ONI 结构化数据传输协议设计	
[备注]	
密级	机密

	修订历史			
版本	修订者	审核者	日期	备注
v0.1	常胤（林卫华）	大鱼（余学稳）	2020-11-09	1. 初版
v0.2	常胤（林卫华）	大鱼（余学稳）	2021-02-23	1. 新增 AI 流
v0.5	常胤（林卫华）	武松（谢利军）	2021-07-19	1. 新增相位流

ONI 结构化数据传输协议设计

1 概述

1.1 背景

昆仑山项目是基于 RK1126 平台，打造 TV 端的 RGBD 软硬件标品方案，为了更好地向客户展示，推广 RGBD 相机在电视行业中的应用，基于产品端提出的演示功能需求，实现 AR 魔镜演示 Demo，主要包括以下功能：

- 3D 骨架
- 实时抠图
- 人体测量
- 平面检测

当前 OpenNI 协议仅支持传输常规的视频流，例如：IR、Depth、Color，为适配设备端的 AI 结构化数据流，需对 OpenNI 协议进行扩展。

1.2 目的

设计先行，为后续方案讨论、编码实现、迭代维护，提供参考依据。

1.3 对象

本文档适用人员（Orbbec STAR 内部人员）：

- 技术
- 产品
- 项目

2 数据结构

2.1 基本单元

数据节点以 TLV 格式为基本结构单元，TLV 即 Tag/Type-Length-Value，是一种可变数据格式，TLV 协议简单高效，广泛适用于各种通信场景，具有良好的可扩展性。TLV 协议主要包括三个域，分别为：

- Tag/Type：标识域，表示编码数据类型
- Length：长度域，表示数据域实际长度

- Value：数据域，实际数据内容

考虑到客户实际应用场景，特别是在 TV 端，普遍采用网络协议，而非 OpenNI 协议传输，故在 TLV 协议基础上，增加时间戳域，即 TTLV 格式，并在数据域增加状态码字段，如下图所示：

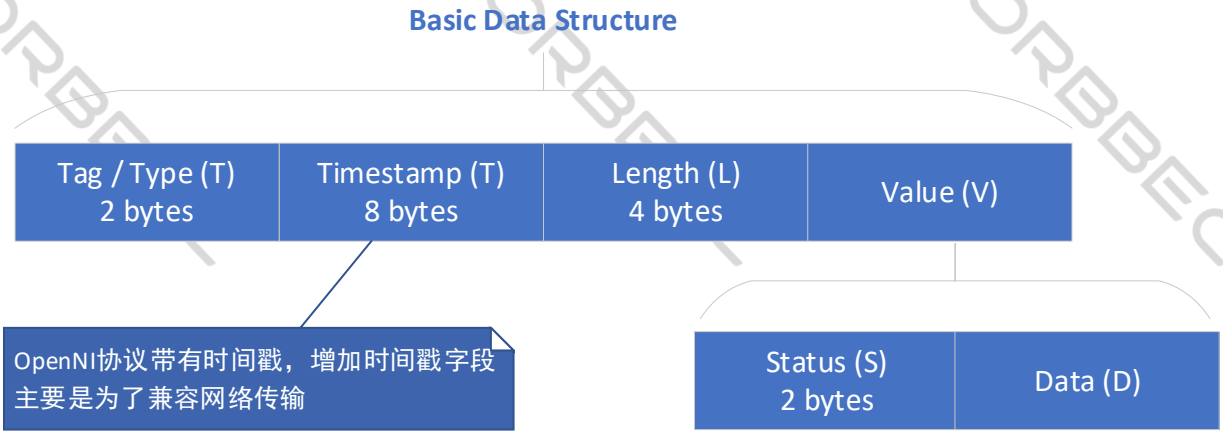


图 2-1

2.2 数据总览

在昆仑山项目中，RV1126 平台集成的数据节点主要有：

- 骨架
- 抠图
- 人体测量
- 平面检测
- 相位流（新增节点，由常规视频流模式调整为 AI 流模式）

数据节点负载链路，如下所示：

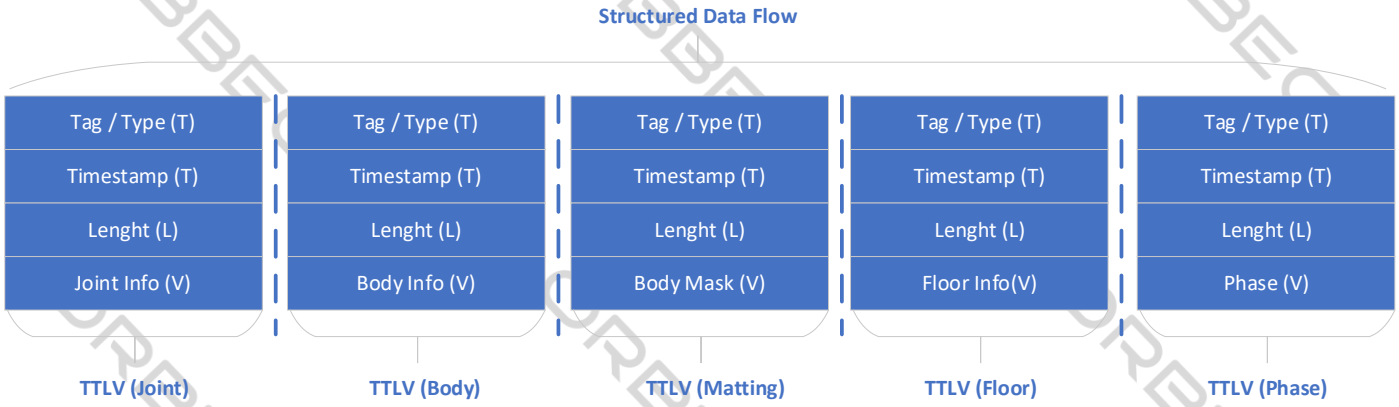


图 2-2

以上数据节点排列顺序并无强制要求，数据节点可传输单个、或多个，可根据实际应用场景排列组合，上层根据标识域进行节点数据解析。

3 数据节点

3.1 骨架

3.1.1 节点定义

骨架算法支持 2D/3D 两种模式，骨架点统一定义如下：

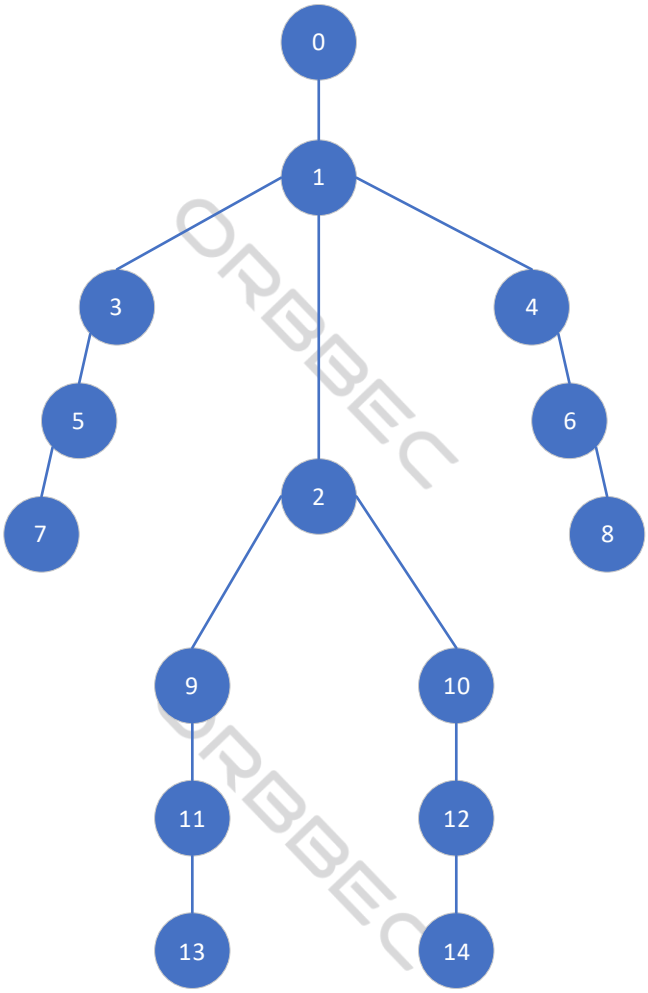


图 3-1

骨架点类型定义如下：

```
typedef enum
{
    ONI_JOINT_HEAD = 0,
    ONI_JOINT_NECK = 1,
    ONI_JOINT_MID_SPINE = 2,
    ONI_JOINT_RIGHT_SHOULDER = 3,
    ONI_JOINT_LEFT_SHOULDER = 4,
    ONI_JOINT_RIGHT_ELBOW = 5,
    ONI_JOINT_LEFT_ELBOW = 6,
    ONI_JOINT_RIGHT_WRIST = 7,
    ONI_JOINT_LEFT_WRIST = 8,
    ONI_JOINT_RIGHT_HIP = 9,
    ONI_JOINT_LEFT_HIP = 10,
    ONI_JOINT_RIGHT_KNEE = 11,
    ONI_JOINT_LEFT_KNEE = 12,
    ONI_JOINT_RIGHT_ANKLE = 13,
    ONI_JOINT_LEFT_ANKLE = 14,
    ONI_JOINT_MAX = 15,
} OniJointType;
```

图 3-2

3.1.2 数据负载

骨架数据负载定义如下：

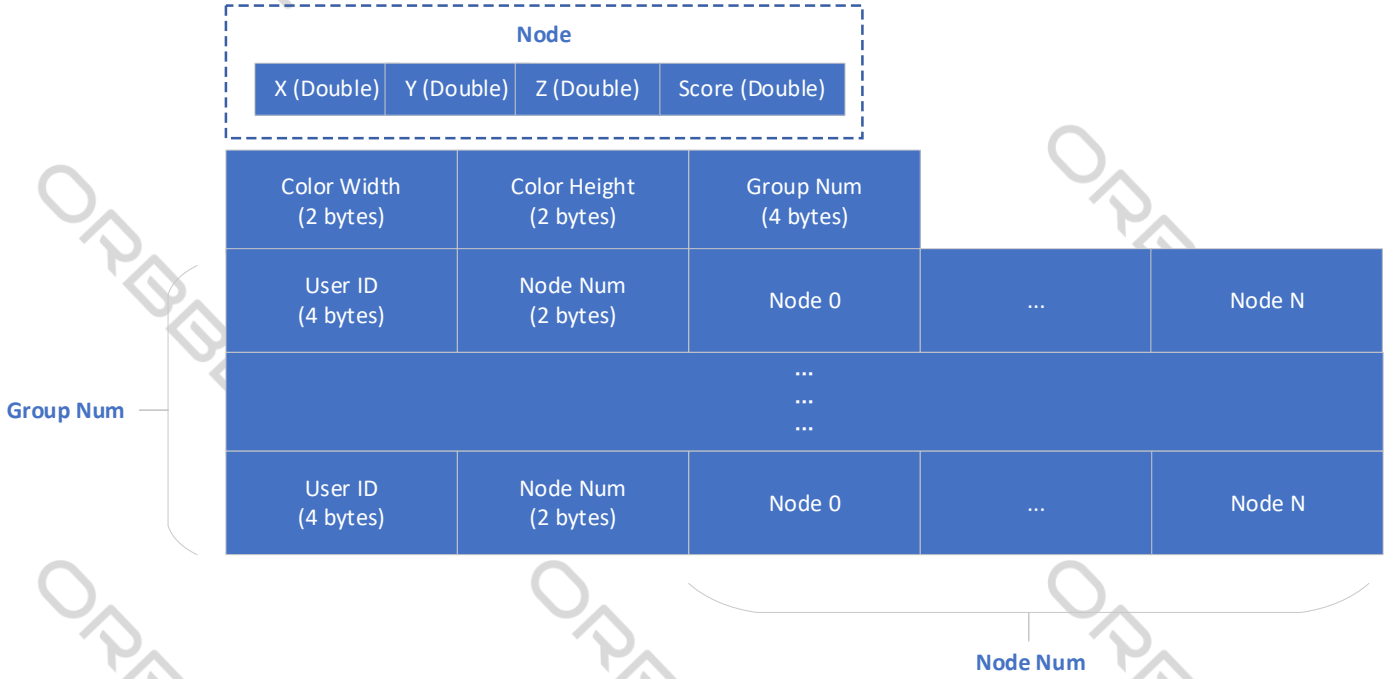


图 3-3

字段说明：

字段	类型	描述
X/Y/Z	double	3D 模式下为空间坐标；2D 模式下为图像坐标，Z 轴为 0

Score	double	当前骨架点置信度（<= 0 为无效骨架点）
Width	uint16	计算骨架时输入的 RGB 图像宽度
Height	uint16	计算骨架时输入的 RGB 图像高度
Group Num	uint32	骨架组数量（一人表示一组）
User ID	uint32	骨架 ID，每组骨架对应一个 ID
Node Num	uint16	每组骨架对应的节点数量（0 < num <= 15）
Node	struct	骨架节点：{ X, Y, Z, Score }

3.2 抠图

3.2.1 数据负载

受 RV1126 平台算力限制，抠图模块输出的结果为人体掩码数据，非扣好的完整图像，上位机接收到掩码数据后，需对彩色图和人体掩码做进一步的融合渲染处理。

人体掩码数据负载定义如下：

图 3-4

字段说明：

字段	类型	描述
Width	uint16	人体掩码图像宽度（像素单位：640）
Height	uint16	人体掩码图像高度（像素单位：384）
Data	uint8[]	人体掩码数据（bytes = width * height）

3.3 人体测量

3.3.1 类型定义

人体测量模块，主要功能是测量人体身高、身材比例等，在 AR 魔镜中，该功能定义为单人测量。

体型类型定义如下：

```
typedef enum
{
    ONI_FIGURE_NONE,
    ONI_FIGURE_THIN,
    ONI_FIGURE_OVERWEIGHT,
    ONI_FIGURE_MEDIUM_BUILT,
} OniFigureTypes;
```

图 3-5

类型说明：

- ONI_FIGURE_THIN：体型偏瘦
- ONI_FIGURE_OVERWEIGHT：体型偏胖
- ONI_FIGURE_MEDIUM_BUILT：体型适中

体型数据定义如下：

```
#pragma pack (push, 1)
typedef struct
{
    uint16_t id;          ///< Reserve
    float ratio;          ///< 比例
    float height;         ///< 身高
    float waist;          ///< 腰宽
    float waistline;      ///< 腰围 (Reserve)
    float bust;           ///< 胸围 (Reserve)
    float hips;           ///< 臀围 (Reserve)
    float shoulder;       ///< 肩宽 (Reserve)
    uint16_t figure;      ///< 体型 (Reserve)
    uint32_t points[6]; ///< Body Points
} OniBodyShape;
#pragma pack (pop)
```

图 3-6

字段说明：

字段	类型	描述
ID	uint16	被测量的人体 ID（保留字段）
Ratio	float	身材比例（保留字段）
Height	float	身高
Waist	float	腰宽（保留字段）
Waistline	float	腰围（保留字段）
Bust	float	胸围（保留字段）

Hips	float	臀围（保留字段）
Shoulder	float	肩宽（保留字段）
Figure	uint32	体型（保留字段）
Points	unt32[]	三个人体点坐标（用于辅助调试，判断是否检测到人体）

3.3.2 数据负载

体型数据负载定义如下：

图 3-7

字段说明：

- 当前算法仅支持单人测量，为方便后续扩展，协议设计为支持单人、或多人测量
- Length 为 TLV 协议头数据长度字段

3.4 平面检测

3.4.1 数据负载

平面检测模块输出结果为平面向量和掩码数据，空间坐标定义如下：

```
typedef struct
{
    float x;
    float y;
    float z;
} OniPoint;
```

图 3-8

平面数据负载定义如下：

Center (sizeof(OniPoint))	Normal (sizeof(OniPoint))	Mask Width (2 bytes)	Mask Height (2 bytes)	Mask Data

size = sizeof (OniPoint) * 2 + width * height * sizeof (unsigned char)

图 3-9

字段说明：

字段	类型	描述
Center	OniPoint	平面中心点
Normal	OniPoint	平面法向量
Width	uint16	平面掩码图像宽度（像素单位：640）
Height	uint16	平面掩码图像高度（像素单位：480）
Data	uint8[]	平面掩码数据（bytes = width * height）

3.5 相位流

3.5.1 数据格式

相位流的数据量和 TOF Sensor 的工作模式密切相关，不同工作模式，其对应的数据量是不一样的，以 33D 和 Pleco 为例，各模式下的相位流数据格式如下表所示：

Sensor/Mode	单频-NoShuffle	双频-NoShuffle	单频-Shuffle	双频-Shuffle
33D	1280 x 960	1280 x 960 x 2	1280 x 960 x 2	1280 x 960 x 4
Pleco	1920 x 480	1920 x 480 x 2	1920 x 480 x 3	1920 x 480 x 6
备注	相位数据解包后每像素占 2 个字节，每张相位图对应一行扩展数据，数据量计算方式： Total Bytes = width x (height + 1) x bit-wide / 8 x Num(图片张数)			

在昆仑山项目开发前期，优先适配了 33D，33D 相位流按逐帧方式输出（非帧组方式），故在 OpenNI 初版设计中，把相位流当成常规的视频流来处理，而 Pleco 是以帧组的方式输出，为兼容 33D 和 Pleco，需额外做以下适配工作：

- 平台端需对 Pleco 帧组进行拆分，按逐帧方式打包发送
- 上位机收到数据后，需把拆分后的帧数据，重新进行组帧

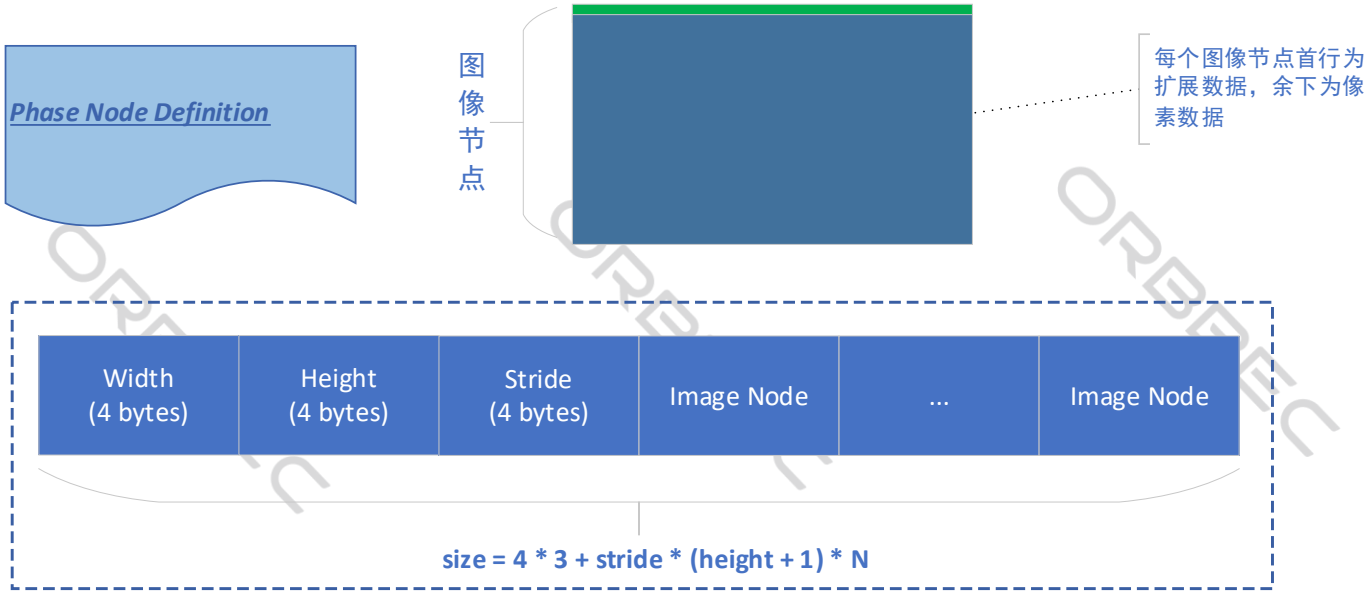
这种方式不仅增加了代码复杂性，在相位流帧率较高的情况下，还会导致上位机应用经常出现组帧失败的情况，为解决以上弊端，统一按帧组的方式，通过 TLV 格式传输，将相位流当成特殊的 AI 流节点来处理，这种方式的优势体现在：

- 省去了拆帧、组帧操作，降低了代码复杂性

- 统一了 33D 和 Pleco 的处理方式，代码更通用
- 避免帧率过高，导致上位机应用出现组帧失败的情况
- TOF Sensor 的工作模式将变得透明，SDK 根据节点标识域即可解析出不同模式下的数据
- 相位流不再和深度流复用一个端点，在平台端支持的情况下，可同时输出相位和深度流

3.5.2 数据负载

相位流节点数据负载定义如下：



字段说明：

字段	类型	描述
Width	uint32	图像宽度（像素单位）
Height	uint32	图像高度（像素单位）
Stride	uint32	行字节数
Image Node	uint8[]	图像数据节点，首行为扩展数据 bytes = stride * (height + 1)

4 交互协议

4.1 AI 端点

指定 EP-0x83，作为 AI 数据流统一传输端点，定义如下：

端点（EP）	流类型	描述
0x0	控制端点	用于控制指令等小型数据传输
0x81	Depth	用于深度流传输
0x82	IR | Color	用于 IR 或彩色流传输（端点复用）
0x83	AI	用于 AI 流传输（相位、骨架、抠图、人体、平面）

新增 USB 数据包类型，以区分 AI 数据流：

Packet Type	IR | Color	Depth	AI	描述
SOF	0x8100	0x7100	0x6100	起始包
Normal	0x8200	0x7200	0x6200	中间包
EOF	0x8500	0x7500	0x6500	结束包

注意：AI 流不是常规视频流，是单个、或多个结构化数据的组合，当数据量较小时，需考虑只有一个数据包的情况。

4.2 AI 流控制

新增 AI Video Mode 类型，用于 AI 开流指令交互：

```
typedef enum
{
    XN_VIDEO_STREAM_OFF = 0,
    XN_VIDEO_STREAM_COLOR = 1,
    XN_VIDEO_STREAM_DEPTH = 2,
    XN_VIDEO_STREAM_IR = 3,
    XN_VIDEO_STREAM_AUDIO = 4,
    XN_VIDEO_STREAM_PHASE = 5,
    XN_VIDEO_STREAM_AI = 6,
} XnVideoStreamMode;
```

图 3-10

新增 AI 流常规属性，用于流属性获取、设置交互：

```
typedef enum
{
    /// IR

    /// Image

    /// Depth

    /// Phase
    PARAM_PHASE_FPS = 87,
    PARAM_PHASE_FORMAT = 88,
    PARAM_PHASE_MIRROR = 89,
    PARAM_PHASE_RESOLUTION = 90,
    PARAM_PHASE_CROP_MODE = 91,
    PARAM_PHASE_CROP_SIZE_X = 92,
    PARAM_PHASE_CROP_SIZE_Y = 93,
    PARAM_PHASE_CROP_OFFSET_X = 94,
    PARAM_PHASE_CROP_OFFSET_Y = 95,

    /// AI
    PARAM_GENERAL_STREAM3_MODE = 96,
    PARAM_AI_FPS = 97,
    PARAM_AI_FORMAT = 98,
    PARAM_AI_MIRROR = 99,
    PARAM_AI_RESOLUTION = 100,

    PARAM_FREQUENCY_MODE = 110,
} XnConfigParams;
```

图 3-11

新增 AI Formats 类型，用于 AI 流模式切换：

```
typedef enum
{
    XN_IO_AI_FORMAT_JOINT_2D = 0x0001,
    XN_IO_AI_FORMAT_JOINT_3D = 0x0002,
    XN_IO_AI_FORMAT_BODY_MASK = 0x0004,
    XN_IO_AI_FORMAT_FLOOR_INFO = 0x0008,
    XN_IO_AI_FORMAT_BODY_SHAPE = 0x0010,
    XN_IO_AI_FORMAT_PHASE = 0x0020,
    XN_IO_AI_FORMAT_FACE = 0x0040,
    XN_IO_AI_FORMAT_GESTURE = 0x0080,
} XnIOAIFormats;
```

图 3-12

字段说明：

字段	类型	描述

JOINT_2D	enum	2D 骨架
JOINT_3D	enum	3D 骨架
BODY_MASK	enum	抠图掩码
FLOOR_INFO	enum	平面信息
BODY_SHAPE	enum	体型数据
PHASE	enum	相位流
FACE	enum	人脸信息（保留字段）
GESTURE	enum	手势信息（保留字段）

AI 模式切换，交互指令如下：

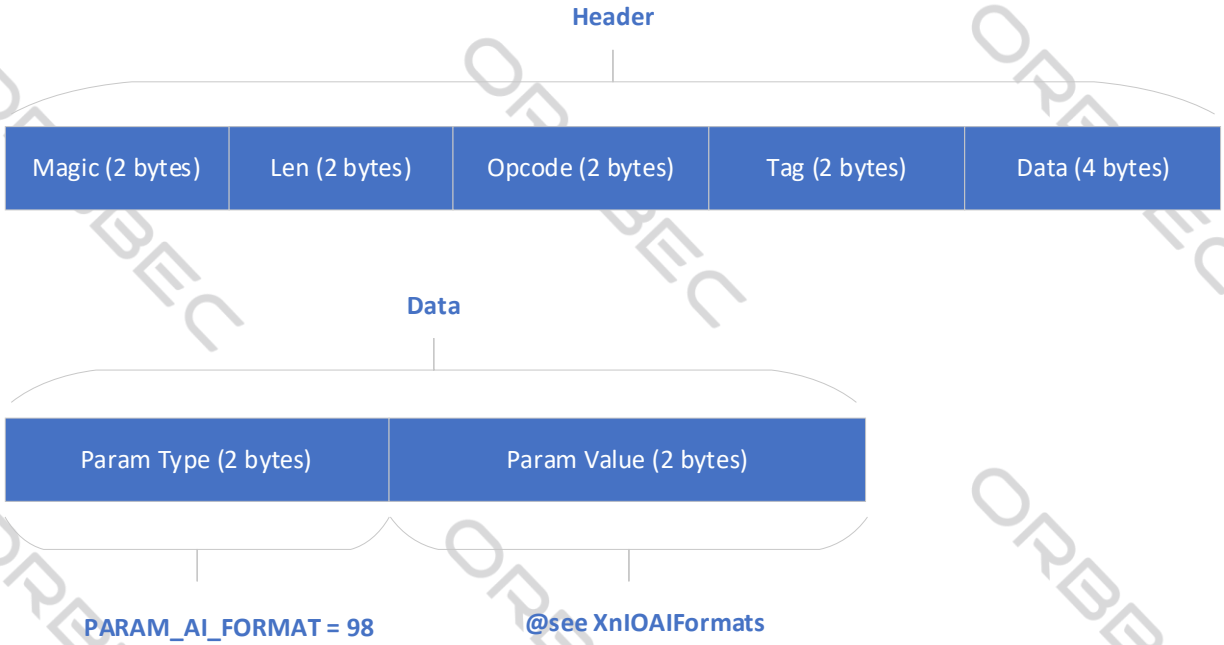


图 3-13

字段说明：

字段	类型	描述
Magic	uint16	魔数（Host -> Device: 0x4D47）
Len	uint16	指令数据长度（DWORD 类型，即：Data Size / 2）
Opcode	uint16	操作码
Tag	uint16	指令标签
Param Type	uint16	参数类型（此处为 AI FORMAT 属性）
Param Value	uint16	参数值（此处为 XnIOAIFormats）

注意：AI 模式可以是单个、或多个类型的组合（除 2D/3D 骨架不能同时输出），例如期望同时获取骨架和体型

数据，Param Value 可设置为：XN_IO_AI_FORMAT_JOINT_3D | XN_IO_AI_FORMAT_BODY_SHAPE。

由于当前 TOF Sensor 不支持同时输出相位和深度流，因此依赖深度的算法节点和相位节点是互斥的，依赖深度的算法节点有：

- 3D 骨架
- 平面检测
- 人体测量

以上算法不能和相位流同时使能。

5 风险

- 在 AR 魔镜功能定义中，AI 功能模块（骨架、抠图、人体测量、平面检测）单独运行，无需同时运行所有 AI 模块，考虑实际应用场景，如游戏开发，可能需要用到多维度数据，如骨架、手势、平面、视频流等，此场景下，平台算力、传输带宽可能成为瓶颈。

6 附录

略。