



ISP 模块开发指南

版本号: 1.0

发布日期: 2020.10.20

版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.10.20	AWA1817	version 1.0



目 录

1	ISP 功能概述	1
1.1	ISP 简介	1
1.2	ISP Server 简介	1
1.2.1	ISP Server 调用参考	2
1.2.2	ISP Server 效果配置	5
1.2.3	ISP Server 调试	6
1.3	ISP tool 简介	6
1.4	ISP 第三方 3A 算法库	7
1.5	ISP 算法库模块简介	7
1.5.1	AE 简介	7
1.5.2	AWB 简介	8
1.5.3	AF 简介	8
1.5.4	宽动态	9
1.5.5	降噪	9
2	ISP 效果接口	10
2.1	标准命令接口	10
2.1.1	亮度设置	12
2.1.2	对比度设置	13
2.1.3	饱和度设置	13
2.1.4	自动白平衡开关设置	13
2.1.5	曝光行设置	13
2.1.6	曝光模式设置	14
2.1.7	曝光时间设置	14
2.1.8	自动增益开关设置	14
2.1.9	增益设置	15
2.1.10	水平镜像设置	15
2.1.11	垂直镜像设置	16
2.1.12	flicker 模式设置	16
2.1.13	flicker 开关设置	16
2.1.14	曝光偏移设置	17
2.1.15	白平衡场景预设设置	17
2.1.16	感光度设置	17
2.1.17	感光度开关设置	18
2.1.18	测光模式设置	18
2.1.19	色调设置	19
2.2	自定义命令接口	19
2.2.1	获取 ISP 数字增益	20
2.2.2	设置 2D 降噪强度	20
2.2.3	设置 3D 降噪强度	20
2.2.4	设置面光程度	20

2.2.5 设置背光程度	21
2.2.6 设置手动白平衡增益	21
2.2.7 设置 AE 模拟和数字增益范围	21
2.2.8 设置滤镜特效	22
2.3 效果文件修改接口	23
2.3.1 ISP 模块开关	24
2.3.2 AE table 设置	24
2.3.3 AE 权重表设置	25
3 统计值获取	26



1 ISP 功能概述

1.1 ISP 简介

ISP 模块主要用于处理 image sensor 输出的 RAW 数据，其主要功能包括：黑电平校正、坏点校正、镜头阴影、2D/3D 降噪、色彩增强、3A 等。ISP 算法包含硬件算法和软件算法库两部分：硬件算法集成在 SOC 上，称为 hawkview ISP；软件算法服务于 ISP 硬件算法，故称为 ISP server。

1.2 ISP Server 简介

ISP Server 模块主要包括 ISP 算法库和 ISP 中间件部分：

ISP 算法库部分，其主要用于在 ISP 运行时图像效果的处理，包括 3A 算法以及一系列 ISP 正常运行所需的基本算法；

ISP 中间件部分，其主要用于控制 ISP 以及 Sensor 驱动、响应 Camera 应用以及 Tuning 工具命令、调度 ISP 相关算法等，包括事件管理、Pipeline 管理、Buffer 管理以及算法调度等模块。

ISP 算法库、中间件、驱动以及 Camera 应用相互关系如下图所示：

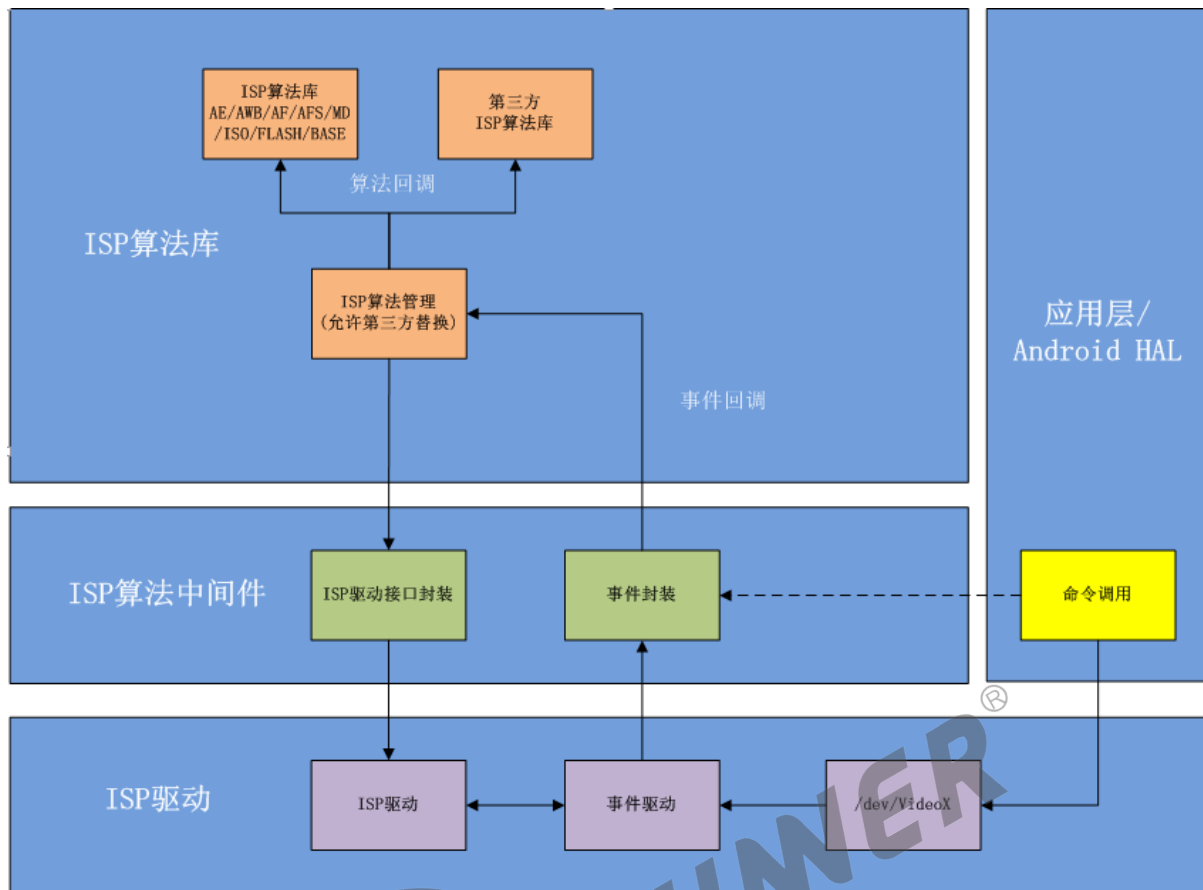


图 1-1: ISP Server 基本框架

1.2.1 ISP Server 调用参考

ISP 算法库调用示例如下，示例 1 为 ISP 库与抓图应用，单独编译成一个 isp server 应用程序：

```

int main(int argc __attribute__((__unused__)), char *argv[] __attribute__((__unused__)))
{
    int isp_id = 0, isp_num = 2;

    if (argc == 1) {
        isp_id = 0;
    } else if (argc == 2) {
        isp_id = atoi(argv[1]);
    } else {
        printf("please select the isp_id: 0~1 .....\\n");
        scanf("%d", &isp_id);
    }

    media_dev_init();

    if (isp_id >= 2) {
        for (isp_id = 0; isp_id < isp_num; isp_id++) {
            isp_init(isp_id);
            isp_run(isp_id);
        }
    }
}

```

```
    }

    for (isp_id = 0; isp_id < isp_num; isp_id++) {
        isp_pthread_join(isp_id);
        isp_exit(isp_id);
    }
} else {
    isp_init(isp_id);
    isp_run(isp_id);
    isp_pthread_join(isp_id);
    isp_exit(isp_id);
}

media_dev_exit();

return 0;
}
```

示例 2 为 ISP 库与抓图程序一起调用编译成一个统一的应用程序：

```
static void isp_server_start(int isp_id)
{
    /*
     * media_open must after AW_MPI_VI_InitCh, otherwise, media_pipeline_get_head
     * will not get sensor entity, and then lead to failure of calling
     * isp_sensor_get_configs,
     * because of sensor link will not enable until s_input is called.
     */

    isp_init(isp_id);
    isp_run(isp_id);
}

static void isp_server_stop(int isp_id)
{
    isp_stop(isp_id);
    isp_pthread_join(isp_id);
    isp_exit(isp_id);
}

static void isp_server_wait_to_exit(int isp_id)
{
    isp_pthread_join(isp_id);
    isp_exit(isp_id);
}

int main_test(int ch_num, int width, int height, int out_fmt)
{
    pthread_t thid[MAX_VIDEO_NUM];
    int ret, i, ch = -1;
    struct vi_info privCap[MAX_VIDEO_NUM];

    if (media == NULL) {
        media = isp_md_open(MEDIA_DEVICE);
        if (media == NULL) {
            ISP_PRINT("unable to open media device %s\n", MEDIA_DEVICE);
            return -1;
        }
    } else {
```

```
ISP_PRINT("mpi_vi already init\n");
}

media_dev_init();

if (ch_num > MAX_VIDEO_NUM)
    ch_num = MAX_VIDEO_NUM;

if (ch_num == 0 || ch_num == 1) {
    ch = ch_num;
    ch_num = 2;
}

for(i = 0; i < ch_num; i++) {
    if (i != ch && ch != -1)
        continue;
    memset(&privCap[i], 0, sizeof(struct vi_info));
    /*Set Dev ID and Chn ID*/
    privCap[i].Chn = i;
    privCap[i].s32MilliSec = 2000;

    privCap[i].vfmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
    privCap[i].vfmt.memtype = V4L2_MEMORY_MMAP;
    switch (out_fmt) {
        case 0: privCap[i].vfmt.format.pixelformat = V4L2_PIX_FMT_SBGGR8; break;
        case 1: privCap[i].vfmt.format.pixelformat = V4L2_PIX_FMT_YUV420M; break;
        case 2: privCap[i].vfmt.format.pixelformat = V4L2_PIX_FMT_YUV420; break;
        case 3: privCap[i].vfmt.format.pixelformat = V4L2_PIX_FMT_NV12M; break;
        default: privCap[i].vfmt.format.pixelformat = V4L2_PIX_FMT_YUV420M; break;
    }
    privCap[i].vfmt.format.field = V4L2_FIELD_NONE;
    privCap[i].vfmt.format.width = width;
    privCap[i].vfmt.format.height = height;
    privCap[i].vfmt.nbufs = 8;
    privCap[i].vfmt.nplanes = 3;
    privCap[i].vfmt.fps = fps;
    privCap[i].vfmt.capturemode = V4L2_MODE_VIDEO;
    privCap[i].vfmt.use_current_win = 0;
    privCap[i].vfmt.wdr_mode = wdr_mode;

    if (isp_video_open(media, i) < 0) {
        printf("isp_video_open vi%d failed\n", i);
        return -1;
    }

    if (video_set_fmt(media->video_dev[i], &privCap[i].vfmt) < 0) {
        printf("video_set_fmt failed\n");
        return -1;
    }

    video_get_fmt(media->video_dev[i], &privCap[i].vfmt);

    privCap[i].isp_id = video_to_isp_id(media->video_dev[i]);
    if (privCap[i].isp_id == -1)
        continue;

    /*Call Video Thread*/
    ret = pthread_create(&thid[i], NULL, loop_cap, (void *)&privCap[i]);
    if (ret < 0) {
        printf("pthread_create loop_cap Chn[%d] failed.\n", i);
    }
}
```



```

        continue;
    }

    /*Call isp server*/
    printf("isp%d server start!!!\n", privCap[i].isp_id);
    isp_server_start(privCap[i].isp_id);
}

for(i = 0; i < ch_num; i++) {
    if (i != ch && ch != -1)
        continue;
    printf("isp%d server wait to exit!!!\n", privCap[i].isp_id);
    isp_server_wait_to_exit(privCap[i].isp_id);
}

media_dev_exit();

for(i = 0; i < ch_num; i++) {
    if (i != ch && ch != -1)
        continue;
    printf("video%d wait to exit!!!\n", i);
    pthread_join(thid[i], NULL);
}

for(i = 0; i < ch_num; i++) {
    if (i != ch && ch != -1)
        continue;
    isp_video_close(media, i);
}

if (media)
    isp_md_close(media);
media = NULL;

return 0;
}

```

1.2.2 ISP Server 效果配置

- 将 ISP 效果文件xxx.h文件拷贝至\isp_cfg\SENSOR_H目录下;
- 在\isp_cfg\isp_ini_parse.c文件中配置 xxx.h;
 - 在isp_ini_parse.c中加入 xxx.h 头文件;
 - 在struct isp_cfg_array cfg_arr[]中加入相应配置文件，如下：

```

struct isp_cfg_array cfg_arr[] = {
    {"imx317_mipi", "imx317_default_ini_4v5", 3840, 2160, 30, 0, 0, &imx317_default_ini_4v5},
    {"imx317_mipi", "imx317_default_ini_4v5", 1920, 1080, 30, 0, 0, &imx317_default_ini_4v5},
    {"imx317_mipi", "imx317_wdr_ini_4v5", 1920, 1080, 30, 1, 0, &imx317_wdr_ini_4v5},
    {"imx274_slvds", "imx274_default_ini_4v5", 1920, 1080, 60, 0, 0, &imx274_default_ini_4v5},
    {"imx274_slvds", "imx274_wdr_ini_4v5", 1920, 1080, 60, 1, 0, &imx274_wdr_ini_4v5},
    {"imx278", "imx278_full_v5", 4208, 3120, 20, 0, 0, &imx278_full_v5},
}

```

```

{"imx278", "imx278_full_v5", 4160, 3120, 20, 0, 0, &imx278_full_v5},
{"imx278", "imx278_4k_v5", 4208, 2368, 30, 0, 0, &imx278_4k_v5},
{"imx278", "imx278_60fps_v5", 2048, 1152, 60, 0, 0, &imx278_60fps_v5},
{"imx278", "imx278_120fps_v5", 1280, 720, 120, 0, 0, &imx278_120fps_v5},
};

```

- 第一个参数 “imx317_mipi” 为 sensor name，和 sensor 驱动文件中 sensor_name 保持一致；
- 第二个参数 “imx317_default_ini_4v5” 为 isp 配置文件名称；
- 第三、四个参数 3840、2160 分别为图像宽和高；
- 第五个参数 30 为 sensor 帧率；
- 第六个参数 0 为 WDR 模式:0 为普通模式，1 为 WDR DOL 模式，2 为 WDR command 模式；
- 第七个参数表示 IR 模式，白天参数为 0，黑夜参数为 1；
- 第八个参数 &imx317_default_ini_4v5 表示配置文件 xxx.h 中 isp_cfg_pt 结构体名称；

1.2.3 ISP Server 调试

ISP server 中各个模块的 log 开关 id 如下（详情请参考 libisp 中的 isp_debug.h）：

```

#define ISP_LOG_AE          (1 << 0)    //0x1
#define ISP_LOG_AWB         (1 << 1)    //0x2
#define ISP_LOG_AF          (1 << 2)    //0x4
#define ISP_LOG_ISO         (1 << 3)    //0x8
#define ISP_LOG_GAMMA       (1 << 4)    //0x10
#define ISP_LOG_COLOR_MATRIX (1 << 5)    //0x20
#define ISP_LOG_AFS         (1 << 6)    //0x40
#define ISP_LOG_MOTION_DETECT (1 << 7)    //0x80
#define ISP_LOG_GAIN_OFFSET (1 << 8)    //0x100
#define ISP_LOG_DEF0G       (1 << 9)    //0x200
#define ISP_LOG_LSC         (1 << 10)   //0x400
#define ISP_LOG_GTM         (1 << 11)   //0x800
#define ISP_LOG_PLTM        (1 << 12)   //0x1000

#define ISP_LOG_SUBDEV       (1 << 13)   //0x2000
#define ISP_LOG_CFG         (1 << 14)   //0x4000
#define ISP_LOG_VIDEO       (1 << 15)   //0x8000
#define ISP_LOG_ISP         (1 << 16)   //0x10000
#define ISP_LOG_FLASH       (1 << 17)   //0x20000

```

如果要开启某个模块的调试信息，需要在效果头文件中将 isp_log_param 修改为对应的 id，或者多个模块的或值，修改完成后重新编译 ISP server 即可。

1.3 ISP tool 简介

Hawkview ISP 调试工具可以通过局域网络连接单板在线调试 ISP 各个模块的参数，使用标定分析工具进行各类数据分析，使用 rtsp 工具实时预览图像效果等。使用 ISP 调试工具时，需

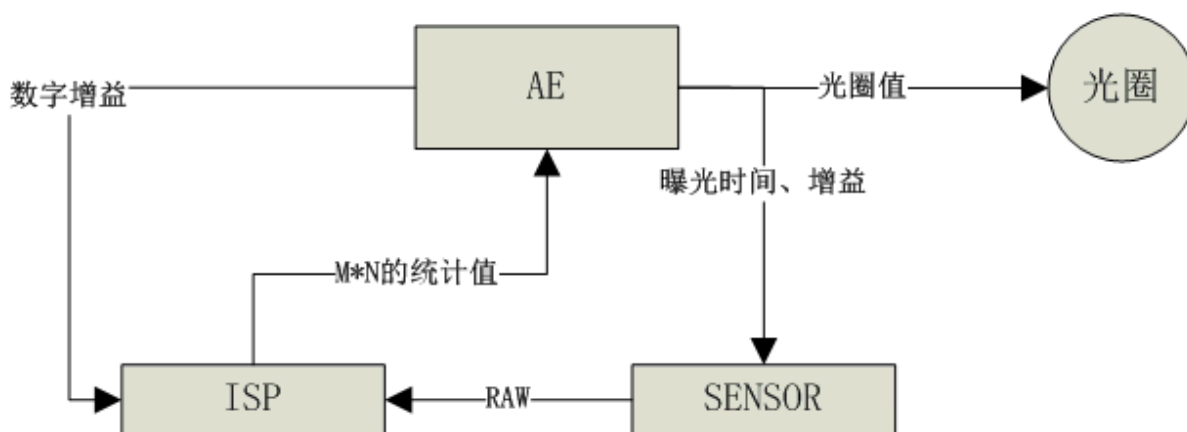


图 1-3: AE 模块流程

1.5.2 AWB 简介

Hawkview ISP AWB 实现的功能是：通过 ISP 获取的统计信息计算出 Rgain 和 Bgain，并与调试工具测量的色温曲线进行对比得到对应的色温以及当前色温下的 RGB 各自的增益，从而获取标准日光下的图像颜色。AWB 算法支持色调的偏好设置，支持蓝天、草地、肤色等特殊场景的处理。

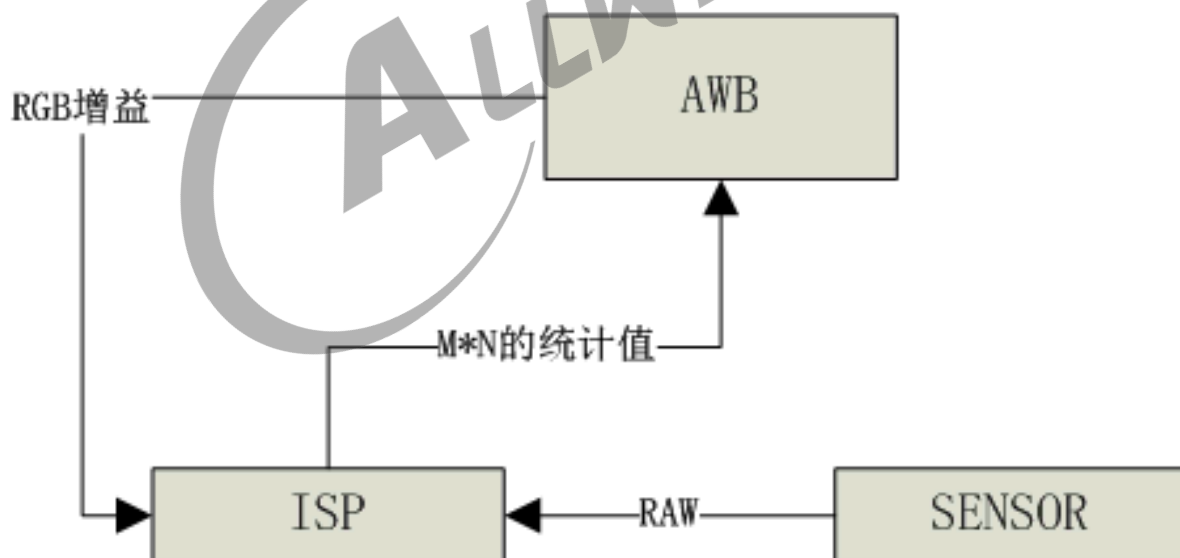


图 1-4: AWB 模块流程

1.5.3 AF 简介

Hawkview ISP AF 实现的功能是：通过 ISP 获取的统计信息计算出 focus value，并通过搜索算法找到 focus value 最大的位置，即为最佳对焦位置，从而获取最清晰的图像。AF 算法支

持场景变换自动对焦和点对焦。

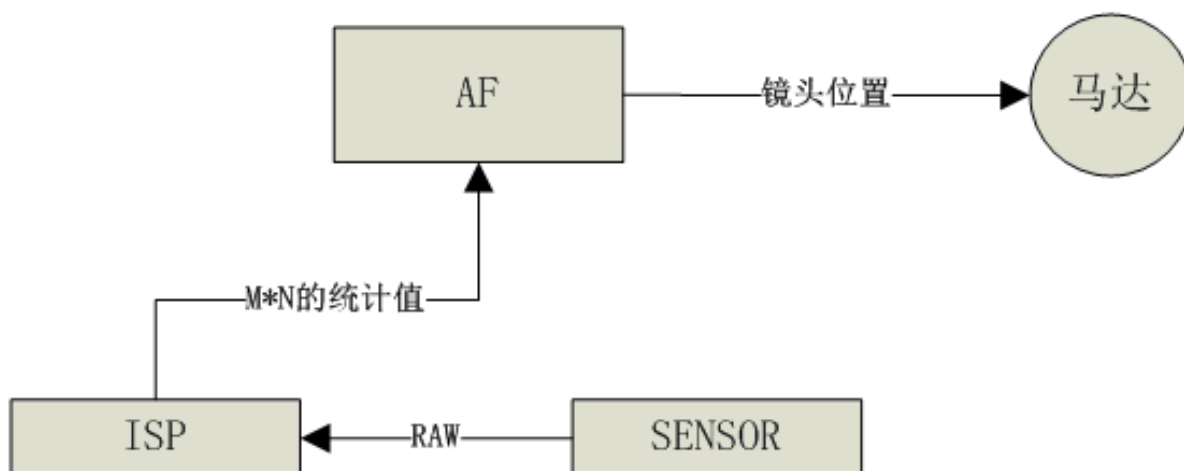


图 1-5: AF 模块流程

1.5.4 宽动态

Hawkview ISP 的宽动态包括两个方面：两帧合成 WDR 以及 PLTM（数字宽动态）。前者是支持 sensor 的 WDR 模式，后者是 ISP 硬件算法实现的数字宽动态。数字宽动态算法在提高暗部细节的同时保留了亮部细节，可以根据场景亮度自适应的调节强度。宽动态算法使得用户在面光和背光场景下也能获取高质量的图像。

1.5.5 降噪

Hawkview ISP 的宽动态包括两个方面：2DNR 和 3DNR。2DNR 在空域进行，3DNR 在空域和时域进行，去噪算法在降低噪声的同时保持边缘和纹理。目前降噪算法支持不同 ISO（增益）等级下降噪强度的自适应调整。

2 ISP 效果接口

ISP 效果接口分为三种：标准命令接口、自定义命令接口以及效果文件修改接口。标准命令接口采用 V4L2 定义的标准命令来进行效果的设置，自定义命令接口采用自定义命令来进行效果设置，这两类接口主要实现比较简单的单一值传递的强度类设置，例如亮度、对比度、降噪强度等设置。效果文件修改接口可以跟 ISP 工具一样直接修改 ISP 效果文件，可以实现更加复杂的大批参数修改的需求，例如修改 AE table 等。

2.1 标准命令接口

ISP 标准命令接口是基于 V4L2 的 s_ctrl 框架来实现的。用户可以调用 `ioctl(fd, VIDIOC_S_CTRL, &ctrl)`；接口来设置实现自己需要的效果。使用实例如下：

- 设置命令参数值：

```
int v4l2_set_control(struct media_entity *entity, unsigned int id, int32_t *value)
{
    struct v4l2_control ctrl;
    int ret;

    if (entity->fd == -1)
        return -1;

    ctrl.id = id;
    ctrl.value = *value;

    ret = ioctl(entity->fd, VIDIOC_S_CTRL, &ctrl);
    if (ret < 0) {
        ISP_ERR("unable to set control: %s (%d).\n",
                strerror(errno), errno);
        return -errno;
    }

    *value = ctrl.value;
    return 0;
}
```

- 获取命令当前值：

```
int v4l2_get_control(struct media_entity *entity, unsigned int id, int32_t *value)
{

```

```

    struct v4l2_control ctrl;
    int ret;

    if (entity->fd == -1)
        return -1;

    ctrl.id = id;

    ret = ioctl(entity->fd, VIDIOC_G_CTRL, &ctrl);
    if (ret < 0) {
        ISP_ERR("unable to get control: %s (%d).\n", strerror(errno), errno);
        return -errno;
    }

    *value = ctrl.value;
    return 0;
}

```

- 查询命令取值范围：

```

int v4l2_query_control(struct media_entity *entity, unsigned int id, struct v4l2_queryctrl
    *ctrl)
{
    int ret;

    if (entity->fd == -1)
        return -1;

    ctrl->id = id;

    ret = ioctl(entity->fd, VIDIOC_QUERYCTRL, ctrl);
    if (ret < 0) {
        ISP_ERR("unable to query control: %s (%d).\n", strerror(errno), errno);
        return -errno;
    }
    return 0;
}

```

- 相关数据结构的变量意义如下：

```

struct v4l2_control {
    __u32      id;
    __s32      value;
};
id:命令号,如: V4L2_CID_BRIGHTNESS、V4L2_CID_CONTRAST等
value:对应id的value值

```

```

struct v4l2_queryctrl {
    __u32      id;
    __u32      type; /* enum v4l2_ctrl_type */
    __u8       name[32]; /* Whatever */
    __s32      minimum; /* Note signedness */
    __s32      maximum;
    __s32      step;
}

```

```

__s32          default_value;
__u32          flags;
__u32          reserved[2];
};
id:命令号如: V4L2_CID_BRIGHTNESS、V4L2_CID_CONTRAST等
type: 命令类型, 如:
enum v4l2_ctrl_type {
    V4L2_CTRL_TYPE_INTEGER      = 1,
    V4L2_CTRL_TYPE_BOOLEAN      = 2,
    V4L2_CTRL_TYPE_MENU         = 3,
    V4L2_CTRL_TYPE_BUTTON       = 4,
    V4L2_CTRL_TYPE_INTEGER64    = 5,
    V4L2_CTRL_TYPE_CTRL_CLASS   = 6,
    V4L2_CTRL_TYPE_STRING       = 7,
    V4L2_CTRL_TYPE_BITMASK      = 8,
    V4L2_CTRL_TYPE_INTEGER_MENU = 9,
};
name: 命令名称
Minimum: 命令最小值
Maximum: 命令最大值
Step: 命令每次改变的步长
Flag: 可读写标志

```

- 目前 ISP 支持的 s_ctrl 命令 id 如下:

```

#define V4L2_CID_BRIGHTNESS    (V4L2_CID_BASE+0)
#define V4L2_CID_CONTRAST      (V4L2_CID_BASE+1)
#define V4L2_CID_SATURATION    (V4L2_CID_BASE+2)
#define V4L2_CID_HUE           (V4L2_CID_BASE+3)
#define V4L2_CID_AUTO_WHITE_BALANCE (V4L2_CID_BASE+12)
#define V4L2_CID_EXPOSURE      (V4L2_CID_BASE+17)
#define V4L2_CID_AUTOGAIN      (V4L2_CID_BASE+18)
#define V4L2_CID_GAIN          (V4L2_CID_BASE+19)
#define V4L2_CID_HFLIP         (V4L2_CID_BASE+20)
#define V4L2_CID_VFLIP         (V4L2_CID_BASE+21)
#define V4L2_CID_POWER_LINE_FREQUENCY (V4L2_CID_BASE+24)
#define V4L2_CID_BAND_STOP_FILTER (V4L2_CID_BASE+33)
#define V4L2_CID_EXPOSURE_AUTO (V4L2_CID_CAMERA_CLASS_BASE+1)
#define V4L2_CID_EXPOSURE_ABSOLUTE (V4L2_CID_CAMERA_CLASS_BASE+2)
#define V4L2_CID_AUTO_EXPOSURE_BIAS (V4L2_CID_CAMERA_CLASS_BASE+19)
#define V4L2_CID_AUTO_N_PRESET_WHITE_BALANCE (V4L2_CID_CAMERA_CLASS_BASE+20)
#define V4L2_CID_ISO_SENSITIVITY (V4L2_CID_CAMERA_CLASS_BASE+23)
#define V4L2_CID_ISO_SENSITIVITY_AUTO (V4L2_CID_CAMERA_CLASS_BASE+24)
#define V4L2_CID_EXPOSURE_METERING (V4L2_CID_CAMERA_CLASS_BASE+25)

```

2.1.1 亮度设置

属性	值
命令 id	V4L2_CID_BRIGHTNESS
Value 取值范围	[-64, 64]
默认值	0
步长	1

2.1.2 对比度设置

属性	值
命令 id	V4L2_CID_CONTRAST
Value 取值范围	[-64, 64]
默认值	0
步长	1

2.1.3 饱和度设置

属性	值
命令 id	V4L2_CID_SATURATION
Value 取值范围	[-128, 128]
默认值	0
步长	1

2.1.4 自动白平衡开关设置

属性	值
命令 id	V4L2_CID_AUTO_WHITE_BALANCE
Value 取值范围	[0, 1]
默认值	0
步长	1

注意事项：该接口设置为 0 时需要手动设置白平衡增益。

2.1.5 曝光行设置

属性	值
命令 id	V4L2_CID_EXPOSURE
Value 取值范围	[1, 65536*16]
默认值	1
步长	1

注意事项：该接口需要在手动曝光时使用，与 V4L2_CID_EXPOSURE_ABSOLUTE 功能一样；该接口设置的是 sensor 的曝光行，16 为 1 行精度 1/16 行，越大曝光时间越长。可以通过 g_ctrl 获取 sensor 当前正在使用的曝光行（曝光行和曝光时间可以相互转化，sensor 驱动实际使用的是曝光行）。

2.1.6 曝光模式设置

属性	值
命令 id	V4L2_CID_EXPOSURE_AUTO
Value 取值范围	[0, 3], 依次为：自动曝光、手动曝光、快门优先、光圈优先，详见 enum v4l2_exposure_auto_type
默认值	0
步长	1

注意事项：在手动曝光或者快门优先时使用，需要调用 V4L2_CID_EXPOSURE_ABSOLUTE 设置曝光行。

```
enum v4l2_exposure_auto_type {
    V4L2_EXPOSURE_AUTO = 0,
    V4L2_EXPOSURE_MANUAL = 1,
    V4L2_EXPOSURE_SHUTTER_PRIORITY = 2,
    V4L2_EXPOSURE_APERTURE_PRIORITY = 3
};
```

2.1.7 曝光时间设置

属性	值
命令 id	V4L2_CID_EXPOSURE_ABSOLUTE
Value 取值范围	[1, 30*1000000]
默认值	0
步长	1

注意事项：该接口需要在手动曝光或者快门优先时使用，与 V4L2_CID_EXPOSURE 功能一样；该接口设置的是 sensor 的曝光时间，单位为 us，越大曝光时间越长。可以通过 g_ctrl 获取 sensor 当前正在使用的曝光时间（由曝光行转换得到）。

2.1.8 自动增益开关设置

属性	值
命令 id	V4L2_CID_AUTOGAIN
Value 取值范围	[0, 1]
默认值	0
步长	1

注意事项：1、该接口设置为 0 时需要调用 V4L2_CID_GAIN 设置 sensor 增益。2、不能与 V4L2_CID_ISO_SENSITIVITY_AUTO 同时使用。3、手动曝光模式时，必须调用该接口设置为手动增益模式。4、自动曝光模式时，调用该接口设置 0 时相当于增益优先模式。算法优先使用增益，其次使用曝光时间来提高亮度。

2.1.9 增益设置

属性	值
命令 id	V4L2_CID_AUTOGAIN
Value 取值范围	[16, 6000*16]
默认值	16
步长	1

注意事项：16 表示一倍增益，当设置手动增益时需要调用此接口设置 sensor 的增益。可以先获取 sensor 当前正在使用的增益值，该增益为 sensor_again 和 sensor_dgain 的乘积，为了简化，将 sensor_again* sensor_dgain 统称为 again，将 isp_dgain 称为 dgain。

2.1.10 水平镜像设置

属性	值
命令 id	V4L2_CID_HFLIP
Value 取值范围	[0, 1]
默认值	0
步长	1

注意事项：1、使用内部 flip 时，在 S_INPUT 接口之后调用；2、如果是使用 sensor 内部 flip 时，则需要在 STREAM_ON 后调用。

2.1.11 垂直镜像设置

属性	值
命令 id	V4L2_CID_VFLIP
Value 取值范围	[0, 1]
默认值	0
步长	1

注意事项：1、使用内部 flip 时，在 S_INPUT 接口之后调用；2、如果是使用 sensor 内部 flip 时，则需要在 STREAM_ON 后调用。

2.1.12 flicker 模式设置

属性	值
命令 id	V4L2_CID_POWER_LINE_FREQUENCY
Value 取值范围	[0, 3], 详见 enum v4l2_power_line_frequency
默认值	3
步长	1

注意事项：无。

```
enum v4l2_power_line_frequency {  
    V4L2_CID_POWER_LINE_FREQUENCY_DISABLED = 0,  
    V4L2_CID_POWER_LINE_FREQUENCY_50HZ = 1,  
    V4L2_CID_POWER_LINE_FREQUENCY_60HZ = 2,  
    V4L2_CID_POWER_LINE_FREQUENCY_AUTO = 3,  
};
```

2.1.13 flicker 开关设置

属性	值
命令 id	V4L2_CID_BAND_STOP_FILTER
Value 取值范围	[0, 1]
默认值	1
步长	1

注意事项：相当于 flicker 模式的 DISABLED 和 AUTO。

2.1.14 曝光偏移设置

属性	值
命令 id	V4L2_CID_AUTO_EXPOSURE_BIAS
Value 取值范围	数组 exp_bias_qmenu[] = { -4, -3, -2, -1, 0, 1, 2, 3, 4, } 的 index
默认值	4
步长	1

注意事项：接口设置的是数组的 index，例如 index = 4，对应的曝光偏移为 0，该接口改变的是曝光后的整体亮度。

2.1.15 白平衡场景预设设置

属性	值
命令 id	V4L2_CID_AUTO_N_PRESET_WHITE_BALANCE
Value 取值范围	[0, 9], 详见 enum v4l2_auto_n_preset_white_balance
默认值	1
步长	1

注意事项：无。

```
enum v4l2_auto_n_preset_white_balance {
    V4L2_WHITE_BALANCE_MANUAL      = 0,
    V4L2_WHITE_BALANCE_AUTO        = 1,
    V4L2_WHITE_BALANCE_INCANDESCENT = 2,
    V4L2_WHITE_BALANCE_FLUORESCENT  = 3,
    V4L2_WHITE_BALANCE_FLUORESCENT_H = 4,
    V4L2_WHITE_BALANCE_HORIZON     = 5,
    V4L2_WHITE_BALANCE_DAYLIGHT     = 6,
    V4L2_WHITE_BALANCE_FLASH       = 7,
    V4L2_WHITE_BALANCE_CLOUDY      = 8,
    V4L2_WHITE_BALANCE_SHADE       = 9,
};
```

2.1.16 感光度设置

属性	值
命令 id	V4L2_CID_ISO_SENSITIVITY
Value 取值范围	数组 iso_qmenu[] = { 100, 200, 400, 800, 1600, 3200, 6400, } 的 index

属性	值
默认值	2
步长	1

注意事项：Gain = $2^{\text{iso_index}}$; 即：ISO100 = 1x/ ISO200 = 2x/ ISO400 = 4x/ ISO800 = 8x...

2.1.17 感光度开关设置

属性	值
命令 id	V4L2_CID_ISO_SENSITIVITY_AUTO
Value 取值范围	[0, 1]
默认值	1
步长	1

注意事项：1、在自动曝光模式下调用该接口设置 0 时，相当于开启 ISO 优先模式，此时需要调用 V4L2_CID_ISO_SENSITIVITY 设置感光度，算法会映射对应的增益。2、ISO 优先模式和增益优先模式对算法库来说是同一个功能，两者不能同时使用，不同的是一个需要设置感光度一个需要设置增益。

2.1.18 测光模式设置

属性	值
命令 id	V4L2_CID_EXPOSURE_METERING
Value 取值范围	[0, 3], 详见 enum v4l2_exposure_metering
默认值	0
步长	1

注意事项：无。

```
enum v4l2_exposure_metering {
    V4L2_EXPOSURE_METERING_AVERAGE      = 0,
    V4L2_EXPOSURE_METERING_CENTER_WEIGHTED = 1,
    V4L2_EXPOSURE_METERING_SPOT          = 2,
    V4L2_EXPOSURE_METERING_MATRIX         = 3,
};
```

2.1.19 色调设置

属性	值
命令 id	V4L2_CID_HUE
Value 取值范围	[-180, 180]
默认值	0
步长	1

注意事项：无。

2.2 自定义命令接口

isp_set/get_attr_cfg

- 目的

设置/获取 ISP 参数配置

- 语法

```
HW_S32 isp_set_attr_cfg (int dev_id, HW_U32 ctrl_id, void *value);
```

- 参数

参数	描述
dev_id	ISP 设备 ID 号
ctrl_id	命令 ID, 详见 typedef enum {...} hw_isp_ctrl_cfg_ids;
value	参数指针

```
typedef enum {  
    /*isp_ctrl*/  
    ISP_CTRL_MODULE_EN      = 0x0000,  
    ISP_CTRL_DIGITAL_GAIN    = 0x0001,  
    ISP_CTRL_PLTMWDR_STR     = 0x0002,  
    ISP_CTRL_DN_STR          = 0x0004,  
    ISP_CTRL_3DN_STR         = 0x0008,  
    ISP_CTRL_HIGH_LIGHT      = 0x0010,  
    ISP_CTRL_BACK_LIGHT      = 0x0020,  
}
```

```
ISP_CTRL_WB_MGAIN      = 0x0040,  
ISP_CTRL_AGAIN_DGAIN   = 0x0080,  
ISP_CTRL_COLOR_EFFECT  = 0x0100,  
} hw_isp_ctrl_cfg_ids;
```

2.2.1 获取 ISP 数字增益

参数	描述
命令 id	ISP_CTRL_DIGITAL_GAIN
Value	指向 int 型数值的指针，返回当前 ISP 数字增益，256 为 1 倍

备注：该命令为只读。

2.2.2 设置 2D 降噪强度

参数	描述
命令 id	ISP_CTRL_DN_STR
Value	指向 int 型数值的指针，传递 2D 降噪强度，50 为 1 倍

备注：具体效果与 ISP 参数有关。

2.2.3 设置 3D 降噪强度

参数	描述
命令 id	ISP_CTRL_3DN_STR
Value	指向 int 型数值的指针，传递 3D 降噪强度，50 为 1 倍

备注：具体效果与 ISP 参数有关。

2.2.4 设置面光程度

参数	描述
命令 id	ISP_CTRL_HIGH_LIGHT
Value	指向 int 型数值的指针，范围 [-31, 31]

备注：面光程度越大，算法启动抗面光算法使最终画面越暗。

2.2.5 设置背光程度

参数	描述
命令 id	ISP_CTRL_BACK_LIGHT
Value	指向 int 型数值的指针，范围 [-31, 31]

备注：背光程度越大，算法启动抗背光算法使最终画面越亮。

2.2.6 设置手动白平衡增益

参数	描述
命令 id	ISP_CTRL_WB_MGAIN
Value	指向白平衡增益结构体的指针，详见：struct isp_wb_gain

备注：使用前需要把白平衡设置成手动模式，白平衡增益，256 为单位 1。

```
struct isp_wb_gain {
    HW_U16 r_gain;
    HW_U16 gr_gain;
    HW_U16 gb_gain;
    HW_U16 b_gain;
};
r_gain: r通道增益;
gr_gain: gr通道增益;
gb_gain: gb通道增益;
b_gain: b通道增益;
```

2.2.7 设置 AE 模拟和数字增益范围

参数	描述
命令 id	ISP_CTRL_AGAIN_DGAIN
Value	指向模拟数字增益结构体的指针，详见：struct gain_cfg

备注：gain_favor 为 0 时，AE 的增益按照 AE table 的配置来限制，自动曝光增益，256 为单位 1。

```
struct gain_cfg {
    HW_S32 gain_favor;
    HW_S32 ana_gain_min;
    HW_S32 ana_gain_max;
    HW_S32 dig_gain_min;
    HW_S32 dig_gain_max;
};

gain_favor: 0:normal, 1:ana_gain first, 2:dig_gain first.
ana_gain_min: 最小模拟增益;
ana_gain_max: 最大模拟增益;
dig_gain_min: 最小数字增益;
dig_gain_max: 最大数字增益.
```

2.2.8 设置滤镜特效

参数	描述
命令 id	ISP_CTRL_COLOR_EFFECT
Value	指向枚举型数值的指针，范围 [0, 6]，依次分别为：正常、中灰..， 详见：enum colorfx

备注：正常、中灰、负片、复古、红调、绿调、蓝调。

```
enum colorfx {
    ISP_COLORFX_NONE = 0,
    ISP_COLORFX_GRAY,
    ISP_COLORFX_NEGATIVE,
    ISP_COLORFX_ANTIQU,
    ISP_COLORFX_RTONE,
    ISP_COLORFX_GTONE,
    ISP_COLORFX_BTONE,
};
```

2.3 效果文件修改接口

效果文件的修改接口复用了 ISP 工具调试参数的接口，使用该接口可以在线修改 ISP 效果文件，并能实现命令接口较难实现的组合参数的修改与设置。

使用建议：先调用 `isp_get_cfg` 获取当前效果文件的配置，再修改默写想要修改的变量后调用 `isp_set_cfg` 设置到 `libisp` 中。

这些接口对应的命令与结构体详见：`libisp/isp_tuning/isp_tuning.c`。下面将对几个典型功能的实现做说明。

`isp_set/get_cfg`

- 目的

修改/获取 ISP 效果配置

- 语法

```
HW_S32 isp_get_cfg(int dev_id, HW_U8 group_id, HW_U32 cfg_ids, void *cfg_data);
```

- 参数

参数	描述
<code>dev_id</code>	ISP 设备 ID 号
<code>group_id</code>	命令 ID，分别是：测试参数配置组..， 详见 <code>typedef enum {...} hw_isp_cfg_groups</code>
<code>cfg_id</code>	配置 ID
<code>cfg_data</code>	配置数据结构指针

备注：命令 ID，分别是：测试参数配置组、3A 配置参数组、Tuning 参数组、动态参数配置组、组计数。

```
typedef enum {  
    HW_ISP_CFG_TEST           = 0x01,  
    HW_ISP_CFG_3A            = 0x02,  
    HW_ISP_CFG_TUNING        = 0x03,  
    HW_ISP_CFG_DYNAMIC       = 0x04,  
    HW_ISP_CFG_GROUP_COUNT  
} hw_isp_cfg_groups;
```

2.3.1 ISP 模块开关

参数	描述
命令组 ID	HW_ISP_CFG_TEST
配置 ID	HW_ISP_CFG_TEST_ENABLE
配置数据结构	详见 struct isp_test_enable_cfg，结构体中每一个参数对应一个模块。

备注：使用时可以先 get 效果文件中默认的配置，然后将某个需要修改的模块单独配置成 0 或者 1。

```
struct isp_test_enable_cfg {  
    HW_S32    manual;  
    HW_S32    afs;  
    HW_S32    sharp;  
    HW_S32    contrast;  
    HW_S32    denoise;  
    HW_S32    drc;  
    HW_S32    cem;  
    HW_S32    lsc;  
    HW_S32    gamma;  
    HW_S32    cm;  
    HW_S32    ae;  
    HW_S32    af;  
    HW_S32    awb;  
    HW_S32    hist;  
    HW_S32    blc;  
    HW_S32    so;  
    HW_S32    wb;  
    HW_S32    otfdpc;  
    HW_S32    cfa;  
    HW_S32    tdf;  
    HW_S32    cnr;  
    HW_S32    saturation;  
    HW_S32    defog;  
    HW_S32    linearity;  
    HW_S32    gtm;  
    HW_S32    dig_gain;  
    HW_S32    pltm;  
    HW_S32    wdr;  
    HW_S32    ctc;  
};
```

2.3.2 AE table 设置

参数	描述
命令组 ID	HW_ISP_CFG_3A

参数	描述
配置 ID	HW_ISP_CFG_AE_PREVIEW_TBL、 HW_ISP_CFG_AE_CAPTURE_TBL、 HW_ISP_CFG_AE_VIDEO_TBL
配置数	详见
数据结构	struct isp_ae_table_cfg

备注：一般建议 preview/capture/video 三种模式的 AE table 同时修改。

```
struct isp_ae_table_cfg {
    HW_S32    length;
    struct ae_table value[7];
};
struct ae_table {
    HW_U32 min_exp; //us
    HW_U32 max_exp;
    HW_U32 min_gain;
    HW_U32 max_gain;
    HW_U32 min_iris;
    HW_U32 max_iris;
};
```

2.3.3 AE 权重表设置

参数	描述
命令组 ID	HW_ISP_CFG_3A
配置 ID	HW_ISP_CFG_AE_WIN_WEIGHT
配置数据结构	传递 8×8 的权重表，详见 struct isp_ae_weight_cfg

备注：与命令接口的中心测光、平均测光不同，权重表可以选择想要的目标区域进行曝光。

```
struct isp_ae_weight_cfg {
    HW_S32    weight[64];
};
```

3 统计值获取

isp_stats_req

- 目的

获取 ISP 统计值申请接口。

- 语法

```
HW_S32 isp_stats_req(int dev_id, struct isp_stats_context *stats_ctx);
```

- 参数

参数	描述
dev_id	ISP 设备 ID 号
stats_ctx	参数结构指针，详见 struct isp_stats_context 结构体

```
struct isp_stats_context {  
    HW_U32 pic_w;  
    HW_U32 pic_h;  
  
    struct isp_stats_s stats;  
    struct isp_wb_gain wb_gain_saved;  
    bool enabled;  
};
```

pic_w: ISP 有效图像宽度; pic_h: ISP 有效图像高度; stats: ISP 统计值; wb_gain_saved: WB 矫正增益, 该增益会修正 WB 统计值; enabled: 暂时无用;

```
struct isp_stats_s {  
    struct isp_ae_stats_s ae_stats;  
    struct isp_awb_stats_s awb_stats;  
    struct isp_af_stats_s af_stats;  
};
```

ae_stats: AE 统计值; awb_stats: AWB 统计值; af_stats: AF 统计值;

```

struct isp_ae_stats_s {
    HW_U32 win_pix_n;
    HW_U32 accum_r[ISP_AE_ROW][ISP_AE_COL];
    HW_U32 accum_g[ISP_AE_ROW][ISP_AE_COL];
    HW_U32 accum_b[ISP_AE_ROW][ISP_AE_COL];
    HW_U32 avg[ISP_AE_ROW*ISP_AE_COL];

    HW_U32 hist[ISP_HIST_NUM];
};

```

win_pix_n: 每个统计窗口的像素数目; ISP_AE_ROW: AE 分块行数 32; ISP_AE_COL: AE 分块列数 48; accum_r: 分块 R 分量的和; accum_g: 分块 G 分量的和; accum_b: 分块 B 分量的和; avg: 分块 RGB 加权平均值 (0~255); hist: 直方图统计值;

```

struct isp_awb_stats_s {
    HW_U32 awb_sum_r[ISP_AWB_ROW][ISP_AWB_COL];
    HW_U32 awb_sum_g[ISP_AWB_ROW][ISP_AWB_COL];
    HW_U32 awb_sum_b[ISP_AWB_ROW][ISP_AWB_COL];
    HW_U32 awb_sum_cnt[ISP_AWB_ROW][ISP_AWB_COL];

    HW_U32 awb_avg_r[ISP_AWB_ROW][ISP_AWB_COL];
    HW_U32 awb_avg_g[ISP_AWB_ROW][ISP_AWB_COL];
    HW_U32 awb_avg_b[ISP_AWB_ROW][ISP_AWB_COL];
    HW_U32 avg[ISP_AWB_ROW][ISP_AWB_COL];
};

```

ISP_AWB_ROW: AWB 分块行数 32; ISP_AWB_COL: AWB 分块列数 48; awb_sum_r: 分块 R 分量的和; awb_sum_g: 分块 G 分量的和; awb_sum_b: 分块 B 分量的和; awb_sum_cnt: 分块有效像素的和;

awb_avg_r: 分块 R 分量的平均值 (0~255); awb_avg_g: 分块 G 分量的平均值 (0~255); awb_avg_b: 分块 B 分量的平均值 (0~255); avg: 分块 RGB 加权平均值 (0~255);

```

struct isp_af_stats_s {
    HW_U32 af_count[ISP_AF_ROW][ISP_AF_COL];
    HW_U32 af_h_d1[ISP_AF_ROW][ISP_AF_COL];
    HW_U32 af_h_d2[ISP_AF_ROW][ISP_AF_COL];
    HW_U32 af_v_d1[ISP_AF_ROW][ISP_AF_COL];
    HW_U32 af_v_d2[ISP_AF_ROW][ISP_AF_COL];
};

```

ISP_AF_ROW: AF 分块行数 8; ISP_AF_COL: AF 分块列数 8; af_count: 分块有效像素的和; af_h_d1: 分块水平方向梯度 1; af_h_d2: 分块水平方向梯度 2; af_v_d1: 分块垂直方向梯度 1; af_v_d2: 分块垂直方向梯度 2;

```

struct isp_wb_gain {
    HW_U16 r_gain;
    HW_U16 gr_gain;
    HW_U16 gb_gain;
    HW_U16 b_gain;
};

```

r_gain: r 分量增益; gr_gain: gr 分量增益; gb_gain: gb 分量增益; b_gain: b 分量增益;
四个增益中最小增益不能小于 256 (1 倍)






著作权声明

版权所有 © 2020 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、、**全志科技**、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的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所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。