPONENT_TYPE_UNKNOWN = 0
- AX_JOINT_COMPONENT_TYPE_NEU = 1
- AX_JOINT_COMPONENT_TYPE_ONNX = 2
- AX_JOINT_COMPONENT_TYPE_MAGMA = 3
- AX_JOINT_COMPONENT_TYPE_AXE = 4

AX_JOINT_ALLOC_BUFFER_STRATEGY_T

【定义 Joint Model IO Buffer 分配策略】

- AX_JOINT_ABST_DEFAULT：使用默认策略。
- AX_JOINT_ABST_CACHED：使用缓存策略。

【Values】

- AX_JOINT_ABST_DEFAULT = 0
- AX_JOINT_ABST_CACHED = 1

3 Structs 结构体

AX_JOINT_IOMETA_EX_T

存储 Joint Model 附加 IO Meta 信息。

```
typedef struct _AX_JOINT_IOMETA_EX_T {
    AX_JOINT_COLOR_SPACE_T eColorSpace;
    AX_JOINT_RTV_TYPE_T eRtvType;
    AX_U64 _reserved[7];
} AX_JOINT_IOMETA_EX_T;
```

eColorSpace：Joint Model 输入数据色彩空间。

eRtvType：RuntimeVar 类型，RuntimeVar 属于 NPU 硬件特有机制，请详询技术支持。

_reserved：保留字段。

AX_JOINT_SDK_ATTR_T

存储 Joint Model 属性信息。

```
typedef struct _AX_JOINT_SDK_ATTR_T {
    AX_NPU_SDK_EX_HARD_MODE_T eNpuMode;
    AX_U64 _reserved[7];
} AX_JOINT_SDK_ATTR_T;
```

eNpuMode：Joint Model VNPU 模式。

_reserved：保留字段。

AX_JOINT_IOMETA_T

存储 Joint Model 内部节点数据 IO Meta 信息。

```
typedef struct _AX_JOINT_IOMETA_T {
    AX_S8 *pName;
    AX_S32 *pShape; // YUV will be treated as 1-ch data
    AX_U8 nShapeSize; // dimension of shape
    AX_JOINT_TENSOR_LAYOUT_T eLayout;
    AX_JOINT_MEMORY_TYPE_T eMemoryType;
    AX_JOINT_DATA_TYPE_T eDataType;
    AX_JOINT_IOMETA_EX_T* pExtraMeta;
    AX_U32 nSize;
    AX_U32 nQuantizationValue;
    AX_S32* pStride;
    AX_U64 _reserved[11];
} AX_JOINT_IOMETA_T;
```

pName：节点名称。

pShape：数据维度信息数组，数组大小等于 nShapeSize。

nShapeSize：数据维度大小。

eLayout：数据 Layout 形式。

eMemoryType：数据所在内存形式。

eDataType：数据类型。

pExtraMeta：附加 IO Meta 信息指针。

nSize：数据总大小。

nQuantizationValue：量化数据 Q 值，大小为 2**Q，例如对于 U4Q12 数据，nQuantizationValue = 2**12=4096。

pStride：数据跨度信息数组，数组大小等于 nShapeSize。

_reserved：保留字段。

AX_JOINT_IO_INFO_T

存储 Joint Model IO 信息。

```
typedef struct _AX_JOINT_IO_INFO_T {
    AX_JOINT_IOMETA_T *pInputs;
    AX_U32 nInputSize;
    AX_JOINT_IOMETA_T *pOutputs;
    AX_U32 nOutputSize;
    AX_U32 nMaxBatchSize;  // 0 for unlimited
    AX_BOOL bDynamicBatchSize;
    AX_U64 _reserved[13];
} AX_JOINT_IO_INFO_T;
```

pInputs：输入节点 IO Meta 信息数组首地址，数组大小等于 nInputSize。

nInputSize：输入节点 IO Meta 信息数量。

pOutputs：输出节点 IO Meta 信息数组首地址，数组大小等于 nOutputSize。

nOutputSize：输出节点 IO Meta 信息数量。

nMaxBatchSize：支持最大 batch 大小，0 为无限制。

bDynamicBatchSize：是否支持动态 batch。

_reserved：保留字段。

AX_JOINT_IO_BUFFER_T

Joint Model 节点数据 IO Buffer 结构体。

```
typedef struct _AX_JOINT_IO_BUFFER_T {
    X_ADDR phyAddr;
    AX_VOID *pVirAddr;
    AX_U32 nSize;
    X_S32 *pStride;
    AX_U8 nStrideSize;
    AX_U64 _reserved[13];
} AX_JOINT_IO_BUFFER_T;
```

phyAddr：数据物理地址。

pVirAddr：数据虚拟地址。

nSize：数据大小。

pStride：数据 stride 信息数组，数组大小等于 nStrideSize。

nStrideSize：数据 stride 信息数组大小。

_reserved：保留字段。

AX_JOINT_IO_T

Joint Model IO Buffer 结构体。

```
typedef struct _AX_JOINT_IO_T {
    AX_JOINT_IO_BUFFER_T *pInputs;
    AX_U32 nInputSize;
    AX_JOINT_IO_BUFFER_T *pOutputs;
    AX_U32 nOutputSize;
    AX_U32 nBatchSize;  // 0 for auto detection
    AX_JOINT_IO_SETTING_T* pIoSetting;
    AX_U64 _reserved[11];
} AX_JOINT_IO_T;
```

pInputs：输入节点数据 Buffer 数组首地址，数组大小等于 nInputSize。

nInputSize：输入节点 Buffer 数量。

pOutputs：输出节点数据 Buffer 数组首地址，数组大小等于 nOutputSize。

nOutputSize：输出节点 Buffer 数量。

nBatchSize：batch 大小，0 为自动探测。

pIoSetting：输入输出设置。

_reserved：保留字段。

AX_JOINT_EXECUTION_CONTEXT_SETTING_T

Joint Model 执行上下文设置。

```
typedef struct _AX_JOINT_EXECUTION_CONTEXT_SETTING_T {
    AX_U32 nBatchSize;
    AX_BOOL bNoCacheMem;
    AX_U64 nReserved[7];
} AX_JOINT_EXECUTION_CONTEXT_SETTING_T;
```

nBatchSize：batch 设置，0 为自动探测。

bNoCacheMem：禁用内存缓存。

nReserved：保留字段。

AX_JOINT_COMPONENT_PROFILE_T

Joint Model 运行耗时结构体。

```
typedef struct _AX_JOINT_COMPONENT_PROFILE_T {
    AX_U32 nTotalUs;
    AX_U32 nCoreUs;
    AX_U32 nInitUs;
    AX_U64 _reserved[6];
} AX_JOINT_COMPONENT_PROFILE_T;
```

nTotalUs：总体运行耗时（us）。

nCoreUs：NPU 运行耗时（us）。

nInitUs：初始化耗时（us）。

nReserved：保留字段。

AX_JOINT_COMPONENT_T

Joint Model 子图信息结构体。

```
typedef struct _ax630a_joint_component {
    AX_JOINT_COMPONENT_TYPE_T eType;
    const AX_S8* pName;
    AX_JOINT_COMPONENT_PROFILE_T tProfile;
    AX_NPU_SDK_EX_MODEL_TYPE_T eVNPUMode;
    AX_U64 _reserved[6];
} AX_JOINT_COMPONENT_T;
```

eType：子图类型。

pName：子图名称。

tProfile：运行耗时信息。

eVNPUMode：模型 VNPU 模式。

_reserved：保留字段。

4 错误码

RUN JOINT SDK 错误码如下表所示：

表4-1 RUN JOINT SDK 错误码列表

错误代码	宏定义	描述
0x80061001	AX_ERR_NPU_JOINT_UNKNOWN_FAILURE	未知错误。
0x80061002	AX_ERR_NPU_JOINT_INVALID_PARAM	无效参数。
0x80061003	AX_ERR_NPU_JOINT_INIT_FAILED	初始化失败。
0x80061004	AX_ERR_NPU_JOINT_MALFORMED_TOPOLOGY	非法模型拓扑。
0x80061005	AX_ERR_NPU_JOINT_CREATE_CONTEXT_FAILED	上下文创建失败。
0x80061006	AX_ERR_NPU_JOINT_RUN_FAILED	推理失败。

5 QA 相关问题

5.1 AX_JOINT_IO_INFO_T 的 nMaxBatchSize 与 shape[0] 是什么关系？

答：nMaxBatchSize 一般用于动态 batch 场景，是指模型推理允许的最大的 batch 个数，小于等于 nMaxBatchSize 的都可以送入到当前模型进行推理；shape[0] 与 nMaxBatchSize 是一样的。