



# CV186AH DPU API 使用手册

Version: 1.0.0

Release date: 2023/12

©2022 北京晶视智能科技有限公司  
本文件所含信息归北京晶视智能科技有限公司所有。  
未经授权，严禁全部或部分复制或披露该等信息。

# 目录

<b>1</b>	<b>声明</b>	<b>2</b>
<b>2</b>	<b>概述</b>	<b>3</b>
2.1	DPU 模块概述 . . . . .	3
2.2	视差概述 . . . . .	3
2.3	DPU 模块规格 . . . . .	4
<b>3</b>	<b>功能</b>	<b>5</b>
3.1	Block Overview . . . . .	5
3.2	Census 变换 . . . . .	5
3.3	Birchfield and Tomasi Cost . . . . .	6
3.4	Add . . . . .	6
3.5	Box Filter . . . . .	7
3.6	DCC . . . . .	7
3.7	Winner Take All . . . . .	8
3.8	Post Processing . . . . .	8
3.9	FGS . . . . .	9
<b>4</b>	<b>数据结构</b>	<b>11</b>
4.1	数据结构总览 . . . . .	11
4.2	数据结构说明 . . . . .	11
4.2.1	DPU_GRP . . . . .	11
4.2.2	DPU_GRP_ATTR_S . . . . .	12
4.2.3	DPU_CHN . . . . .	14
4.2.4	DPU_CHN_ATTR_S . . . . .	15
4.2.5	VIDEO_FRAME_INFO_S . . . . .	15
4.2.6	DPU_DISP_RANGE_E . . . . .	16
4.2.7	DPU_MASK_MODE_E . . . . .	17
4.2.8	DPU_DEPTH_UNIT_E . . . . .	18
4.2.9	DPU_DCC_DIR_E . . . . .	19
4.2.10	DPU_MODE_E . . . . .	20
<b>5</b>	<b>应用程序接口</b>	<b>23</b>
5.1	API 总览 . . . . .	23
5.2	API 流程 . . . . .	24
5.3	API 说明 . . . . .	24
5.3.1	CVI_DPU_CreateGrp . . . . .	24
5.3.2	CVI_DPU_DestroyGrp . . . . .	26
5.3.3	CVI_DPU_SetGrpAttr . . . . .	27
5.3.4	CVI_DPU_GetGrpAttr . . . . .	28
5.3.5	CVI_DPU_StartGrp . . . . .	29

5.3.6	CVI_DPU_StopGrp . . . . .	30
5.3.7	CVI_DPU_SetChnAttr . . . . .	32
5.3.8	CVI_DPU_GetChnAttr . . . . .	33
5.3.9	CVI_DPU_EnableChn . . . . .	34
5.3.10	CVI_DPU_DisableChn . . . . .	35
5.3.11	CVI_DPU_SendFrame . . . . .	36
5.3.12	CVI_DPU_GetFrame . . . . .	38
5.3.13	CVI_DPU_ReleaseFrame . . . . .	39
<b>6</b>	<b>错误码</b>	<b>41</b>
6.1	错误码 . . . . .	41

## 修订记录

Revision	Date	Description
1.0.0	2023/08/30	初稿

# 1 声明

---



## 法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

## 联系我们

**地址** 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

**电话** +86-10-57590723 +86-10-57590724

**邮编** 100094（北京）518100（深圳）

**官方网站** <https://www.sophgo.com/>

**技术论坛** <https://developer.sophgo.com/forum/index.html>

# 2 概述

## 2.1 DPU 模块概述

DPU (Depth Process Unit) 深度处理单元，其可通过两张经过双目校正后的左图和右图，来计算图像的视差/深度信息。它主要包括双目立体匹配 (Semi-Global Block Matching, SGBM) 模块与快速全局平滑滤波 (Fast Global Smooth, FGS) 模块。其中 SGBM 模块用以计算视差图，包括以下功能：Census Transform、BT cost 计算、Box filter、代价聚合、视差计算、亚像素内插、唯一性检查、中值滤波器。FGS 模块用以平滑视差图，且包含视差转深度功能。DPU 示意图如下：



图 2.1: DPU 示意图

## 2.2 视差概述

视差 (Disparity) 是指同一个物体点在左图与右图上的像素差 ( $X_R - X_T$ )。

当物体距离相机越近，左右图的视差越大，而距离越远，视差越小。因此我们可以通过视差来感知到物体的远近和立体效果。

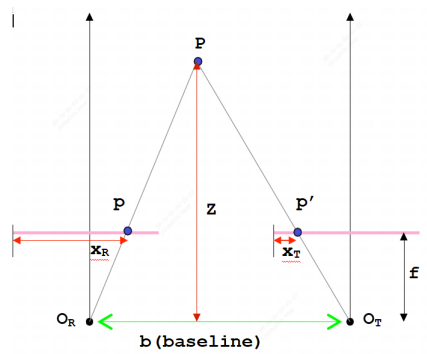


图 2.2: 双目视差示意

## 2.3 DPU 模块规格

项目	描述
Input Format	Y only 8 bit
Output Format	Disparity 8 bit、Disparity 16 bit、Depth 16 bit
Performance	1080p@30
Max resolution	1920x1080
Min resolution	64x64
Max search Depth	128
Align	16 bytes

# 3 功能

## 3.1 Block Overview

下图展示了 DPU 的处理流程，旨在获得最佳的视差图。处理流程如下：首先，左图和右图是经过双目校正后的图像，先对左图和右图进行 Census Transform，并使用 BT 方法计算 Cost map。为了获得最佳效果，还将原始输入灰度图像计算的 Cost map 与之相加。然后，通过 Box filter 来计算出 SAD (Sum of Absolute Differences)。接着，对来自多个方向的代价进行聚合，使用 Winner Take All 机制获得视差图。最后，通过唯一性检查、亚像素内插、中值滤波来优化视差图，并考虑保留边缘信息以实现平滑效果 (FGS)。

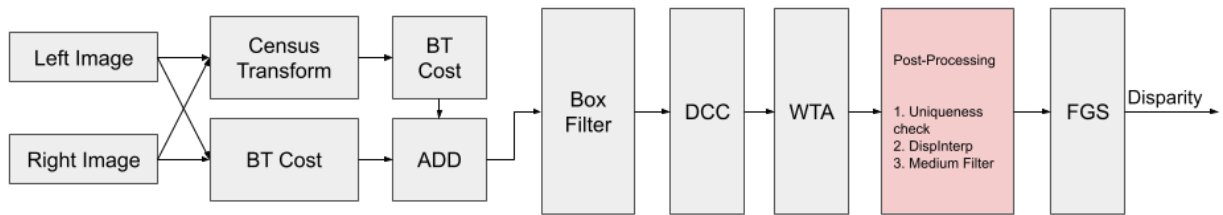


图 3.1: DPU 处理流程

## 3.2 Census 变换

Census 变换 (Census Transform) 主要是将像素值与领域像素值进行比较，来获得一个二进制编码。该二进编码可以用来进行像素相似性匹配，来减少左右图的亮度差影响。其原理可以参考下图：



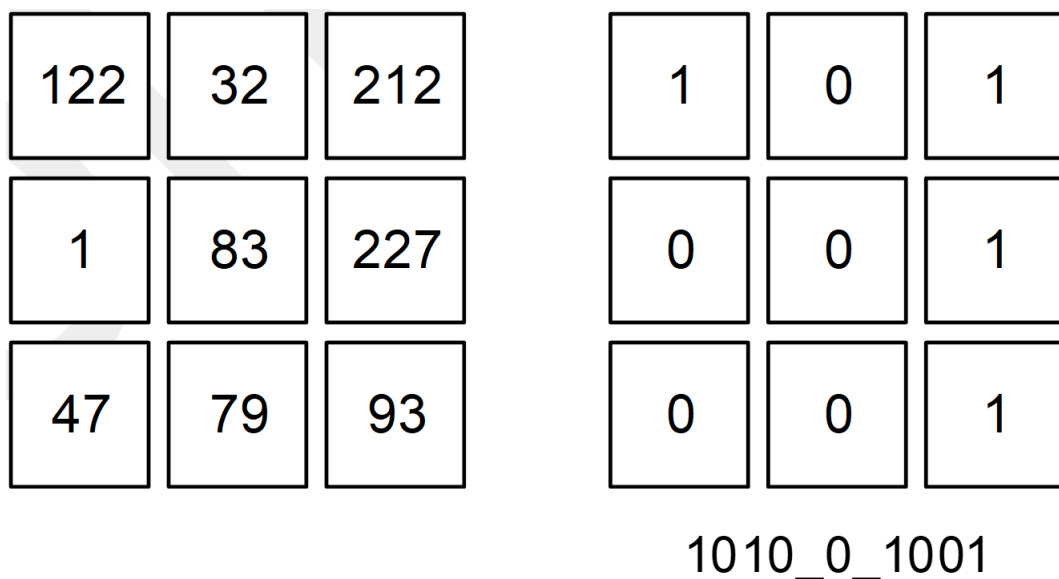


图 3.2: Census 变换

### 3.3 Birchfield and Tomasi Cost

Birchfield & Tomasi Cost 主要是对左图和右图的亚像素进行线性插值采样，并求出与参考像素相差最小的 value 值，最后取左图和右图最小的 value 值为 cost。其主要是来解决深度不连续问题。下图为 Birchfield & Tomasi Cost 算法示例。

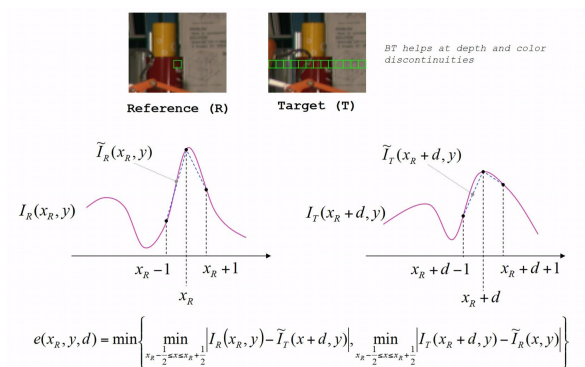


图 3.3: Birchfield &amp; Tomasi Cost 示例

### 3.4 Add

Add 主要是将原图的 BT Cost 和 Census 图的 BT cost 进行加权相加，从而获得最终的 Cost。

## 3.5 Box Filter

Box Filter(盒子滤波) 主要是对指定区域的像素进行加速求和。

- 支持 Boxfilter 窗口大小调节, 调节范围为 1x1(无滤波)、3x3、5x5、7x7。
- DPU 支持将 Box filter 的输出结果直接写出 DRAM, 为用户设计 DPU 算法提供数据, 但在正常下模式下不建议使用, 耗时较大。

## 3.6 DCC

代价聚合 (Disparity Cost Aggregation, DCC) 主要是将不同方向上的代价聚合值进行求和, 来弥补固定 SAD 窗口在深度不连续、无纹理等情况下计算出的代价不够精确的问题。其计算公式如下

$$L_r(p, d) = C(p, d) + \min \left\{ \begin{array}{l} L_r(p-r, d) \\ L_r(p-r, d-1) + P_1 \\ L_r(p-r, d+1) + P_1 \\ \min_i L_r(p-r, i) + P_2 \end{array} \right\} - \min_i L_r(p-r, i)$$

图 3.4: 代价聚合计算公式

DPU 的 DCC 固定支持 A1 方向的代价聚合, 第二方向可由用户在 A2、A3、A4 方向自由选择其一。

- A1 方向: 由左向右
- A2 方向: 由左上向右下
- A3 方向: 由上向下
- A4 方向: 由右上向左下

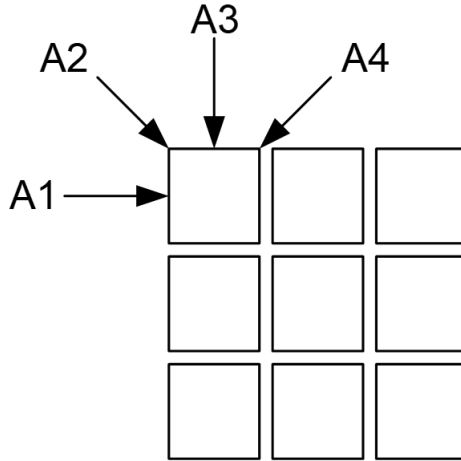
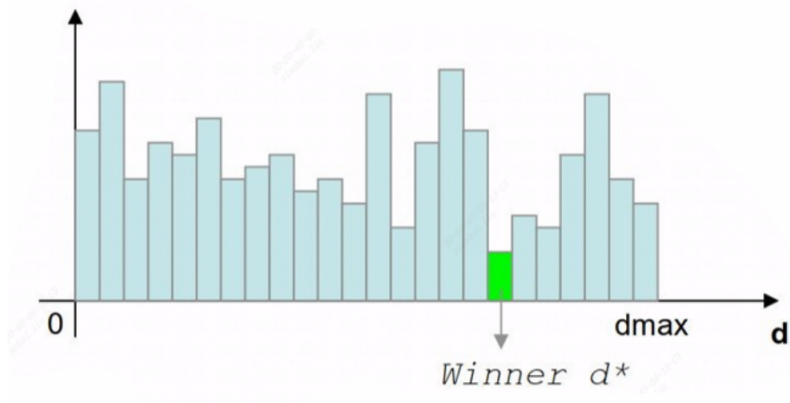


图 3.5: DCC 聚合方向

### 3.7 Winner Take All

赢者通吃 (Winner-take-all) 主要在指定的视差范围内提取出代价聚合值最小的视差。



### 3.8 Post Processing

后处理经过四个步骤：唯一性检查、亚像素内插、中值滤波和 U16 转 U8。

- **唯一性检查：** 唯一性检查通过比较最小代价值与次小代价值之间的距离，来确定该点的结果是否有效。可根据 `dpu_uniq_ratio` 来定义阈值距离。
- **亚像素内插：** 经过唯一性检查后，将确定的最小代价值与相邻代价值进行二次曲线拟合，以获得更准确的最小代价值。而更准确的最小代价值所对应的极值坐标即为准确的视差值  $d_{sub}$ 。下图展示了 Disparity Interpolation 原理。

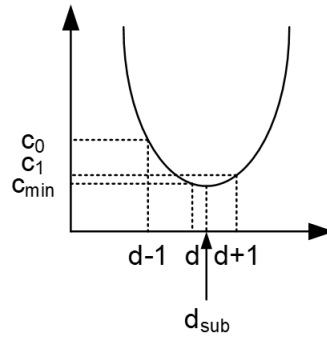
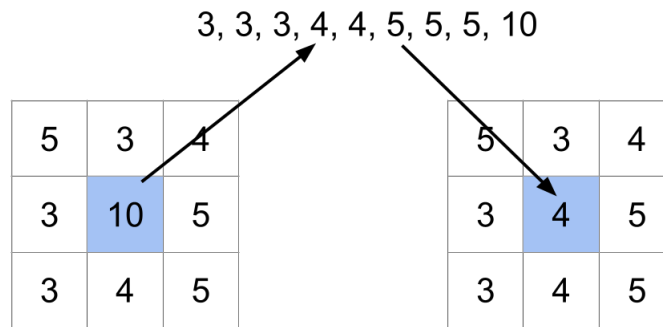


图 3.6: Disparity Interpolation 原理

#### · 中值滤波:

使用给定的滑动窗口，在窗口内对视差值进行排序比较，确定中间的视差值，并将该值替代原窗口中心点的视差值。对于原图像中某点  $(i, j)$ ，中值滤波以该点为中心的邻域内的所有像素的统计排序中值作为  $(i, j)$  点的计算结果。下图展示了中值滤波器的计算方式。



- **U16 转 U8:** 由于 FGS 仅支持处理 8 位数据，因此 SGBM 出口需要将经过亚像素内插后扩展的 16 位视差值转换为 8 位。DPU 支持将中值滤波后的 U16 或 U8 数据写入 DRAM，以满足用户对不同视差精度的需求。

## 3.9 FGS

FGS (Fast Global Smooth) 快速全局平滑滤波，是一种基于优化的全局保边平滑滤波器，主要通过最小化能量函数（如下图公式）来实现对图像做平滑操作的同时，还可对边缘进行保护。FGS 不仅支持是视差图输出还支持深度图输出。

$$J(u) = \sum_p ((u_p - f_p)^2 + \lambda \sum_{q \in N(p)} w_{p,q}(g)(u_p - u_q)^2)$$

如下图所示，输入两张图（源图及噪声图），通过 FGS 处理后可得到一张去噪图片。

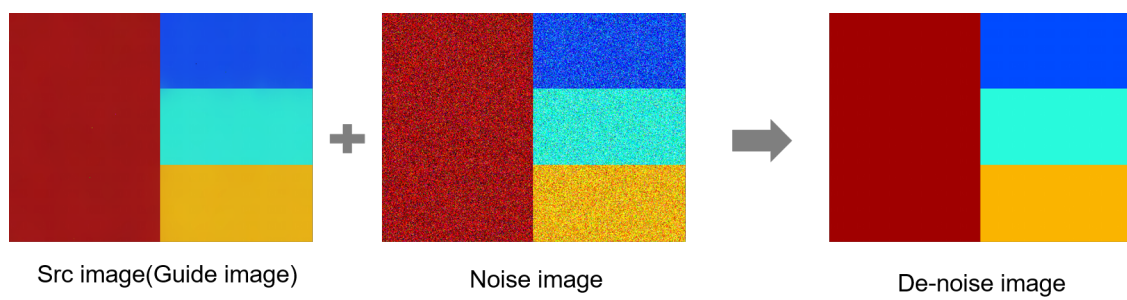


图 3.7: FGS 处理示例

# 4 数据结构

## 4.1 数据结构总览

· DPU 模块主要提供了以下数据结构:

- `DPU_GRP`: 定义 DPU GROUP ID
- `DPU_GRP_ATTR_S`: 定义 DPU GROUP 属性
- `DPU_CHN`: 定义 DPU channel ID
- `DPU_CHN_ATTR_S`: 定义 DPU channel ID 属性
- `VIDEO_FRAME_INFO_S`: 定义视频帧信息
- `DPU_DISP_RANGE_E`: 定义 DPU 表示的深度范围
- `DPU_MASK_MODE_E`: 定义 DPU 的掩码模式
- `DPU_DEPTH_UNIT_E`: 定义 DPU 的深度单位
- `DPU_DCC_DIR_E`: 定义 DPU 的 DCC 代价聚合方向
- `DPU_MODE_E`: 定义 DPU 的输出模式

## 4.2 数据结构说明

### 4.2.1 `DPU_GRP`

#### 【描述】

定义 DPU GROUP ID。

#### 【定义】

```
typedef CVI_S32 DPU_GRP;
```

**【成员】**

无

**【注意事项】**

无

**【相关类型及接口】**

- CVI\_DPU\_CreateGrp
- CVI\_DPU\_DestroyGrp
- CVI\_DPU\_StartGrp
- CVI\_DPU\_StopGrp

## 4.2.2 DPU\_GRP\_ATTR\_S

**【描述】**

定义 DPU GROUP 属性。

**【定义】**

```
typedef struct _DPU_GRP_ATTR_S {  
  
    SIZE_S stLeftImageSize;  
    SIZE_S stRightImageSize;  
    DPU_MODE_E enDpuMode;  
    DPU_MASK_MODE_E enMaskMode;  
    DPU_DISP_RANGE_E enDispRange;  
    CVI_U16 u16DispStartPos;  
    CVI_U32 u32Rshift1;  
    CVI_U32 u32Rshift2;  
    CVI_U32 u32CaP1;  
    CVI_U32 u32CaP2;  
    CVI_U32 u32UniqRatio;  
    CVI_U32 u32DispShift;  
    CVI_U32 u32CensusShift;  
    CVI_U32 u32FxBaseline;  
    DPU_DCC_DIR_E enDccDir;  
    CVI_U32 u32FgsMaxCount;  
    CVI_U32 u32FgsMaxT;  
}
```

(下页继续)

(续上页)

```

DPU_DEPTH_UNIT_E enDpuDepthUnit;
CVI_BOOL bIsBtcostOut;
CVI_BOOL bNeedSrcFrame;
FRAME_RATE_CTRL_S stFrameRate;

} DPU_GRP_ATTR_S;

```

## 【成员】

表 4.1: DPU\_GRP\_ATTR\_S 成员表

成员名称	描述
stLeftImageSize	左图分辨率，取值范围：64x64~1920x1080
stRightImageSize	右图分辨率，取值范围：64x64~1920x1080
enDpuMode	DPU 输出模式，取值可参考 DPU_MODE_E 说明
enMaskMode	SAD 滤波窗口大小，取值可参考 DPU_MASK_MODE_E 说明
enDispRange	右图的搜索范围，取值可参考 DPU_DISP_RANGE_E 说明
u16DispStartPos	右图的搜索起始位置
u32Rshift1	census image 的右偏移量
u32Rshift2	原图的右偏移量
u32CaP1	P1 惩罚因子
u32CaP2	P2 惩罚因子
u32UniqRatio	唯一性检查因子，取值范围 [0,100]
u32DispShift	视差偏移量
u32CensusShift	Census 偏移量
u32FxBaseline	左右摄像头的基线值
enDccDir	代价聚合的方向，取值可参考 DPU_DCC_DIR_E 说明
u32FgsMaxCount	Fgs 中转变为 0 的最大次数
u32FgsMaxT	Fgs 的最大迭代次数
enDpuDepthUnit	深度的度量单位，取值可参考 DPU_DEPTH_UNIT_E 说明
bIsBtcostOut	是否输出 btcost 的结果
bNeedSrcFrame	是否需要源图像
stFrameRate	帧率控制

## 【注意事项】

- 右图像的 width 必须大于或等于视差搜索的起始位置与视差搜索范围的和，即：  
 $\text{stRightImageSize.u32Width} \geq \text{u16DispStartPos} + \text{enDispRange}$



- 左右图像的 height 和 width 必须相同。
- 左右图像的 width 必须按要求对齐。

**【相关类型及接口】**

- `CVI_DPU_SetGrpAttr`
- `CVI_DPU_GetGrpAttr`

### 4.2.3 DPU\_CHN

**【描述】**

定义 DPU channel ID。

**【定义】**

```
typedef CVI_S32 DPU_CHN;
```

**【成员】**

无

**【注意事项】**

无

**【相关类型及接口】**

- `CVI_DPU_EnableChn`
- `CVI_DPU_DisableChn`

## 4.2.4 DPU\_CHN\_ATTR\_S

### 【描述】

定义 DPU channel 属性。

### 【定义】

```
typedef struct _DPU_CHN_ATTR_S {  
    SIZE_S stImgSize;  
} DPU_CHN_ATTR_S;
```

### 【成员】

表 4.2: DPU\_CHN\_ATTR\_S 成员表

成员名称	描述
stImgSize	输出结果图的分辨率，取值范围：64x64~1920x1080

### 【注意事项】

- 输出图像的 height 和 width 必须按要求对齐。

### 【相关类型及接口】

- [CVI\\_DPU\\_SetChnAttr](#)
- [CVI\\_DPU\\_GetChnAttr](#)

## 4.2.5 VIDEO\_FRAME\_INFO\_S

### 【描述】

定义视频帧信息。

### 【定义】

```
typedef struct _VIDEO_FRAME_INFO_S {  
    VIDEO_FRAME_S stVFrame;  
    CVI_U32 u32PoolId;  
} VIDEO_FRAME_INFO_S;
```

**【成员】**

表 4.3: VIDEO\_FRAME\_INFO\_S 成员表

成员名称	描述
stVFrame	视频帧信息
u32PoolId	VB pool ID

**【注意事项】**

无

**【相关类型及接口】**

- CVI\_DPU\_SendFrame
- CVI\_DPU\_GetFrame
- CVI\_DPU\_ReleaseFrame

## 4.2.6 DPU\_DISP\_RANGE\_E

**【描述】**

定义 DPU 表示的深度范围。

**【定义】**

```
typedef enum _DPU_DISP_RANGE_E{  
    DPU_DISP_RANGE_DEFAULT = 0x0,  
    DPU_DISP_RANGE_16      = 0x1,  
    DPU_DISP_RANGE_32      = 0x2,  
    DPU_DISP_RANGE_48      = 0x3,  
    DPU_DISP_RANGE_64      = 0x4,  
    DPU_DISP_RANGE_80      = 0x5,  
    DPU_DISP_RANGE_96      = 0x6,
```

(下页继续)

(续上页)

```
DPU_DISP_RANGE_112 = 0x7,  
DPU_DISP_RANGE_128 = 0x8,  
DPU_DISP_RANGE_BUTT  
}DPU_DISP_RANGE_E;
```

**【成员】**

表 4.4: DPU\_DISP\_RANGE\_E 成员表

成员	描述
DPU_DISP_RANGE_DEFAULT	默认深度范围
DPU_DISP_RANGE_16	深度范围为 16 pixel
DPU_DISP_RANGE_32	深度范围为 32 pixel
DPU_DISP_RANGE_48	深度范围为 48 pixel
DPU_DISP_RANGE_64	深度范围为 64 pixel
DPU_DISP_RANGE_80	深度范围为 80 pixel
DPU_DISP_RANGE_96	深度范围为 96 pixel
DPU_DISP_RANGE_112	深度范围为 112 pixel
DPU_DISP_RANGE_128	深度范围为 128 pixel
DPU_DISP_RANGE_BUTT	枚举数值的最大值，用于判断输入的枚举值是否在范围内

**【注意事项】**

无

**【相关类型及接口】**

· DPU\_GRP\_ATTR\_S

## 4.2.7 DPU\_MASK\_MODE\_E

**【描述】**

定义 DPU 的掩码模式。

**【定义】**

```
typedef enum _DPU_MASK_MODE_E{
    DPU_MASK_MODE_DEFAULT = 0x0,
    DPU_MASK_MODE_1x1     = 0x1,
    DPU_MASK_MODE_3x3     = 0x2,
    DPU_MASK_MODE_5x5     = 0x3,
    DPU_MASK_MODE_7x7     = 0x4,
    DPU_MASK_MODE_BUTT
}DPU_MASK_MODE_E;
```

**【成员】**

表 4.5: DPU\_MASK\_MODE\_E 成员表

成员	描述
DPU_MASK_MODE_DEFAULT	默认 SAD 滤波窗口
DPU_MASK_MODE_1x1	SAD 滤波窗口为 1x1
DPU_MASK_MODE_3x3	SAD 滤波窗口为 3x3
DPU_MASK_MODE_5x5	SAD 滤波窗口为 5x5
DPU_MASK_MODE_7x7	SAD 滤波窗口为 7x7
DPU_MASK_MODE_BUTT	枚举数值的最大值，用于判断输入的枚举值是否在范围内

**【注意事项】**

无

**【相关类型及接口】**

· DPU\_GRP\_ATTR\_S

## 4.2.8 DPU\_DEPTH\_UNIT\_E

**【描述】**

定义 DPU 的深度单位。

**【定义】**

```
typedef enum _DPU_DEPTH_UNIT_E{
    DPU_DEPTH_UNIT_DEFAULT = 0x0,
    DPU_DEPTH_UNIT_MM      = 0x1,
    DPU_DEPTH_UNIT_CM      = 0x2,
    DPU_DEPTH_UNIT_DM      = 0x3,
    DPU_DEPTH_UNIT_M       = 0x4,
    DPU_DEPTH_UNIT_BUTT
}DPU_DEPTH_UNIT_E;
```

**【成员】**

表 4.6: DPU\_DEPTH\_UNIT\_E 成员表

成员	描述
DPU_DEPTH_UNIT_DEFAULT	默认深度单位
DPU_DEPTH_UNIT_MM	深度单位为 mm
DPU_DEPTH_UNIT_CM	深度单位为 cm
DPU_DEPTH_UNIT_DM	深度单位为 dm
DPU_DEPTH_UNIT_M	深度单位为 m
DPU_DEPTH_UNIT_BUTT	枚举数值的最大值，用于判断输入的枚举值是否在范围内

**【注意事项】**

无

**【相关类型及接口】**

· DPU\_GRP\_ATTR\_S

## 4.2.9 DPU\_DCC\_DIR\_E

**【描述】**

定义 DPU 的 DCC 代价聚合方向。

**【定义】**

```
typedef enum _DPU_DCC_DIR_E{
    DPU_DCC_DIR_DEFAULT = 0x0,
    DPU_DCC_DIR_A12      = 0x1,
    DPU_DCC_DIR_A13      = 0x2,
    DPU_DCC_DIR_A14      = 0x3,
    DPU_DCC_DIR_BUTT
}DPU_DCC_DIR_E;
```

**【成员】**

表 4.7: DPU\_DCC\_DIR\_E 成员表

成员	描述
DPU_DCC_DIR_DEFAULT	DCC 的默认代价聚合方向
DPU_DCC_DIR_A12	DCC 的代价聚合方向为 A1+A2
DPU_DCC_DIR_A13	DCC 的代价聚合方向为 A1+A3
DPU_DCC_DIR_A14	DCC 的代价聚合方向为 A1+A4
DPU_DCC_DIR_BUTT	枚举数值的最大值，用于判断输入的枚举值是否在范围内

**【注意事项】**

无

**【相关类型及接口】**

· DPU\_GRP\_ATTR\_S

## 4.2.10 DPU\_MODE\_E

**【描述】**

定义 DPU 的输出模式。

**【定义】**

```
typedef enum _DPU_MODE_E{
    DPU_MODE_DEFAULT = 0x0,
    DPU_MODE_SGBM_MUX0 = 0x1,
```

(下页继续)

(续上页)

```

DPU_MODE_SGBM_MUX1 = 0x2,
DPU_MODE_SGBM_MUX2 = 0x3,
DPU_MODE_SGBM_FGS_ONLINE_MUX0 = 0x4,
DPU_MODE_SGBM_FGS_ONLINE_MUX1 = 0x5,
DPU_MODE_SGBM_FGS_ONLINE_MUX2 = 0x6,
DPU_MODE_FGS_MUX0 = 0x7,
DPU_MODE_FGS_MUX1 = 0x8,
DPU_MODE_BUTT
}DPU_MODE_E;

```

## 【成员】

表 4.8: DPU\_MODE\_E 成员表

成员	描述
DPU_MODE_DEFAULT	DPU 的默认输出模式
DPU_MODE_SGBM_MUX0	该模式下，使用 SGBM 处理左图和右图，输出一张没有经过后处理的 8bit 视差图
DPU_MODE_SGBM_MUX1	该模式下，使用 SGBM 处理左图和右图，输出一张经过后处理的 16bit 视差图
DPU_MODE_SGBM_MUX2	该模式下，使用 SGBM 处理左图和右图，输出一张经过后处理的 8bit 视差图
DPU_MODE_SGBM_FGS_ONLINE_MUX0	该模式下，使用 SGBM、FGS 处理左图和右图，输出一张 8bit 视差图
DPU_MODE_SGBM_FGS_ONLINE_MUX1	该模式下，使用 SGBM、FGS 处理左图和右图，输出一张 16bit 深度图
DPU_MODE_SGBM_FGS_ONLINE_MUX2	该模式下，使用 SGBM 处理左图和右图，输出一张 16bit 深度图
DPU_MODE_FGS_MUX0	该模式下，使用 FGS 处理左图和右图，输出一张 8bit 视差图（也可用于图像的降噪，类似于引导滤波）
DPU_MODE_FGS_MUX1	该模式下，使用 FGS 处理左图和右图，输出一张 16bit 深度图
DPU_MODE_BUTT	枚举数值的最大值，用于判断输入的枚举值是否在范围内

## 【注意事项】

无

## 【相关类型及接口】



· DPU\_GRP\_ATTR\_S

# 5 应用程序接口

---

## 5.1 API 总览

- DPU 模块主要为用户提供以下 API:
  - CVI\_DPU\_CreateGrp : 创建一个 DPU GROUP
  - CVI\_DPU\_DestroyGrp : 销毁一个 DPU GROUP
  - CVI\_DPU\_SetGrpAttr : 配置 DPU GROUP 的属性
  - CVI\_DPU\_GetGrpAttr : 获取 DPU GROUP 的属性
  - CVI\_DPU\_StartGrp : 启用 DPU GROUP
  - CVI\_DPU\_StopGrp : 禁用 DPU GROUP
  - CVI\_DPU\_SetChnAttr : 设置 DPU CHN 属性
  - CVI\_DPU\_GetChnAttr : 获取 DPU CHN 属性
  - CVI\_DPU\_EnableChn : 启用 DPU channel
  - CVI\_DPU\_DisableChn : 禁用 DPU channel
  - CVI\_DPU\_SendFrame : 发送输入视频帧数据
  - CVI\_DPU\_GetFrame : 获取处理后的视频帧数据
  - CVI\_DPU\_ReleaseFrame : 释放缓冲区的视频帧数据

## 5.2 API 流程

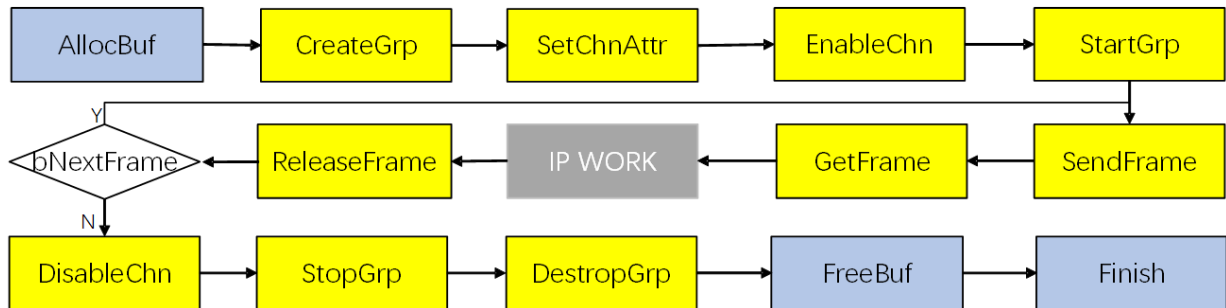


图 5.1: 应用层函数调用流程

在使用 DPU 处理进行视差/深度计算时，需要遵循以下流程：

1. 首先需要通过 VB 模块来预分配输入输出图像的 buffer，然后调用 CVI\_DPU\_CreateGrp 来创建一个新的任务 Group，同时需要 CVI\_DPU\_SetGrpAttr 设置 Group 的属性。接下来需要调用 CVI\_DPU\_EnableChn 来启动一个指定的 Channel，并调用 StartGrp 来开启当前 Group。
2. 之后通过 CVI\_DPU\_SendFrame 唤起内核线程发送输入的视频帧数据，通过 CVI\_DPU\_GetFrame 阻塞线程获取处理后的视频帧数据，最后通过 CVI\_DPU\_ReleaseFrame 释放缓冲区的视频帧数据。
3. 如果存在多帧数据，重复执行步骤 2 直到处理完全部输入数据。
4. 在 DPU 处理完图像后，需要通过 CVI\_DPU\_DisableChn 禁用 Channel，通过 CVI\_DPU\_StopGrp 停止 Group 任务，通过 CVI\_DPU\_DestroyGrp 销毁 Group，最后通过 VB 模块来释放预分配的 buffer。

## 5.3 API 说明

### 5.3.1 CVI\_DPU\_CreateGrp

#### 【描述】

创建一个 DPU GROUP。

#### 【语法】

```
CVI_S32 CVI_DPU_CreateGrp(DPU_GRP DpuGrp, const
DPU_GRP_ATTR_S *pstGrpAttr);
```

**【参数】**

表 5.1: CVI\_DPU\_CreateGrp 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
pstGrpAttr	DPU Group 属性, 该参数不能为空	输入

**【返回值】**

表 5.2: CVI\_DPU\_CreateGrp 返回值表

参数名称	描述
CVI_SUCCESS	创建 DPU GROUP 成功
其他值	创建 DPU GROUP 失败, 返回错误码。错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 不能重复调用该 API。如果需要重新创建 DPU GROUP, 需要先调用 CVI\_DPU\_DestroyGrp 销毁 DPU GROUP, 然后再调用该 API 创建 DPU GROUP。

**【举例】**

```
struct dpu_grp_attr *attr = (struct dpu_grp_attr *)kdata;  
ret = CVI_DPU_CreateGrp(attr->DpuGrp, &attr->stGrpAttr);
```

### 5.3.2 CVI\_DPU\_DestroyGrp

#### 【描述】

销毁一个 DPU Group。

#### 【语法】

```
CVI_S32 CVI_DPU_DestroyGrp(DPU_GRP DpuGrp);
```

#### 【参数】

表 5.3: CVI\_DPU\_DestroyGrp 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入

#### 【返回值】

表 5.4: CVI\_DPU\_DestroyGrp 返回值表

参数名称	描述
CVI_SUCCESS	销毁 DPU GROUP 成功
其他值	销毁 DPU GROUP 失败, 返回错误码。错误码定义参考表 1

#### 【需求】

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

#### 【注意】

- 调用该 API 之前, 需要先调用[CVI\\_DPU\\_CreateGrp](#) 创建 DPU GROUP。
- 调用该 API 销毁某一个 DPU Group 之前, 需要先调用[CVI\\_DPU\\_StopGrp](#) 停止该 DPU Group。
- 调用该 API 之后, 只有在当前 DPU Group 的任务完成后才能销毁该 DPU Group。

**【举例】**

```
struct dpu_grp_cfg *cfg = ((struct dpu_grp_cfg *)kdata);  
ret = CVI_DPU_DestroyGrp(cfg->DpuGrp);
```

### 5.3.3 CVI\_DPU\_SetGrpAttr

**【描述】**

配置 DPU GROUP 的属性。

**【语法】**

```
CVI_S32 CVI_DPU_SetGrpAttr(DPU_GRP DpuGrp,const  
DPU_GRP_ATTR_S *pstGrpAttr);
```

**【参数】**

表 5.5: CVI\_DPU\_SetGrpAttr 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
pstGrpAttr	DPU Group 属性, 该参数不能为空	输入

**【返回值】**

表 5.6: CVI\_DPU\_SetGrpAttr 返回值表

参数名称	描述
CVI_SUCCESS	配置 DPU GROUP 的属性成功
其他值	配置 DPU GROUP 的属性失败, 返回错误码。错误码定义参考表 1

**【需求】**

· 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前，需要先调用 `CVI_DPU_CreateGrp` 创建 DPU GROUP。
- GROUP 内的属性必须有效，并且其中某些静态属性不可被动态设置。详细信息参考 `DPU_GRP_ATTR_S`。

**【举例】**

```
struct dpu_grp_attr *cfg = (struct dpu_grp_attr *)kdata;  
DPU_GRP DpuGrp = cfg->DpuGrp;  
const DPU_GRP_ATTR_S *pstGrpAttr = &cfg->stGrpAttr;  
ret = CVI_DPU_SetGrpAttr(DpuGrp, pstGrpAttr);
```

### 5.3.4 CVI\_DPU\_GetGrpAttr

**【描述】**

获取 DPU GROUP 的属性。

**【语法】**

```
CVI_S32 CVI_DPU_GetGrpAttr(DPU_GRP DpuGrp, DPU_GRP_ATTR_S  
*pstGrpAttr);
```

**【参数】**

表 5.7: CVI\_DPU\_GetGrpAttr 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
pstGrpAttr	DPU Group 属性, 该参数不能为空	输入

**【返回值】**

表 5.8: CVI\_DPU\_GetGrpAttr 返回值表

参数名称	描述
CVI_SUCCESS	获取 DPU GROUP 的属性成功
其他值	获取 DPU GROUP 的属性失败, 返回错误码。错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用CVI\_DPU\_CreateGrp 创建 DPU GROUP。

**【举例】**

```
struct dpu_grp_attr *cfg = (struct dpu_grp_attr *)kdata;  
DPU_GRP DpuGrp = cfg->DpuGrp;  
DPU_GRP_ATTR_S *pstGrpAttr = &cfg->stGrpAttr;  
ret = CVI_DPU_GetGrpAttr(DpuGrp, pstGrpAttr);
```

### 5.3.5 CVI\_DPU\_StartGrp

**【描述】**

启用 DPU GROUP。

**【语法】**

```
CVI_S32 CVI_DPU_StartGrp(DPU_GRP DpuGrp);
```

**【参数】**



表 5.9: CVI\_DPU\_StartGrp 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入

**【返回值】**

表 5.10: CVI\_DPU\_StartGrp 返回值表

参数名称	描述
CVI_SUCCESS	启动 DPU GROUP 成功
其他值	启动 DPU GROUP 失败, 返回错误码。 错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用 `CVI_DPU_CreateGrp` 创建 DPU GROUP。
- 当重复调用该 API 时, 将返回一个表示成功的代码。

**【举例】**

```
struct dpu_grp_cfg *cfg = (struct dpu_grp_cfg *)kdata;  
ret = CVI_DPU_StartGrp(cfg->DpuGrp);
```

### 5.3.6 CVI\_DPU\_StopGrp

**【描述】**

禁用 DPU GROUP。

**【语法】**

```
CVI_S32 CVI_DPU_StopGrp(DPU_GRP DpuGrp);
```

**【参数】**

表 5.11: CVI\_DPU\_StopGrp 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入

**【返回值】**

表 5.12: CVI\_DPU\_StopGrp 返回值表

参数名称	描述
CVI_SUCCESS	停止 DPU GROUP 成功
其他值	停止 DPU GROUP 失败, 返回错误码。 错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用 CVI\_DPU\_CreateGrp 创建 DPU GROUP。
- 当重复调用该 API 时, 将返回一个表示成功的代码。

**【举例】**

```
struct dpu_grp_cfg *cfg = (struct dpu_grp_cfg *)kdata;  
ret = CVI_DPU_StopGrp(cfg->DpuGrp);
```

### 5.3.7 CVI\_DPU\_SetChnAttr

#### 【描述】

设置 DPU channel 属性。

#### 【语法】

```
CVI_S32 CVI_DPU_SetChnAttr(DPU_GRP DpuGrp, DPU_CHN DpuChn,  
const DPU_CHN_ATTR_S *pstChnAttr);
```

#### 【参数】

表 5.13: CVI\_DPU\_SetChnAttr 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
DpuChn	DPU channel 号, 取值范围: [0, DPU_MAX_CHN_NUM)	输入
pstChnAttr	Channel 属性, 该参数值不能为空	输入

#### 【返回值】

表 5.14: CVI\_DPU\_SetChnAttr 返回值表

参数名称	描述
CVI_SUCCESS	设置 DPU channel 属性成功
其他值	设置 DPU channel 属性失败, 返回错误码。错误码定义参考表 1

#### 【需求】

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

#### 【注意】

- 调用该 API 之前, 需要先调用 [CVI\\_DPU\\_CreateGrp](#) 创建 DPU GROUP。

## 【举例】

```
struct dpu_chn_attr *attr = (struct dpu_chn_attr *)kdata;
DPU_GRP DpuGrp = attr->DpuGrp;
DPU_CHN DpuChn = attr->DpuChn;
const DPU_CHN_ATTR_S *pstChnAttr = &attr->stChnAttr;
ret = CVI_DPU_SetChnAttr(DpuGrp, DpuChn, pstChnAttr);
```

### 5.3.8 CVI\_DPU\_GetChnAttr

## 【描述】

获取 DPU channel 属性。

## 【语法】

```
CVI_S32 CVI_DPU_GetChnAttr(DPU_GRP DpuGrp, DPU_CHN DpuChn,
DPU_CHN_ATTR_S *pstChnAttr);
```

## 【参数】

表 5.15: CVI\_DPU\_GetChnAttr 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
DpuChn	DPU channel 号, 取值范围: [0, DPU_MAX_CHN_NUM)	输入
pstChnAttr	Channel 属性, 该参数值不能为空	输入

## 【返回值】

表 5.16: CVI\_DPU\_GetChnAttr 返回值表

参数名称	描述
CVI_SUCCESS	获取 DPU channel 属性成功
其他值	获取 DPU channel 属性失败, 返回错误码。错误码定义参考 <a href="#">表 1</a>

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用CVI\_DPU\_CreateGrp 创建 DPU GROUP。

**【举例】**

```
struct dpu_chn_attr *attr = (struct dpu_chn_attr *)kdata;
DPU_GRP DpuGrp = attr->DpuGrp;
DPU_CHN DpuChn = attr->DpuChn;
DPU_CHN_ATTR_S *pstChnAttr = &attr->stChnAttr;
ret = CVI_DPU_GetChnAttr(DpuGrp, DpuChn, pstChnAttr);
```

### 5.3.9 CVI\_DPU\_EnableChn

**【描述】**

启用 DPU channel。

**【语法】**

```
CVI_S32 CVI_DPU_EnableChn(DPU_GRP DpuGrp, DPU_CHN DpuChn);
```

**【参数】**

表 5.17: CVI\_DPU\_EnableChn 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
DpuChn	DPU channel 号, 取值范围: [0, DPU_MAX_CHN_NUM)	输入

**【返回值】**

表 5.18: CVI\_DPU\_EnableChn 返回值表

参数名称	描述
CVI_SUCCESS	启用 DPU channel 成功
其他值	启用 DPU channel 属性失败, 返回错误码。错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用CVI\_DPU\_CreateGrp 创建 DPU GROUP。
- 当重复调用该 API 时, 将返回一个表示成功的代码。

**【举例】**

```
struct dpu_chn_cfg *cfg = (struct dpu_chn_cfg *)kdata;  
DPU_GRP DpuGrp = cfg->DpuGrp;  
DPU_CHN DpuChn = cfg->DpuChn;  
ret = CVI_DPU_EnableChn(DpuGrp, DpuChn);
```

### 5.3.10 CVI\_DPU\_DisableChn

**【描述】**

禁用 DPU channel。

**【语法】**

```
CVI_S32 CVI_DPU_DisableChn(DPU_GRP DpuGrp, DPU_CHN DpuChn);
```

**【参数】**

表 5.19: CVI\_DPU\_DisableChn 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
DpuChn	DPU channel 号, 取值范围: [0, DPU_MAX_CHN_NUM)	输入

**【返回值】**

表 5.20: CVI\_DPU\_DisableChn 返回值表

参数名称	描述
CVI_SUCCESS	禁用 DPU channel 成功
其他值	禁用 DPU channel 属性失败, 返回错误码。错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用CVI\_DPU\_CreateGrp 创建 DPU GROUP。
- 当重复调用该 API 时, 将返回一个表示成功的代码。

**【举例】**

```
struct dpu_chn_cfg *cfg = (struct dpu_chn_cfg *)kdata;  
ret = CVI_DPU_DisableChn(cfg->DpuGrp, cfg->DpuChn);
```

### 5.3.11 CVI\_DPU\_SendFrame

**【描述】**

发送输入视频帧数据。

**【语法】**

```
CVI_S32 CVI_DPU_SendFrame(DPU_GRP DpuGrp, const
VIDEO_FRAME_INFO_S *pst_left_frame, const VIDEO_FRAME_INFO_S
*pst_right_frame, CVI_S32 s32Millisec);
```

### 【参数】

表 5.21: CVI\_DPU\_SendFrame 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
pst_left_frame	左图数据, 该参数值不能为空。	输入
pst_right_frame	右图数据, 该参数值不能为空。	输入
s32Millisec	超时时间。	输入

表 5.22: 输入图像数据参数表

参数名称	支持的数据格式	数据对齐	支持的分辨率
pst_left_frame	Y only 8 bit	16 字节对齐	64 x 64 ~ 1920 x 1080
pst_right_frame	Y only 8 bit	16 字节对齐	64 x 64 ~ 1920 x 1080

### 【返回值】

表 5.23: CVI\_DPU\_SendFrame 返回值表

参数名称	描述
CVI_SUCCESS	发送图像数据成功
其他值	发送图像数据失败, 返回错误码。错误码定义参考表 1

### 【需求】

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

### 【注意】

- 调用该 API 之前, 需要先调用 CVI\_DPU\_CreateGrp 创建 DPU GROUP。
- 调用该 API 时, 可以控制帧率。



- pst\_left\_frame 与 pst\_left\_frame 的图像地址必须根据 VB 的要求进行 16 字节对齐。

**【举例】**

```
struct dpu_set_frame_cfg *cfg = (struct dpu_set_frame_cfg *)kdata;
DPU_GRP DpuGrp = cfg->DpuGrp;
const VIDEO_FRAME_INFO_S *pstSrcLeftFrame = &cfg->stSrcLeftFrame;
const VIDEO_FRAME_INFO_S *pstSrcRightFrame = &cfg->stSrcRightFrame;
CVI_S32 s32MilliSec = cfg->s32MilliSec;
ret = CVI_DPU_SendFrame(DpuGrp, pstSrcLeftFrame, pstSrcRightFrame, s32MilliSec);
```

### 5.3.12 CVI\_DPU\_GetFrame

**【描述】**

获取处理后的视频帧数据。

**【语法】**

```
CVI_S32 CVI_DPU_GetFrame(DPU_GRP DpuGrp, DPU_CHN DpuChn,
VIDEO_FRAME_INFO_S *pstFrameInfo, CVI_S32 s32Millisec);
```

**【参数】**

表 5.24: CVI\_DPU\_GetFrame 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
DpuChn	DPU channel 号, 取值范围: [0, DPU_MAX_CHN_NUM)	输入
pstFrameInfo	结果数据, 该参数值不能为空。	输出
s32Millisec	超时时间。	输入

表 5.25: 输出图像数据参数表

参数名称	支持的数据格式	数据对齐	支持的分辨率
pstFrameInfo	Y only 8 bit	16 字节对齐	64 x 64 ~ 1920 x 1080

**【返回值】**

表 5.26: CVI\_DPU\_GetFrame 返回值表

参数名称	描述
CVI_SUCCESS	获取处理后的视频帧数据成功
其他值	获取处理后的视频帧数据失败, 返回错误码。错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用CVI\_DPU\_CreateGrp 创建 DPU GROUP。
- 之后当 GROUP 属性的队列深度值不是 0 时, 才能获得输出图像。
- pstFrameInfo 的数据由系统保证。
- 输出的视差图使用输入的右侧图像作为参照图像。
- 输出图像的分辨率与输入的右图像的分辨率相同。

**【举例】**

```
struct dpu_get_frame_cfg *cfg = (struct dpu_get_frame_cfg *)kdata;  
DPU_GRP DpuGrp = cfg->DpuGrp;  
DPU_CHN DpuChn = cfg->DpuChn;  
VIDEO_FRAME_INFO_S *pstFrameInfo = &cfg->stFrameInfo;  
CVI_S32 s32MilliSec = cfg->s32MilliSec;  
ret = CVI_DPU_GetFrame(DpuGrp,DpuChn,pstFrameInfo, s32MilliSec);
```

### 5.3.13 CVI\_DPU\_ReleaseFrame

**【描述】**

释放缓冲区的视频帧数据。

**【语法】**

```
CVI_S32 CVI_DPU_ReleaseFrame(DPU_GRP DpuGrp, DPU_CHN DpuChn,  
const VIDEO_FRAME_INFO_S *pstVideoFrame);
```

**【参数】**

表 5.27: CVI\_DPU\_ReleaseFrame 参数表

参数名称	描述	输入/输出
DpuGrp	DPU Group 号, 取值范围: [0, DPU_MAX_GRP_NUM)	输入
DpuChn	DPU channel 号, 取值范围: [0, DPU_MAX_CHN_NUM)	输入
pstVideoFrame	结果数据, 该参数值不能为空。	输入

**【返回值】**

表 5.28: CVI\_DPU\_ReleaseFrame 返回值表

参数名称	描述
CVI_SUCCESS	释放缓冲区的视频帧数据成功
其他值	释放缓冲区的视频帧数据失败, 返回错误码。错误码定义参考表 1

**【需求】**

- 头文件: cvi\_dpu.h, cvi\_debug.h, cvi\_base.h

**【注意】**

- 调用该 API 之前, 需要先调用CVI\_DPU\_CreateGrp 创建 DPU GROUP。
- pstVideoFrame 的值来自CVI\_DPU\_GetFrame , 且参数与CVI\_DPU\_GetFrame 配套使用。

**【举例】**

```
struct dpu_release_frame_cfg *cfg = (struct dpu_release_frame_cfg *)kdata;  
DPU_GRP DpuGrp = cfg->DpuGrp;  
DPU_CHN DpuChn = cfg->DpuChn;  
VIDEO_FRAME_INFO_S *pstFrameInfo = &cfg->stFrameInfo;  
ret = CVI_DPU_ReleaseFrame(DpuGrp, DpuChn, pstFrameInfo);
```

# 6 错误码

---

## 6.1 错误码

表 1 列出了 DPU 模块的 API 错误代码。

表 6.1: 表 1 DPU 模块的 API 错误代码表

错误码	宏定义	描述
0xc0228001	CVI_ERR_DPU_INVALID_DEVID	不合法的 device ID
0xc0228002	CVI_ERR_DPU_INVALID_CHNID	不合法的 channel ID
0xc0228003	CVI_ERR_DPU_ILLEGAL_PARAM	至少存在一个不合法的参数或不合法的枚举值
0xc0228004	CVI_ERR_DPU_EXIST	资源存在
0xc0228005	CVI_ERR_DPU_UNEXIST	资源不存在
0xc0228006	CVI_ERR_DPU_NULL_PTR	使用了空指针
0xc0228007	CVI_ERR_DPU_NOT_CONFIG	在配置属性之前尝试启用、初始化系统、设备或通道
0xc0228008	CVI_ERR_DPU_NOT_SUPPORT	操作或类型在当前情况下不支持
0xc0228009	CVI_ERR_DPU_NOT_PERM	操作不被允许, 例如尝试修改静态属性
0xc022800c	CVI_ERR_DPU_NOMEM	由于内存分配引起的失败
0xc022800d	CVI_ERR_DPU_NOBUF	由于缓冲区分配引起的失败
0xc022800e	CVI_ERR_DPU_BUF_EMPTY	缓冲区为空
0xc022800f	CVI_ERR_DPU_BUF_FULL	缓冲区已满
0xc0228010	CVI_ERR_DPU_NOTREADY	系统尚未准备就绪, 可能尚未初始化或加载, 打开设备文件失败
0xc0228011	CVI_ERR_DPU_ERR_BADADDR	地址错误
0xc0228012	CVI_ERR_DPU_BUSY	资源繁忙