项目：DUOBAO　　编号：3 位数字编号
类别：参考手册　　密级：绝密/保密/普通

DUOBAO 媒体处理函数开发参考

版本：0.13

(草稿)

部门：　芯片研发部

编写：

审核：

批准：

Fullhan 上海富瀚微电子有限公司
Shanghai Fullhan Microelectronics Co., Ltd.

二零一五年二月五日

前言

设计 duobao 平台 SDK 时，考虑 API 应用 IPC 的需求和特定应用场景，基本思路基于：

1. 清晰的模块化设计；
2. 清晰的接口参数定义；
3. 灵活的数据流配置；
4. 详细的错误码反馈；
5. 特别是内存使用静态分配，内存消耗较少；

修订记录

修订时间	修订版本	修订人	修订描述
2015/2/5	V0.1		初稿
2015/5/28	V0.12	陈佳男	修改 DSP 部分文档
2015/5/28	V0.12	胡鑫源	修改音频部分文档
2015/6/5	V0.13	陈佳男	修改 DSP API 部分文档

目录

图形目录

表格目录

未找到图形项目表。

1　引言

DuoBao 是面向中低网络摄像机的应用的 H.264 编码芯片，主芯片为 ARM1176EJS。

1.1 编写目的

本文档

1.2 背景

项目名称：　DUOBAO

1.3 定义

1.4 参考资料

2 系统控制

2.1 功能描述

参考目前主流 IPC 需求，结合 FH81 芯片特性，提供媒体处理软件开发包 SDK，支持 IPC 应用软件快速开发。SDK 封装了芯片内核的复杂的底层处理，对应用软件提供 API 接口完成相应功能。支持应用软件快速开发以下功能：视频图像处理、H.264/ JPEG 编码、视频输出、等功能。下面是 FH81 概要框图：

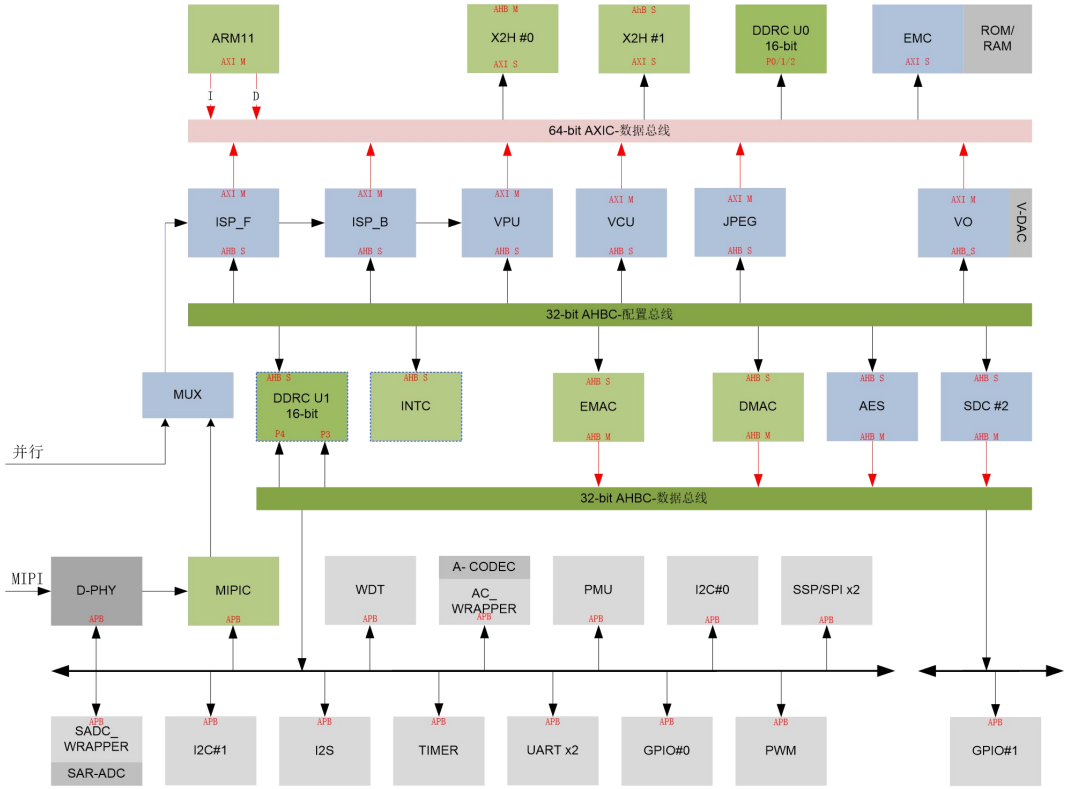


图 2-1 FH81 概要框图

系统控制部分负责完成硬件各个部件驱动的加载、硬件复位、初始化工作，同时完成各功能模块的初始化、退出以及管理功能模块的状态、版本、提供专用物理内存管理等功能。应用程序启动业务前，须完成系统初始化。应用程序退出时，调用系统退出工作，释放相关资源。

数据通路主要包括的功能模块：

1. VPU： 视频处理单元，主要负责视频编码前的预处理
2. PAE： 并行 H.264 编码器
3. VOU：视频输出单元
4. JPEG：JPEG 编码单元

IPC 的编码通道是根据不同分辨率输出需求而抽象出来的概念，VPU 输出支持 4 个（硬件固定）通道，其中通道 0、1、3 可以分别与编码通道绑定，进行绑定是通道间宽高必须一致，否则绑定失败。通道 2 固定与视频输出绑定，输出格式是 YUV422。

进行 JPEG 抓图时，指定通道绑定到 JPEG 编码模块，进行 JPEG 编码，同时并不影响 H.264 编码，JPEG 编码完成后自动解绑，即 JPEG 编码每帧需进行一次绑定。

绑定关系如下：

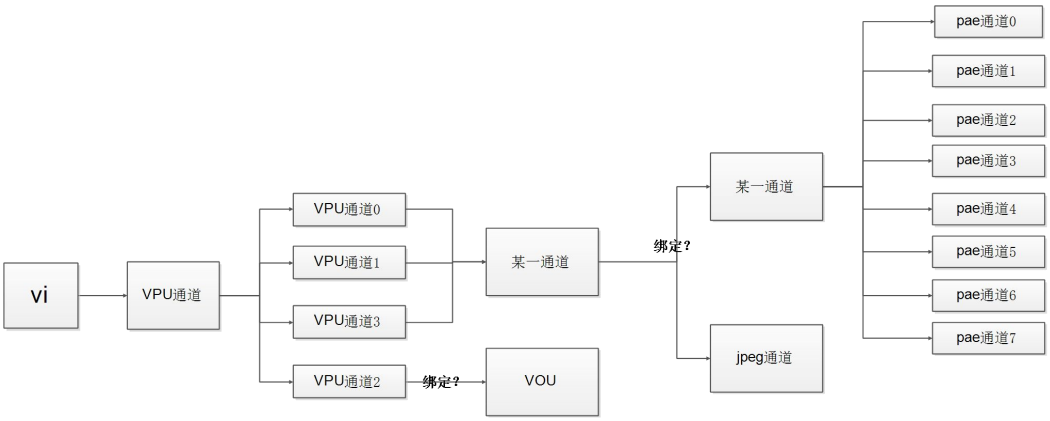


图 2-2 绑定关系图

为了通道设置灵活性，暂把 PAE 通道设定为 8 个(可通过软件配置)。

2.2 API 参考

系统控制实现系统初始化、版本信息、资源配置、模块绑定关系配置等功能。

该功能模块提供以下 API：

- FH_SYS_Init：初始化系统。
- FH_SYS_Exit：退出系统。
- FH_SYS_Set_Resource：配置系统资源参数。
- FH_SYS_BindVpu2Enc：数据源绑定到 PAE 编码通道。
- FH_SYS_BindVpu2Vou：数据源绑定到视频输出通道。
- FH_SYS_BindVpu2Jpeg：数据源绑定到 JPEG 编码通道。
- FH_SYS_UnBindbySrc：通过数据源通道号直接解除绑定。
- FH_SYS_GetBindbyDest：通过目标通道获取绑定的数据源通道号。
- FH_SYS_GetVersion：获取 MPP 的版本号。
- FH_SYS_SetReg：设置寄存器的值。
- FH_SYS_GetReg：获取寄存器的值。
- ~~*FH_SYS_GetVRegAddr：获取寄存器的地址。*~~
- ~~*FH_SYS_CloseFd：关闭文件系统。*~~
- ~~*FH_SYS_MmzAlloc：在用户态分配 MMZ 内存。*~~
- ~~*FH_SYS_MmzAlloc_Cached：在用户态分配 MMZ 内存，该内存支持 cache 缓存。*~~
- ~~*FH_SYS_MmzFlushCache：清空 cache 里的内容到内存并且使 cache 里的内容无效。*~~
- ~~*FH_SYS_MmzFree：在用户态释放 MMZ 内存。*~~

1. FH_SYS_Init

■ 功能说明

初始化 MPP 系统。

■ 函数定义

FH_SINT32 FH_SYS_Init();

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

2. FH_SYS_Exit

- 功能说明

  退出系统。

- 函数定义

  FH_SINT32　FH_SYS_Exit();

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

3. FH_SYS_BindVpu2Enc

- 功能说明

  数据源绑定到编码通道。

- 函数定义

  FH_SINT32　FH_SYS_BindVpu2Enc(FH_UINT32 srcchn,FH_UINT32 dstschn)

- 输入参数

  srcchn：数据源通道号；取值 0、1 或 2：

  - 0 表示主码流，对 FH81 来所，一般为 720P；

- 1 为第 1 辅码流，一般设置 D1，
- 2 为第 2 辅码流，一般设为 D1 或 CIF (说明：此处非规格定义的 0、1 或 3)

dstschn：编码通道，取值 0~7。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

4. FH_SYS_BindVpu2Vou

■ ~~功能说明~~

~~数据源绑定视频输出。~~

■ ~~函数定义~~

~~FH_SINT32 FH_SYS_BindVpu2Vou();~~

■ ~~输入参数~~

~~无。~~

■ ~~输出参数~~

~~无。~~

■ ~~返回值~~

~~RETURN_OK：函数调用成功。~~

~~其他：失败，函数调用失败的原因见出错信息。~~

5. FH_SYS_BindVpu2Jpeg

■ 功能说明

数据源绑定 JPEG 编码通道。

■ 函数定义

FH_SINT32 FH_SYS_BindVpu2Jpeg(FH_UINT32 srcchn)

■ 输入参数

srcchn：数据源通道，源通道信息参考 FH_SYS_BindVpu2Enc 的输入参数描述

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

6. FH_SYS_GetBindbyDest

- 功能说明

  通过获取绑定的源信息。

- 函数定义

  FH_SINT32　FH_SYS_GetBindbyDest(FH_UINT32 *srcchn,FH_UINT32 dstschn)

- 输入参数

  dstschn：编码通道。

- 输出参数

  srcchn：数据源通道。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

7. FH_SYS_UnBindbySrc

- 功能说明

  解除绑定关系。

- 函数定义

  FH_SINT32　FH_SYS_UnBindbySrc (FH_UINT32 srcchn)

- 输入参数

  srcchn：数据源通道。

- 输出参数

  无。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

8. FH_SYS_GetVersion

- 功能说明

  获得版本号。

- 函数定义

  FH_SINT32　FH_SYS_GetVersion(FH_VERSION_INFO *pstSystemversion)

- 输入参数

  无。

- 输出参数

  pstSystemversion：版本号信息。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9. FH_SYS_SetReg

- 功能说明

  设置寄存器的值。

- 函数定义

  FH_SINT32 FH_SYS_SetReg(FH_UINT32 addr, FH_UINT32 value);

- 输入参数

  addr：寄存器地址。

  value：寄存器设置值。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

10. FH_SYS_GetReg

- 功能说明

  获得寄存器的值。

- 函数定义

  FH_SINT32 FH_SYS_GetReg(FH_UINT32 u32Addr, FH_UINT32 *pu32Value);

- 输入参数

  u32Addr：寄存器地址。

- 输出参数

  u32Value：寄存器值。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

2.3 数据结构

2.3.1 基本数据类型

通用数据类型

```
typedef unsigned long long      FH_UINT64;
typedef long long               FH_SINT64;
typedef unsigned int            FH_UINT32;
typedef int                     FH_SINT32;
typedef unsigned short          FH_UINT16;
typedef short                   FH_SINT16;
typedef unsigned char           FH_UINT8;
typedef char                    FH_SINT8;
typedef FH_UINT8*               FH_ADDR;
```

FH_BOOL

```
typedef enum {
    FH_FALSE = 0,
    FH_TRUE = 1,
} FH_BOOL;
```

基本常量定义

序号	定义	说明	值
1.	MAX_VPU_CHANNO	VPU 处理的最大通道数	4
2.	MAX_ENC_CHANNO	PAE 编码的最大通道数	8
3.	PAE_MAX_NALU_CNT	PAE 编码的最大 NALU 数	20
4.	MAX_OSD_LINE	文本 OSD 数	3
5.	MAX_INVERT_CNT	文本 OSD TXT2 的显示行数	8
6.	MAX_MASK_AREA	最大屏蔽区域数	8
7.	**MIN_PITCH_ALIGN**	**图片 OSD 像素对齐**	**8**
8.	MAX_TEXT_LINE	文本 OSD 数	2
9.	DEFALT_ALIGN	默认数据对齐	4
10.	MOU_ENABLE	模块使能	1
11.	MOU_UNABLE	模块关闭	0

基本结构定义

FH_POINT

- 功能说明

  点坐标

- 定义

```
typedef struct fhPOINT_
{
    FH_UINT32           u32X; /**< 水平坐标 */
    FH_UINT32           u32Y; /**< 垂直坐标 */
}FH_POINT;
```

FH_SIZE

- 功能说明

  尺寸

- 定义

```
typedef struct fhSIZE_S
{
    FH_UINT32           u32Width; /**< 宽度像素  */
    FH_UINT32           u32Height; /**<高度像素  */
} FH_SIZE;
```

FH_AREA

- 功能说明

  区域信息

- 定义

```
typedef struct fhRECT_S
{
    FH_UINT32           u32X;   /**< 起始点水平坐标 */
    FH_UINT32           u32Y;   /**< 起始点垂直坐标*/
    FH_UINT32           u32Width; /**< 区域宽度*/
    FH_UINT32           u32Height; /**< 区域高度 */
}FH_AREA;
```

FH_ADDR_INFO

- 功能说明

  数据内存信息

- 定义

```
typedef struct
{
    FH_ADDR             addr;
```

```
    FH_UINT32               size;
}FH_ADDR_INFO;
```

FH_MEM_INFO

■ 功能说明

数据内存信息

■ 定义

```
typedef struct
{
    unsigned int base;  /**< 物理地址 */
    void * vbase;       /**< 虚拟地址 */
    unsigned int size;  /**< 内存大小 */
}FH_MEM_INFO;
```

2.3.2 系统数据类型

FH_VERSION_INFO

■ 功能说明

描述版本信息

■ 定义

```
typedef struct fhVERSION_INFO_
{
    FH_UINT32           build_data;         /**<构建号*/
    FH_UINT32           sw_version;         /**<软件版本号*/
    FH_UINT32           hw_version;         /**<硬件驱动版本号*/
}FH_VERSION_INFO;
```

FH_FRAMERATE

■ 功能说明

帧率描述结构体，实际帧率 为 frame_count/frame_time

■ 定义

```
typedef struct
{
    FH_UINT16               frame_count;      /**< 帧数 */
    FH_UINT16               frame_time;       /**< 统计时间 (s)*/
} FH_FRAMERATE;
```

FH_ROTATE

■ 功能说明

旋转参数控制：
0→不旋转
1→旋转 90 度
2→旋转 180 度
3→旋转 270 度

■ 定义

```
typedef enum
{
    ROTATE_NONE = 0,
    ROTATE_90   = 1,
    ROTATE_180  = 2,
    ROTATE_270  = 3,
} FH_ROTATE;
```

2.4 错误码

➢ **常规返回**

```
#define RETURN_OK                 0       /**< 返回正确 */
#define NO_DEVICE                 (-1)    /**< 设备不存在 */
#define OPEN_DEVICE_ERR           (-1)    /**<  */
#define CLOSE_DEVICE_ERR          (-2)    /**<  */
#define FIND_DEVICE_ERR           (-3)    /**<  */
#define PARAM_NULL                (-4)    /**<  */
#define PARAM_ERR                 (-5)    /**< 参数错误 */
#define ALIGN_ERR                 (-6)    /**< 对齐错误 */
#define MODULE_ENABLE_ERR         (-7)    /**< 模块使能错误 */
#define CHAN_ERR                  (-8)    /**< 通道错误 */
#define MEM_NULL                  (-9)    /**< 内存错误 */
```

➢ **系统错误**

```
#define MEDIA_ERR_BASE                (-500) /**< 系统错误位 */
#define MEDIA_ERR_NOT_MATCH           (-501) /**< 参数不匹配 */
#define MEDIA_ERR_PARAM_NULL          (-502) /**< 参数为 NULL */
#define MEDIA_ERR_UNCLEAR_PARAM       (-503) /**< 参数无效 */
#define MEDIA_ERR_HAVE_BEEN_BINDED    (-504) /**< 已经绑定 */
#define MEDIA_ERR_INVALID_CHAN        (-505) /**< 无效通道 */
#define MEDIA_ERR_INVALID_BIND        (-506) /**< 无效绑定 */
#define MEDIA_ERR_NOT_ENOUGH_MEM      (-507) /**< 内存不足 */
```

3 视频处理

3.1 功能描述

视频输入模块，由于 sensor 输入的 VI 和 ISP 以在线模式的形式绑定在一起，故不在此配置，这里仅指输入给视频处理子系统的图像宽高。由于相关 API 比较少，就加到了 VPU 模块里面了。

VPU（视频处理单元），这里 VPU 功能相当于 VPSS。支持对一幅输入图像进行统一预处理，如去噪、锐化，然后进行叠加 LOGO、MASK、OSD 等图像处理，再对各通道分别进行缩放等处理，最后输出多种不同分辨率的图像。

其中内容叠加部分内容较多，单独章节描述，参考内容叠加。

VPU 单元支持的具体图像处理功能包括切割、去噪、图形叠加、文字叠加、屏蔽、缩放、均值计算等。功能框图：

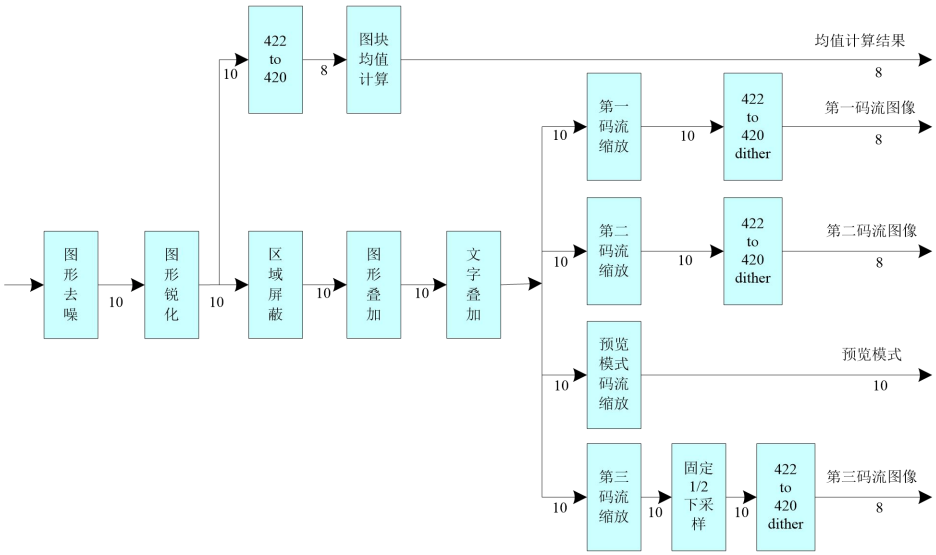


图 3-1 VPU Core 框图

VPU 支持输入图像处理能力支持 1280x960@30fps，10bits 输入幅面最大支持 1280x960，对于输入幅面大于 1280x960，进行切割处理。总共有 4 个码流通道

第一子码流幅面最大宽度 1280，第二子码流幅面最大宽度 720，支持原图像 Crop。

支持 4:2:2 待处理图像数据，输出编码图像为 8 位，输出预览图像为 10 位；工作模式主要有以下 4 个：

- **预览模式 1：**

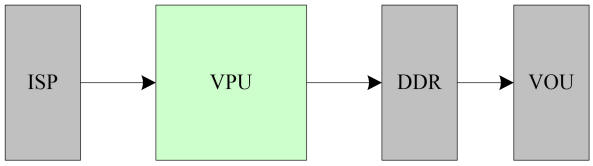


图 3-2 VPU 预览模式 1

- 预览图像来自第二个码流，第二个码流以 422-10bits 的方式写入 DDR
- 对于超过 1280x960 的幅面，需要 VPU 和 ISP 同时分块处理
- 允许 ISP 主动通过相关启动信号来控制 VPU 的启动

➢ **编码模式 0：**

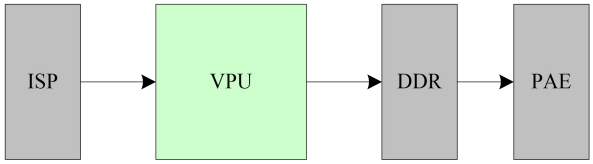


图 3-3 VPU-PAE 编码模式

- 三个码流以 420 的方式写入 DDR
- 对于超过 1280x960 的幅面，需要 VPU 和 ISP 同时分块处理
- 允许 ISP 主动通过相关启动信号来控制 VPU 的启动

➢ **编码模式 1：**

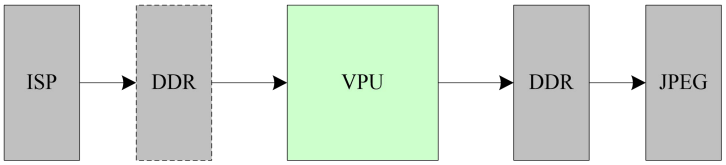


图 3-4 VPU-JPEG 编码模式

- VPU 中的 Jpeg 码流图像以 420 的方式写入 DDR
- 对于超过 1280x960 的幅面，如果图像来自 ISP，需要 VPU 和 ISP 同时分块处理
- 对于超过 1280x960 的幅面，如果图像来自 DDR，仅仅 VPU 需要分块处理
- 如果 VPU 输入图像来自 DDR，VPU 启动受软件控制。
- 如果 VPU 输入图像来自 ISP，允许 ISP 主动通过相关启动信号来控制 VPU 的启动

➢ **编码模式 2**

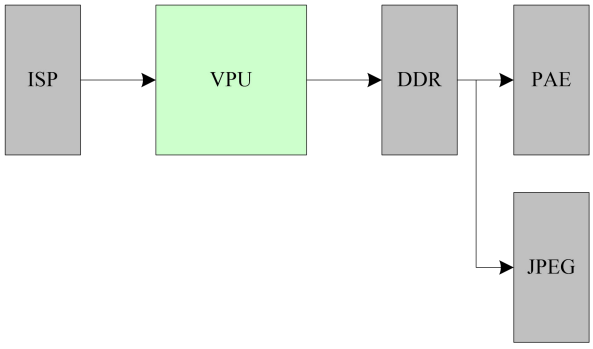


图 3-5 VPU-JPEG-PAE 编码模式

- PAE 和 JPEG 同时编码
- VPU 中的 Jpeg 码流图像以 420 的方式写入 DDR
- 三个码流以 420 的方式写入 DDR
- 允许 ISP 主动通过相关启动信号来控制 VPU 的启动

3.2 API 参考

- FH_VPSS_SetViAttr：设置 VI 设备属性。
- FH_VPSS_GetViAttr：获取 VI 设备属性。
- ~~FH_VPSS_GetChnCapality：获取通道内存分配能力。已改成内部函数~~
- FH_VPSS_SysInitMem：初始化 VPU 模块的内存。
- ~~FH_VPSS_ChnInitMem：初始化 VPU 通道的内存。已改成内部函数~~
- FH_VPSS_Enable：使能 VPU 通道。
- FH_VPSS_Disable：关闭 VPU 通道。
- FH_VPSS_FreezeVideo：冻结视频编码。
- FH_VPSS_UnfreezeVideo：解冻视频编码。
- FH_VPSS_SendUserPic：支持用户发送图片信息进行编码。
- FH_VPSS_GetChnFrame：从 VPU 通道获取图像。
- FH_VPSS_OpenModule：打开 VPU 通道中相应模块。
- FH_VPSS_CloseModule：关闭 VPU 通道中相应模块。
- FH_VPSS_QueryModule：查询 VPU 中打开的相应模块状态。
- FH_VPSS_SetChnAttr：设置 VPU 通道属性。
- FH_VPSS_GetChnAttr：获取 VPU 通道属性。
- FH_VPSS_OpenChn：打开 VPU 通道。
- FH_VPSS_CloseChn：关闭 VPU 通道。
- FH_VPSS_SetMask: 设置 VPU 通道视频覆盖区域。
- FH_VPSS_GetMask：获取 VPU 通道视频覆盖区域。
- FH_VPSS_ClearMask：清除 VPU 视频覆盖区域块。
- FH_VPSS_SetGraph：设置 VPU 通道图层叠加区域。
- FH_VPSS_GetGraph：获取 VPU 通道图层叠加区域。
- FH_VPSS_SetOsd：设置 VPU 通道字符叠加。
- FH_VPSS_GetOsd：获取 VPU 通道字符叠加信息。
- FH_VPSS_SetOsdInvert：设置字符叠加反色信息。
- FH_VPSS_GetOsdInvert：获取字符叠加反色信息。
- FH_VPSS_SetYCmean：设置 yc 均值统计。
- FH_VPSS_GetYCmean：获取 yc 均值统计。
- FH_VPSS_SetFramectrl：设置帧率控制参数。
- FH_VPSS_GetFramectrl：获取帧率控制参数。
- FH_VPSS_GetFrameRate：获取当前的帧率。
- FH_VPSS_SetCrop：设置 VPU 通道裁剪功能属性。
- FH_VPSS_GetCrop：获取 VPU 通道裁剪功能属性。

- ~~FH_VPSS_SetYCnr：设置 VPU 通道 NR 使能开关。~~
- ~~FH_VPSS_GetYCnr：获取 VPU 通道 NR 开关状态。~~
- ~~FH_VPSS_SetApe：设置边界锐化功能属性。~~
- ~~FH_VPSS_GetApe：获取边界锐化功能属性。~~
- ~~FH_VPU_SetPurple：设置紫边功能属性。~~
- ~~FH_VPU_GetPurple：获取紫边功能属性。~~
- ~~FH_VPU_GetPkginfo：获取 PKG 的设置信息。~~

3.2.1 视频输入部分(vi)

1. FH_VPSS_SetViAttr

■ 功能说明

设置视频输入属性。

■ 函数定义

FH_SINT32 FH_VPSS_SetViAttr(FH_VPU_SIZE *pstViconfig);

■ 输入参数

pstViconfig：视频输入属性值的指针。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

1：失败，函数调用失败的原因见出错信息。

2. FH_VPSS_GetViAttr

■ 功能说明

获取视频输入属性。

■ 函数定义

FH_SINT32 FH_VPSS_GetViAttr(FH_VPU_SIZE *pstViconfig);

■ 输入参数

无。

■ 输出参数

pstViconfig：视频输入属性值的指针。

- 返回值

  RETURN_OK：函数调用成功。

  1：失败，函数调用失败的原因见出错信息。

3.2.2 视频处理单元（vpss）

1. FH_VPSS_SysInitMem

- 功能说明

  初始化 VPSS 模块的系统内存,一般情况下由 FH_SYS_Set_Resource 内部调用，无需显式调用。

- 函数定义

  FH_SINT32 FH_VPSS_SysInitMem();

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

2. FH_VPSS_Enable

- 功能说明

  指定模式启用 VPU 通道，如果更换模式时，需先调用 FH_VPSS_Disable,然后再调用此函数

- 函数定义

  FH_SINT32 FH_VPSS_Enable(FH_VPU_VI_MODE mode)

- 输入参数

  mode 启用模式

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

3. FH_VPSS_Disable

- 功能说明

  禁用 VPSS 通道。

- 函数定义

  FH_SINT32 FH_VPSS_Disable()

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

4. FH_VPSS_FreezeVideo

- 功能说明

  视频冻结。

- 函数定义

  FH_SINT32 FH_VPSS_FreezeVideo ()

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

5. FH_VPSS_UnfreezeVideo

- 功能说明

  视频解冻。

- 函数定义

  FH_SINT32 FH_VPSS_ UnfreezeVideo ()

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

6. FH_VPSS_GetChnFrame

- 功能说明

  从 VPU 通道获取一帧图像。

- 函数定义

  FH_SINT32 FH_VPSS_GetChnFrame(FH_UINT32 chan, FH_VPU_STREAM *pstVpuframeinfo)

- 输入参数

  chan: 指定通道

  ptsVpuframeinfo: 获取的图像信息指针

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

7. FH_VPSS_SendUserPic

- 功能说明

  支持用户发送图片信息进行编码，可用于视频丢失时的插入自定义图片。

- 函数定义

  FH_SINT32 FH_VPSS_SendUserPic(const FH_VPU_USER_PIC *pstUserPic)

- 输入参数

  pstUserPic：用户发送到图片信息指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

8. FH_VPSS_GetUserPicAddr

- 功能说明

  获取用于存放用户图片的内存地址，可用于视频丢失时的插入自定义图片。

- 函数定义

FH_SINT32 FH_VPSS_GetUserPicAddr(FH_VPU_USER_PIC *pstUserPic);

- 输入参数

  pstUserPic：用户发送到图片信息指针。

  [pic_size] [ystride] [cstride]

- 输出参数

pstUserPic：用户发送到图片信息指针。

[yluma] [chroma]

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9. FH_VPSS_SetChnAttr

- 功能说明

  设置 VPU 通道属性。

- 函数定义

  FH_SINT32 FH_VPSS_SetChnAttr(FH_UINT32 chan, const FH_VPU_CHN_CONFIG *pstChnconfig);

- 输入参数

  chan：通道号。

  pstChnconfig：VPU 通道的属性指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

10. FH_VPSS_GetChnAttr

- 功能说明

  获取 VPU 通道设置的属性值。

- 函数定义

  FH_SINT32 FH_VPSS_GetChnAttr(FH_UINT32 chan, FH_VPU_CHN_CONFIG *pstChnconfig)

- 输入参数

  chan：通道号。

- 输出参数

  pstChnconfig：VPU 通道属性指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

11. FH_VPSS_OpenChn

- 功能说明

  打开 VPU 通道。

- 函数定义

  FH_SINT32　FH_VPSS_OpenChn(FH_UINT32 chan)

- 输入参数

  chan：通道值。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

12. FH_VPSS_CloseChn

- 功能说明

  关闭 VPU 通道。

- 函数定义

  FH_SINT32　FH_VPSS_CloseChn(FH_UINT32 chan)

- 输入参数

  chan：通道值。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

13. FH_VPSS_SetMask

- 功能说明

  设置 VPU 通道视频覆盖区域，通过设置参数 Enable 成员控制有效。

- 函数定义

  FH_SINT32 FH_VPSS_SetMask(const FH_VPU_MASK *pstVpumaskinfo)

- 输入参数

  pstVpumaskinfo：视频覆盖参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

14. FH_VPSS_GetMask

- 功能说明

  获取 VPU 通道视频覆盖区域。

- 函数定义

  FH_SINT32 FH_VPSS_GetMask(FH_VPU_MASK *pstVpumaskinfo)

- 输入参数

  无。

- 输出参数

  pstVpumaskinfo：视频覆盖参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

15. FH_VPSS_ClearMask

- 功能说明

  清除相应的视频覆盖区域。

- 函数定义

  FH_SINT32 FH_VPSS_ClearMask(FH_UINT32 maskAreaIdx);

- 输入参数

maskAreaIdx：视频覆盖区域号。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

16. FH_VPSS_SetGraph

- 功能说明

  设置 VPU 通道 logo 叠加信息，通过设置参数 Enable 成员控制有效。

- 函数定义

  FH_SINT32 FH_VPSS_SetGraph(const FH_VPU_LOGO *pstVpugraphinfo)

- 输入参数

  pstVpugraphinfo: logo 叠加参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

17. FH_VPSS_GetGraph

- 功能说明

  获得 VPU 通道 logo 叠加信息。

- 函数定义

  FH_SINT32 FH_VPSS_GetGraph(FH_VPU_LOGO *pstVpugraphinfo)

- 输入参数

  无。

- 输出参数

  pstVpugraphinfo: logo 叠加参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

18. FH_VPSS_SetOsd

- 功能说明

  设置 VPU 通道字符叠加。

- 函数定义

  FH_SINT32 FH_VPSS_SetOsd(const FH_VPU_OSD *pstVpuosdinfo)

- 输入参数

  pstVpuosdinfo：字符叠加参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

19. FH_VPSS_GetOsd

- 功能说明

  获取 VPU 通道字符叠加信息。

- 函数定义

  FH_SINT32 FH_VPSS_GetOsd(FH_VPU_OSD *pstVpuosdinfo)

- 输入参数

  无。

- 输出参数

  pstVpuosdinfo：字符叠加参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

20. FH_VPSS_ SetRotate

- 功能说明

  VPU 通道字符叠加旋转。

- 函数定义

  FH_SINT32 FH_VPSS_SetRotate(const FH_OSD_ROTATE Rotate)

- 输入参数

  Rotate 旋转角度设置。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

21. FH_VPSS_SetOsdInvert

- 功能说明

  设置 VPU 通道字符叠加反色控制信息。

- 函数定义

  FH_SINT32 FH_VPSS_SetOsdInvert(const FH_INVERT_CTRL *pstOsdinvertctl)

- 输入参数

  pstOsdinvertctl：字符叠加的反色控制位指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

22. FH_VPSS_GetOsdInvert

- 功能说明

  获取 VPU 通道字符叠加反色控制信息。

- 函数定义

  FH_SINT32 FH_VPSS_GetOsdInvert(FH_INVERT_CTRL *pstOsdinvertctl)

- 输入参数

  pstOsdinvertctl：字符叠加的反色控制位指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

23. FH_VPSS_EnableYCmean

- 功能说明

  开启 YC 均值统计值。

- 函数定义

FH_SINT32 FH_VPSS_EnableYCmean();

- 输入参数

  无。

- 输出参数

  无

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

24. FH_VPSS_DisableYCmean

- 功能说明

  关闭 YC 均值统计值。

- 函数定义

FH_SINT32 FH_VPSS_DisableYCmean();

- 输入参数

无。

- 输出参数

  无

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

25. FH_VPSS_GetYCmean

- 功能说明

  获取 YC 均值统计值。

- 函数定义

  FH_SINT32 FH_VPSS_GetYCmean(FH_VPU_YCMEAN *pstVpuycmeaninfo)

- 输入参数

  无。

- 输出参数

  pstVpuycmeaninfo：YC 均值计算参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

26. FH_VPSS_SetFramectrl *

- 功能说明

  设置帧率控制参数。

- 函数定义

  FH_SINT32 FH_VPSS_SetFramectrl(FH_UINT32 chan, const FH_FRAMERATE *pstFramerate)

- 输入参数

  chan：通道号。

pstFramerate：帧率控制参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

27. FH_VPSS_GetFramectrl *

- 功能说明

  获取帧率控制参数。

- 函数定义

  FH_SINT32 FH_VPSS_GetFramectrl(FH_UINT32 chan, FH_FRAMERATE *pstFramerate)

- 输入参数

  chan：通道号。

- 输出参数

  pstFramerate：帧率控制参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

28. FH_VPSS_GetFrameRate *

- 功能说明

  获取当前的帧率。

- 函数定义

  FH_SINT32 FH_VPSS_GetFrameRate(FH_UINT32 chan, FH_FRAMERATE *pstFramerate)

- 输入参数

  chan：通道值。

- 输出参数

pstFramerate：获取当前帧率信息指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

29. FH_VPSS_ LOW_LATENCY_Enable *

- 功能说明

  设置低延时模块。

- 函数定义

  FH_SINT32 FH_VPSS_LOW_LATENCY_Enable(FH_UINT32 chan);

- 输入参数

  chan：编码通道号。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

30. FH_VPSS_LOW_LATENCY_Disable *

- 功能说明

  获取低延时模块参数。

- 函数定义

  FH_SINT32 FH_VPSS_LOW_LATENCY_Disable(void);

- 输入参数
- 输出参数
- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

31. FH_VPSS_SetCrop

- 功能说明

  设置 VPU 通道裁剪功能参数。

- 函数定义

  FH_SINT32 FH_VPSS_SetCrop(FH_UINT32 chan, const FH_VPU_CROP *pstVpucropinfo)

- 输入参数

  chan：通道值。

  pstVpucropinfo：通道裁剪功能参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

32. FH_VPSS_GetCrop

- 功能说明

  获取 VPU 通道裁剪功能属性。

- 函数定义

  FH_SINT32 FH_VPSS_GetCrop(FH_UINT32 chan, FH_VPU_CROP *pstVpucropinfo)

- 输入参数

  chan：通道值。

- 输出参数

  pstVpucropinfo：通道裁剪功能参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

以下函数在相应的 ISP 文档中说明

3.2.3 数据结构

FH_VPU_SIZE

- 功能说明
- 定义

```
typedef struct
{
    FH_SIZE             vi_size;        /**< 视频输入幅面 */
}FH_VPU_SIZE;
```

FH_VPU_VI_MODE

- 功能说明
- 定义

```
typedef enum
{
    VPU_MODE_ISP        = 0,            /**< 从ISP输入模式*/
    VPU_MODE_MEM        = 1,            /**< 从MEM输入模式*/
}FH_VPU_VI_MODE;
```

FH_VPU_CHN_CONFIG

- 功能说明
- 定义

```
typedef struct
{
    FH_SIZE                 vpu_chn_size;       /**< 通道配置图片尺寸 */
}FH_VPU_CHN_CONFIG;
```

FH_VPU_USER_PIC

- 功能说明
- 定义

```
typedef struct
{
    FH_SIZE                 pic_size;   /**< 图片尺寸 */
    FH_ADDR                 yluma;      /**< 图片数据的 luma 地址 */
    FH_ADDR                 chroma;     /**< 图片数据的 chroma 地址 */
    FH_UINT32               ystride;    /**< 图片数据的 luma stride */
```

```
    FH_UINT32                   cstride;    /**< 图片数据的 chroma stride */
}FH_VPU_USER_PIC;
```

FH_VPU_MODULE

■ 功能说明

■ 定义

```
typedef enum
{
    FH_VPU_MOD_MASK         = (1 << 1),    /**<隐私屏蔽，覆盖 */
    FH_VPU_MOD_LOGO         = (1 << 3),    /**<图像层 */
    FH_VPU_MOD_STAT_MEAN    = (1 << 4),    /**<统计输出 */
    FH_VPU_MOD_OSD0         = (1 << 5),    /**<文字层 0 */
    FH_VPU_MOD_OSD1         = (1 << 6),    /**<文字层 1 */
    FH_VPU_MOD_OSD2         = (1 << 7),    /**<文字层 2 */
    FH_VPU_MOD_DITHER       = (1 << 15),   /**<防抖动*/
    FH_VPU_MOD_CNR          = (1 << 16),   /**<色度降噪 */
    FH_VPU_MOD_YNR          = (1 << 17),   /**<亮度降噪 */
    FH_VPU_MOD_APC          = (1 << 18),   /**<锐化 */
    FH_VPU_MOD_PURPLE       = (1 << 19),   /**<去紫边*/
} FH_VPU_MODULE;
```

FH_VPU_PIXELFORMAT

■ 功能说明

■ 定义

```
/**<YUV 格式
typedef enum
{
    PIXEL_FORMAT_MONOCHROME  = 0, /**<单色*/
    PIXEL_FORMAT_420         = 1, /**<420*/
    PIXEL_FORMAT_422         = 2, /**<422*/
    PIXEL_FORMAT_444         = 3  /**<444*/
}FH_VPU_PIXELFORMAT;
```

FH_VPU_STREAM

■ 功能说明

■ 定义

```
typedef struct
{
```

```
    FH_VPU_PIXELFORMAT  pixelFormat;  /**< 像素格式 */
    FH_UINT32           stride;       /**< 图像跨度 */
    FH_UINT32           pts;          /**< 时间戳 */
    FH_SIZE             size;         /**< 图像数据长度 */
    FH_MEM_INFO         yluma;        /**< luma 地址*/
    FH_MEM_INFO         chroma;       /**< chroma 地址*/
}FH_VPU_STREAM;
```

FH_MASK_MASAIC

■ 功能说明

■ 定义

```
typedef struct
{
    FH_ BOOL            masaic_enable;    /**<使能，1 显示 0 不显示 */
    FH_UINT32           masaic_size;      /**<尺寸 0: 16x16，1: 32x32 */
}FH_MASK_MASAIC;
```

FH_VPU_MASK

■ 功能说明

■ 定义

```
typedef struct
{
    FH_ BOOL            mask_enable[MAX_MASK_AREA];  /**< 覆盖使能 */
    FH_AREA             area_value[MAX_MASK_AREA];   /**< 屏蔽区域 */
    FH_UINT32           color;        /**< 显示颜色 格式为 0x XX(alpha) XX(Y)
XX(Cb) XX(cr)，MASK 的 Alpha 值实际并不生效仅作格式统一用*/
    FH_MASK_MASAIC      masaic;       /**< 马赛克属性 */
}FH_VPU_MASK;
```

FH_LOGO_CFG

■ 功能说明

■ 定义

```
typedef struct
{
    FH_UINT32           alpha_value;  /**< 图形层的图像格式是 ARGB1555 格式，当 A=1
的时候此处的 Alpha 起作用，取值[0..127] 取值越小，越透明 */
    FH_UINT32           dtvmode;      /**< 0 为 SDTV 模式，1 为 HDTV 模式*/
    FH_UINT32           rgbmode;      /**< RGB 模式：0 为 stdio RGB，1 为 computer
```

```
RGB*/
    FH_SIZE                 logo_size;      /**< logo 大小  */
}FH_LOGO_CFG;
```

FH_LOGO_STRIDE

■ 功能说明

■ 定义

~~typedef struct~~
~~{~~
~~FH_UINT32 start_offset; /**<水平坐标偏移，图片切割显示 */~~
~~FH_UINT32 stride_value; /**<行间补偿，~~
~~}FH_LOGO_STRIDE;~~

FH_VPU_LOGO (modified)

■ 功能说明

■ 定义

```
typedef struct
{
    FH_BOOL             logo_enable;   /**<  图片叠加使能  */
    FH_ADDR             logo_addr;     /**< logo 数据的物理地址  */
    FH_LOGO_CFG         logo_cfg;      /**< logo  配置  */
    FH_POINT            logo_startpos;/**< logo  左上角起始点  */
    FH_POINT            logo_cutpos;   /**<logo 切割显示起始点  */
    FH_UINT32           stride_value;    /**<行长度  */
}FH_VPU_LOGO;
```

FH_OSD_CFG

■ 功能说明

■ 定义

```
typedef struct
{
    FH_BOOL             Vamp;/*是否水平放大一倍*/
    FH_BOOL             Hamp;/*是否垂直放大一倍*/
    FH_ROTATE           txtrotate;/*整个字符旋转*/
    FH_UINT32           txtw;/*字符宽度  取值范围[1-32]*/
    FH_UINT32           txth;/*字符高度,只有 txt2 模式时设置取值范围[1-32]*/
    FH_UINT32           fontsize;/*字体占的内存大小*/
    FH_UINT32           fontw;
```

```
    FH_UINT32                   fonth;
}FH_OSD_CFG;
```

FH_OSD_COLOR

- 功能说明
- 定义

```
typedef struct
{
    FH_UINT32           norcolor;/*正常颜色  格式为 0x XX(alpha)XX(Y)XX(Cb)XX(cr)*/
    FH_UINT32           invcolor;/*反色颜色  格式为 0x XX(alpha)XX(Y)XX(Cb)XX(cr)*/
}FH_OSD_COLOR;
```

~~typedef struct~~
~~{~~
~~FH_UINT32 start_line;~~
~~FH_UINT32 start_col;~~
~~}FH_OSD_STARTPOS;~~

FH_INVERT_CTRL

- 功能说明

控制 OSD 字符颜色是否反色控制位

- 定义

```
typedef struct
{
    FH_BOOL             invert_enable; /**< 是否启用反色显示 */
~~FH_UINT32 text_idex; /** < TXT OSD的序号 */~~
    FH_UINT32           invert_word[MAX_INVERT_CNT]; /** 反色控制字，每个数组元素表示1行字符，每个bit对应1字符，最多每行32个字符；注：TXT0和TXT1仅invert_word[0]有效。 */
}FH_INVERT_CTRL;
```

FH_VPU_OSD(modified)

- 功能说明

3 个字符 OSD，其中 TXT0 和 TXT1 支持单行字符内容显示，TXT2 支持 8 行字符内容显示

- 定义

```
typedef struct
{
    FH_UINT32           idex; /*字符标记，0 为 txt0,1 为 txt1,2 为 txt2*/
    FH_BOOL             osd_enable;   /**< 是否显示使能 */
    FH_UINT8            *osdtxtaddr;  /*字符的起始地址，当 txt2 时，用字符'0x0A'
作为换行符(暂未加入)*/
    FH_UINT8            *osdfontaddr; /*字符库的地址*/
    FH_INVERT_CTRL      invert_ctl;   /*字符颜色反色控制*/
    FH_OSD_CFG          osd_cfg;      /**< 配置 */
    FH_OSD_COLOR        font_color;   /**< 字符颜色 */
    FH_OSD_COLOR        blg_color;    /**< 背景颜色 */
    FH_POINT            osd_startpos; /**< 图像中显示的位置 */
}FH_VPU_OSD;
```

FH_VPU_YCMEAN

■ 功能说明

YC 均值统计结构体

■ 定义

```
typedef struct
{
    FH_UINT32           frame_id;     /**< 帧序号 */
    FH_ADDR_INFO        ymean;        /**< Y 均值地址 */
    FH_ADDR_INFO        cmean;        /**< C 均值地址 */
}FH_VPU_YCMEAN;
```

~~typedef struct~~
~~{~~
~~FH_UINT32 VO_framerate;~~
~~}FH_VPU_FRAME_CTRL;~~

FH_VPU_CROP

■ 功能说明

裁剪图片信息结构体

■ 定义

```
typedef struct
{
    FH_AREA             vpu_crop_area;    /**< 裁剪区域 */
}FH_VPU_CROP;
```

FH_VPU_YCNR_PARAMS (move 2 isp doc)

```
typedef struct
{
    FH_UINT16               CNREdgeT;/*边界阈值强度 U10*/
    FH_UINT16               CNREdgeT1;/*边界阈值强度 U10*/
    FH_UINT16               CNRYDwSkipMode;/*CNR 中的 YNR 信息下采样模式 U2*/
    FH_UINT16               CNRSigma;/*噪声方差 U8*/
    FH_UINT16               CNRAlpha;/*噪声级别 U4*/
    FH_UINT16               YNRTh;/*YNR 中的阈值 U10*/
}FH_VPU_YCNR_PARAMS;

typedef struct
{
    FH_UINT8                APCPGain;/*总体 APC 正向增益 U4.4*/
    FH_UINT8                APCNGain;/*总体 APC 负向增益 U4.4*/
    FH_UINT8                MergeSel;/*细节增强和边界锐化合并模式 U1*/
    FH_UINT8                DEPGain;/*细节增强正向增益 U7*/
    FH_UINT8                DENGain;/*细节增强负向增益 U4.4*/
    FH_UINT16               DESTHL;/* 细节增强 Slice LOW 门限值 U10*/
    FH_UINT16               DESTHH;/* 细节增强 Slice high 门限值 U10*/
    FH_UINT8                ESPGain;/*边界锐化正向增益*/
    FH_UINT8                ESNGain;/*边界锐化负向增益*/
    FH_UINT16               ESSTHL;/*边界锐化 slice low 门限值*/
    FH_UINT16               ESSTHH;/*边界锐化 slice HIGH 门限值*/
}FH_VPU_APC_PARAMS;

typedef struct
{
    FH_UINT8                *base;
    FH_UINT32                   size;
    FH_UINT32                   user_w;
    FH_UINT32                   user_h;
}FH_VPU_CHN_INFO;

typedef struct
{
    FH_UINT32               base;
    void                    *vbase;
    FH_UINT32               size;
}FH_PKG_INFO;
```

3.2.4 错误码

```
#define VPU_ERR_BASE                (-200)
#define VPU_ERR_PTR_NULL            (VPU_ERR_BASE-1)
#define VPU_ERR_SIZE_NOT_ENOUGH     (VPU_ERR_BASE-2)
#define VPU_ERR_SYS_MEM_NOT_INIT    (VPU_ERR_BASE-3)
#define VPU_ERR_CHN_MEM_NOT_INIT    (VPU_ERR_BASE-4)
#define VPU_ERR_PARAM_NOT_INIT      (VPU_ERR_BASE-5)
#define VPU_ERR_CHN_OVER_LIMITED    (VPU_ERR_BASE-6)
#define VPU_ERR_INVALID_PARAM       (VPU_ERR_BASE-7)
#define VPU_ERR_INVALID_OPERATION   (VPU_ERR_BASE-8)
#define VPU_ERR_STRUCT_NULL         (VPU_ERR_BASE-9)
#define VPU_ERR_CMD_NOT_MATCH       (VPU_ERR_BASE-10)
#define VPU_ERR_CHN_NOT_DESTORY     (VPU_ERR_BASE-11)
#define VPU_ERR_BUSY_NOW            (VPU_ERR_BASE-12)
#define VPU_ERR_TIMOUT              (VPU_ERR_BASE-13)
```

4 视频编码

4.1 功能描述

PAE(Parallel AVC Encoder)模块即 VENC 模块，视频编码模块。本模块支持多路实时编码，且每路编码独立，编码协议和编码 profile 可以不同。

支持 AVC/H.264 编码，支持 CABAC 编码、支持一个参考帧、支持数字水印、支持 ROI、支持 slice 级和宏块级速率控制、支持运动向量输出、支持 4 个 90 度方向旋转以及以宏块级行编码方式编码。

PAE 工作启动方式：PAE 外层会控制在 PAE 正式开始工作之前将 PAE 的固件搬进 AHB_RAM 中（具体搬移的方式待定），当配置用于控制 PAE 内部 CPU 启动的寄存器 PAE_CPU_CFG 为 1 时，CPU 便开始开始去读放在 AHB_RAM 中的固件并启动 PAE 的编码及转码等工作。

固件控制流程：编码器、转码器、中断处理等。PAE 的固件控制所需要完成的工作主要是配置 PAE 的全局寄存器和各 CORE 的内部寄存器，启动 PAE 各个 CORE 并行执行编码，并处理中断。

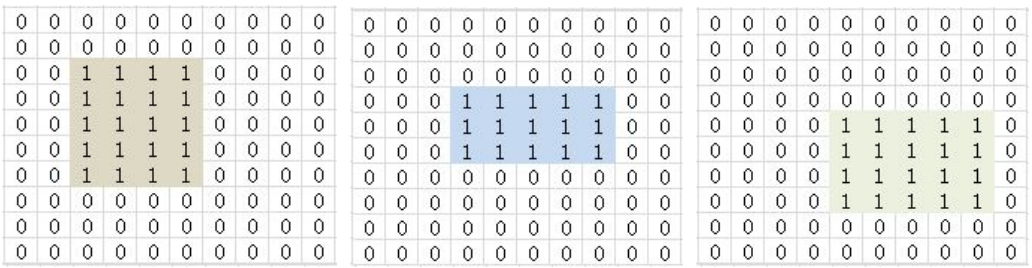
编码通道：PAE 的编码通道是个虚拟通道的概念，并不与 VPU 通道一一对应，而是通过绑定函数 FH_SYS_BindVpu2Enc 建立流的关系，可以创建最多 8 个虚拟通道。

ROI

整个图片支持 4 个 level 的 QP 值设置，每个 QP 值以 16x16 宏块为 1bit 组成的位图进行描述，从 0～3 依次从下到上覆盖：

QP0 层，为默认的 QP 层

QP1～QP3，叠加层，位图中为 1 的表示以此层 QP 值编码，为 0 的表示以默认 QP（QP0）编码



QP1 位图　　QP2 位图　　QP3 位图

图 4-1 ROI QP 层位图一、二、三

叠加以后的 ROI 控制效果为：

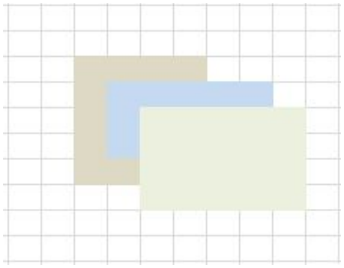


图 4-2 叠加后 QP 示意图

4.2 API 参考

视频编码模块主要提供视频编码通道的创建和销毁、开启和停止接收图像、设置和获取编码通道属性、获取和释放码流等功能。

该功能模块提供以下 API：

- ~~FH_VENC_SysInitMem:初始化系统编码内存。~~
- ~~FH_VENC_ChnInitMem:初始化通道编码内存。~~
- FH_VENC_CreateChn：创建编码通道。
- FH_VENC_DestroyChn：销毁编码通道。
- FH_VENC_StartRecvPic：开启编码通道接收输入图像。
- FH_VENC_StopRecvPic：停止编码通道接收输入图像。
- FH_VENC_ClearQueue：清除编码通道队列。
- FH_VENC_Submit_ENC：支持用户发送原始图像进行编码。
- FH_VENC_Query：查询编码通道状态。
- FH_VENC_GetStream：获取编码码流。
- FH_VENC_ReleaseStream：释放码流缓存。
- FH_VENC_SetChnAttr：设置编码通道的编码属性。
- FH_VENC_GetChnAttr：获取编码通道的编码属性。
- ~~FH_VENC_SetWaterMark：插入用户水印数据。~~
- ~~FH_VENC_GetWaterMark：获取用户水印数据。~~
- ~~FH_VENC_ClearWaterMark：清除水印数据。~~
- FH_VENC_SetRotate：设置编码图片旋转。
- FH_VENC_GetRotate：获取编码图片旋转信息。
- FH_VENC_SetRoiCfg：设置编码通道的感兴趣区域编码配置。
- FH_VENC_GetRoiCfg：获取编码通道的感兴趣区域编码配置。
- FH_VENC_ClearRoi：清除编码通道感兴趣区域块。
- FH_VENC_SetH264eRefMode：设置 H.264 编码通道跳帧参考模式。
- FH_VENC_GetH264eRefMode：获取 H.264 编码通道跳帧参考模式。
- FH_VENC_SetH264Entropy：设置熵编码模式信息。
- FH_VENC_GetH264Entropy：获取熵编码模式信息。
- FH_VENC_SET_ADV_DEBLOCKING_FILTER：设置环路滤波参数 。
- FH_VENC_GET_ADV_DEBLOCKING_FILTER： 获取环路滤波参数。
- FH_VENC_SET_ADV_INTERMBSCE：设置单系数消除模式。

- FH_VENC_GET_ADV_INTERMBSCE：获取单系数消除模式。
- FH_VENC_SET_ADV_SLICE_SPLIT：设置 Slice 分割。
- FH_VENC_GET_ADV_SLICE_SPLIT：获取 slice 分割属性。
- FH_VENC_SET_ADV_LOW_LATENCY：设置低延时模块。
- FH_VENC_GET_ADV_LOW_LATENCY：获取低延时模式信息。
- FH_VENC_RequestIDR：请求 IDR 帧。
- FH_VENC_GetCurPts：获取当前的 PTS 值。
- ~~fh_enc_init：初始化编码通道驱动。~~
- ~~fh_enc_close：关掉编码初始化驱动。~~

1. FH_VENC_SysInitMem

■ ~~功能说明~~

~~初始化编码系统内存~~

■ ~~函数定义~~

~~FH_SINT32 FH_VENC_SysInitMem()~~

■ ~~输入参数~~

~~无。~~

■ ~~输出参数~~

~~无。~~

■ ~~返回值~~

~~RETURN_OK：函数调用成功。~~

~~其他：失败，函数调用失败的原因见出错信息。~~

2. FH_VENC_CreateChn

■ 功能说明

创建编码通道，配置通道的属性值。

■ 函数定义

FH_SINT32 FH_VENC_CreateChn(FH_UINT32 chan, const FH_ENC_CHN_CONFIG *stVencChnConf);

■ 输入参数

chan：编码通道号。

stVencChnConf：通道参数指针。

■ 输出参数

无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

3. FH_VENC_StartRecvPic

- 功能说明

  开始接收图片输入进行编码。

- 函数定义

  FH_SINT32　FH_VENC_StartRecvPic(FH_UINT32 chan)

- 输入参数

  chan：编码通道号。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

4. FH_VENC_StopRecvPic

- 功能说明

  停止编码。

- 函数定义

  FH_SINT32　FH_VENC_StopRecvPic(FH_UINT32 chan)

- 输入参数

  chan：编码通道号。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

5. FH_VENC_ClearQueue

■ 功能说明

清除编码队列。

■ 函数定义

FH_SINT32 FH_VENC_ClearQueue(FH_UINT32 chan)

■ 输入参数

chan：编码通道号。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

6. FH_VENC_Submit_ENC

■ 功能说明

VENC 没有和 VPU 进行通道绑定，使用用户提供图片进行编码。

■ 函数定义

FH_SINT32 FH_VENC_Submit_ENC(FH_UINT32 chan, const FH_ENC_FRAME *pstVencsubmitframe)

■ 输入参数

chan：编码通道号。

pstVencsubmitframe：用户提供的图片信息指针。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

7. FH_VENC_Query

- 功能说明

  查询编码器状态信息。

- 函数定义

  FH_SINT32　FH_VENC_Query(FH_ENC _SYS_STATUS *pstVencstatus)

- 输入参数

  无。

- 输出参数

  pstVencstatus：编码状态信息指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

8. FH_VENC_GetStream

- 功能说明

  获取编码通道码流。

- 函数定义

  FH_SINT32 FH_VENC_GetStream (FH_UINT32 chan ,FH_ENC_STREAM_ELEMENT *pstVencstreamAttr)

- 输入参数

  chan：编码通道号。

- 输出参数

  pstVencstreamAttr：编码通道码流信息指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9. FH_VENC_ReleaseStream

- 功能说明

释放码流缓存。

- 函数定义

  FH_SINT32　FH_VENC_ReleaseStream(FH_UINT32 chan)

- 输入参数

  chan：编码通道号。

- 输出参数

  无

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

10. FH_VENC_SetChnAttr

- 功能说明

  设置编码通道属性。

- 函数定义

  FH_SINT32 FH_VENC_SetChnAttr(FH_UINT32 chan , cosnt FH_ENC_CHN_CONFIG * pstVencChnAttr)

- 输入参数

  chan：编码通道号。

  pstVencChnAttr：编码通道参数指针

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

11. FH_VENC_GetChnAttr

- 功能说明

  获取编码通道的编码属性。

- 函数定义

  FH_SINT32 FH_VENC_GetChnAttr(FH_UINT32 chan , FH_ENC_CHN_CONFIG * pstVencChnAttr)

- 输入参数

  chan：编码通道号。

- 输出参数

pstVencChnAttr：编码通道参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

12. FH_VENC_SetRotate

- 功能说明

  设置图像旋转。

- 函数定义

  FH_SINT32 FH_VENC_SetRotate(FH_UINT32 chan, const FH_ROTATE *pstVencrotateinfo)

- 输入参数

  chan：编码通道号。

  pstVencrotateinfo：旋转信息指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

13. FH_VENC_GetRotate

- 功能说明

  获取图像旋转信息。

- 函数定义

  FH_SINT32 FH_VENC_GetRotate(FH_UINT32 chan, FH_ROTATE *pstVencrotateinfo)

- 输入参数

  chan：编码通道号。

- 输出参数

  pstVencrotateinfo：旋转信息指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

14. FH_VENC_SetRoiCfg

- 功能说明

  设置编码图像的 ROI 信息。

- 函数定义

  FH_SINT32　FH_VENC_Set_RoiCfg(FH_UINT32 chan, const FH_ROI *pstVencroiinfo)

- 输入参数

  chan：编码通道号。

  pstVencroiinfo：ROI 信息指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

15. FH_VENC_GetRoiCfg

- 功能说明

  获取编码图像的 ROI 信息。

- 函数定义

  FH_SINT32　FH_VENC_GetRoiCfg(FH_UINT32 chan, FH_ROI *pstVencroiinfo)

- 输入参数

chan：编码通道号。

- 输出参数

  pstVencroiinfo：ROI 信息指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

16. FH_VENC_ClearRoi

- 功能说明

  清除编码通道的 ROI 信息。

- 函数定义

  FH_SINT32　FH_VENC_ClearRoi(FH_UINT32 chan)

- 输入参数

  chan：编码通道号。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

17. FH_VENC_SetH264eRefMode

- 功能说明

  设置 H.264 编码通道跳帧参考模式。

- 函数定义

  FH_SINT32　FH_VENC_SetH264eRefMode(FH_UINT32　chan,　FH_REF_MODE_OPS Vencreferencemode)

- 输入参数

  chan：编码通道号。

  Vencreferencemode：跳帧模式参数值。

- 输出参数

无。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息

18. FH_VENC_GetH264eRefMode

- 功能说明

获取 H.264 编码通道跳帧参考模式。

- 函数定义

FH_SINT32 FH_VENC_GetH264eRefMode(FH_UINT32 chan, FH_REF_MODE_OPS *pVencreferencemode)

- 输入参数

chan：编码通道号。

- 输出参数

pVencreferencemode：跳帧模式参数值指针。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息

19. FH_VENC_SetH264Entropy

- 功能说明

设置熵编码模式。

- 函数定义

FH_SINT32 FH_VENC_SetH264Entropy(FH_UINT32 chan, const FH_ENTROPY_OPS *pstVencentropy)

- 输入参数

chan：编码通道号。

pstVencentropy：熵编码的参数指针。

- 输出参数

无。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息

20. FH_VENC_GetH264Entropy

- 功能说明

  获取熵编码模式的参数。

- 函数定义

  FH_SINT32 FH_VENC_GetH264Entropy(FH_UINT32 chan, FH_ENTROPY_OPS *pstVencentropy)

- 输入参数

  chan：编码通道号。

- 输出参数

  pstVencentropy：熵编码的参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

21. FH_VENC_SetAdvDeblockingFilter

- 功能说明

  设置环路滤波参数。

- 函数定义

  FH_SINT32 FH_VENC_SetAdvDeblockingFilter(FH_UINT32 chan, const FH_DEBLOCKING_FILTER_PARAM *pstVencfilter)

- 输入参数

  chan：编码通道号。

  pstVencfilter：环路滤波参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

22. FH_VENC_GetAdvDeblockingFilter

- 功能说明

  获取环路滤波参数。

- 函数定义

  FH_SINT32 FH_VENC_GetAdvDeblockingFilter(FH_UINT32 chan, FH_DEBLOCKING_FILTER_PARAM *pstVencfilter)

- 输入参数

  chan：编码通道号。

- 输出参数

  pstVencfilter： 环路滤波参数指针。

- 返回值

  RETURN_OK：函数调用成功。
  其他：失败，函数调用失败的原因见出错信息

23. FH_VENC_SetAdvIntermbsce

- 功能说明

  设置单系数消除模式。

- 函数定义

  FH_SINT32 FH_VENC_ SetAdvIntermbsce(FH_UINT32 chan, FH_INTERMBSCE_OPS Vencintermbsce)

- 输入参数

  chan：编码通道号。

  Vencintermbsce：单系数消除模式参数值。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。
  其他：失败，函数调用失败的原因见出错信息

24. FH_VENC_GetAdvIntermbsce

- 功能说明

  获取单系数消除模式参数。

- 函数定义

  FH_SINT32 FH_VENC_GetAdvIntermbsce(FH_UINT32 chan, FH_INTERMBSCE_OPS Vencintermbsce)

- 输入参数

  chan：编码通道号。

- 输出参数

  Vencintermbsce： 单系数消除模式参数值。

- 返回值

  RETURN_OK：函数调用成功。
  其他：失败，函数调用失败的原因见出错信息

25. FH_VENC_SetAdvSliceSplit

- 功能说明

  设置 Slice 分割。

- 函数定义

  FH_SINT32 FH_VENC_SetAdvSliceSplit(FH_UINT32 chan, const FH_SLICE_SPLIT Vencslicesplit)

- 输入参数

  chan：编码通道号。

  Vencslicesplit：Slice 分割属性值。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。
  其他：失败，函数调用失败的原因见出错信息

26. FH_VENC_GetAdvSliceSplit

- 功能说明

  获取 Slice 分割参数。

- 函数定义

  FH_SINT32 FH_VENC_GetAdvSliceSplit(FH_UINT32 chan, FH_SLICE_SPLIT *Vencslicesplit)

- 输入参数

  chan：编码通道号。

- 输出参数

  Vencslicesplit：Slice 分割属性值指针。

- 返回值

  RETURN_OK：函数调用成功。
  其他：失败，函数调用失败的原因见出错信息

27. FH_VENC_RequestIDR

- 功能说明

  请求 I 帧。

- 函数定义

  FH_SINT32 FH_VENC_RequestIDR(FH_UINT32 chan)

- 输入参数

  chan：编码通道号。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。
  其他：失败，函数调用失败的原因见出错信息

28. FH_VENC_GetCurPts

- 功能说明

  获取当前的系统 PTS 值。

- 函数定义

  FH_SINT32　FH_VENC_GetCurPts(FH_UINT64 *Systemcurpts)

- 输入参数

  无。

- 输出参数

  Systemcurpts：当前的 PTS 值。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息

4.2.1　5.3 数据结构

```
typedef struct
{
    FH_UINT32                   chan;
    FH_UINT32                   level;
}PAE_CHN_HL;
```

FH_ENC_FRAME

```
typedef struct
{
    FH_ADDR                     lumma_addr; /**< YUV 亮度数据指针  */
    FH_ADDR                     chroma_addr; /**< YUV 色度数据指针  */
    FH_UINT64                   time_stamp;
    FH_SIZE                     size;
}FH_ENC_FRAME;
```

FH_ENC_PROFILE_IDC

- 功能说明

编码规格

- 定义

```
typedef enum
{
    FH_PRO_BASELINE,   /**< baseline    */
    FH_PRO_MAIN,       /**< main profile */
}FH_ENC_PROFILE_IDC;
```

FH_ENC_CHN_ATTR

■ 功能说明

编码器通道配置

■ 定义

```
typedef struct
{
    FH_BOOL                 chn_en; /**<  通道使能  */
    FH_ENC_PROFILE_IDC      profile; /**<  编码规格  */
    FH_UINT32               i_frame_intterval; /**< I 帧间隔，即 GOP 长度  */
    FH_SIZE                 size; /**<  图片尺寸，即编码分辨率*/
}FH_ENC_CHN_ATTR;
```

FH_ENC_RC_MODE

■ 功能说明

编码码流控制模式

■ 定义

```
typedef enum{
    FH_RC_VBR,
    FH_RC_CBR,
    FH_RC_FIXEQP,
}FH_ENC_RC_MODE;
```

FH_ENC_RC_LEVEL

■ 功能说明

编码码流控制等级

■ 定义

```
typedef enum{
    FH_RC_VERYLOW,
    FH_RC_LOW,
    FH_RC_MIDDLE,
    FH_RC_HIGH,
```

```
    FH_RC_VERYHIGH, //* ++
}FH_ENC_RC_LEVEL;
```

FH_RC_CONFIG

■ 功能说明

码控配置

■ 定义

```
typedef struct
{
    FH_ENC_RC_MODE          RCmode;
    FH_ENC_RC_LEVEL         RClevel;
    FH_UINT32               bitrate;
    FH_UINT32               IminQP;
    FH_UINT32               ImaxQP;
    FH_UINT32               PminQP;
    FH_UINT32               PmaxQP;
    FH_UINT32               max_delay; /**< 延时 [1..60] */
    FH_FRAMERATE            FrameRate; /**< 编码输出帧率 */
}FH_ PAE_RC_CONFIG;
```

FH_ENC_CHN_CONFIG

■ 功能说明

编码通道配置信息

■ 定义

```
typedef struct
{
    FH_UINT32               init_qp; /**< 初始 QP 值，取值范围 [0..51] */
    FH_ENC_CHN_ATTR         chn_attr;
    FH_ENC_RC_CONFIG        rc_config;
}FH_ENC_CHN_CONFIG;
```

FH_ENC_SYS_STATUS

■ 功能说明

编码器状态

■ 定义

```
typedef struct
{
```

```
    FH_UINT32                   totalfrmcnt;    /**< 总共完成编码帧数 */
    FH_UINT32                   totalstreamcnt;/**< 所有输出队列中待获取的帧数 */
}FH_ENC_SYS_STATUS;
```

FH_ENC_CHN_STATUS

■ 功能说明

编码通道状态

■ 定义

```
typedef struct
{
    FH_UINT32                   fps;
    FH_UINT32                   bps;
    FH_UINT32                   FrameToEnc; //待编码帧数
    FH_UINT32                   framecnt; //已编码帧数
}FH_ PAE_CHN_STATUS;
```

FH_ENC_NALU_TYPE

```
typedef enum {
    NALU_P_SLICE =0 ,
    NALU_I_SLICE = 2,
    NALU_IDR = 5,
    NALU_SEI = 6,
    NALU_SPS = 7,
    NALU_PPS = 8,
    NALU_AUD = 9,
} FH_ENC_NALU_TYPE;
```

FH_ENC_STREAM_NALU

■ 功能说明

■ 定义

```
typedef struct
{
    FH_ENC_NALU_TYPE    type;       /**< NALU 类型 */
    FH_UINT32           length;     /**< 长度，4 字节对齐*/
    FH_UINT8            *start;     /**< 起始地址，4 字节对齐不保障，打包时注意 */
}FH_ENC_STREAM_NALU;
```

FH_ENC_SLICE_TYPE

■ 功能说明

帧类型定义

■ 定义

```
typedef enum {
    P_SLICE =0 ,
    I_SLICE = 2,
} FH_ENC_SLICE_TYPE;
```

FH_ENC_STREAM_ELEMENT

■ 功能说明

■ 定义

```
typedef struct
{
    FH_UINT32               chan;          /**< 通道号   */
    FH_ADDR                 start;         /**< 码流起始地址 */
    FH_ENC_SLICE_TYPE       frame_type;    /**< 帧类型   */
    FH_UINT32               length;        /**< 数据长度 */
    FH_UINT64               time_stamp;    /**< 帧时间戳 */
    FH_UINT32               nalu_cnt;      /**< NALU 数量 */
    FH_ENC_STREAM_NALU      nalu[PAE_MAX_NALU_CNT];   /**< nalu 信息   */
}FH_ENC_STREAM_ELEMENT;

typedef struct
{
    FH_UINT8                *base;
    FH_UINT32               size;
}FH_ENC_MEM;
```

FH_ROTATE_OPS

■ 功能说明

■ 定义

```
typedef enum
{
    FH_RO_OPS_0         = 0,
    FH_RO_OPS_90        = 1,
    FH_RO_OPS_180       = 2,
    FH_RO_OPS_270       = 3
```

}FH_ROTATE_OPS;

FH_ROTATE

■ 功能说明

■ 定义

```
typedef struct
{
    FH_UINT32           enable;
    FH_ROTATE_OPS       rotate;
}FH_ROTATE;

typedef enum
{
    FH_ROI_L0       = 0,
    FH_ROI_L1       = 1,
    FH_ROI_L2       = 2,
    FH_ROI_L3       = 3
}FH_ROI_LEVEL;

typedef struct
{
    FH_AREA                 area;
    FH_ROI_LEVEL            level;
}FH_ROI_AREA;

typedef struct
{
    FH_UINT32               qp;
    FH_ROI_LEVEL            level;
}FH_ROI_QP;
```

**FH_ROI *

■ 功能说明

■ 定义

```
typedef struct
{
    FH_UINT32               enable;
    FH_UINT32               qp;
    FH_AREA                 area;
    FH_ROI_LEVEL            level;
```

```
}FH_ROI;
```

FH_REF_MODE_OPS

- 功能说明
- 定义

```
typedef enum
{
    FH_REF_MODE_1X    = 0,
    FH_REF_MODE_2X    = 1,
    FH_REF_MODE_4X    = 2,
}FH_REF_MODE_OPS;
```

FH_ENTROPY_MODE

- 功能说明
- 定义

```
typedef enum
{
    FH_CAVLC   = 0,
    FH_CABAC   = 1,
}FH_ENTROPY_MODE;
```

FH_CACBC_INIT_IDC

- 功能说明
- 定义

```
typedef enum
{
    FH_IDC_0    = 0,
    FH_IDC_1    = 1,
    FH_IDC_2    = 2,
}FH_CACBC_INIT_IDC;
```

FH_ENTROPY_OPS

- 功能说明
- 定义

```
typedef struct
{
```

```
    FH_ENTROPY_MODE          entropy_coding_mode;
    FH_CACBC_INIT_IDC        cabac_init_idc;
}FH_ENTROPY_OPS;
```

FH_DEBLOCKING_FILTER_PARAM

■ 功能说明

■ 定义

```
typedef struct
{
    FH_UINT32                deblocking_filter;//0,1
    FH_UINT32                disable_deblocking;//0,1,2
    FH_UINT32                slice_alpha;//-6~+6 :   S4
    FH_UINT32                slice_beta;//-6~+6 :    S4
}FH_DEBLOCKING_FILTER_PARAM;
```

FH_SLICE_SPLIT

■ 功能说明

■ 定义

```
typedef struct
{
    FH_UINT32                enable;
    FH_UINT32                slicesplit;
}FH_SLICE_SPLIT;

typedef struct
{
    FH_UINT32                user_w;
    FH_UINT32                user_h;
    FH_MEM_INFOs             mem_info;
}FH_ENC_CHN_INFO;
```

4.2.2 5.4 错误码

```
#define PAE_ERR_BASE                (-400)
#define PAE_ERR_INVALID_CHAN        (PAE_ERR_BASE-1)
#define PAE_ERR_INVALID_PARAM       (PAE_ERR_BASE-2)
#define PAE_ERR_MEM_SIZE_ERROR      (PAE_ERR_BASE-3)
#define PAE_ERR_MEM_PTR_NULL        (PAE_ERR_BASE-4)
#define PAE_ERR_TIME_OUT            (PAE_ERR_BASE-5)
#define PAE_ERR_FW_UNREADY          (PAE_ERR_BASE-6)
```

```
#define PAE_ERR_SYS_NOT_INIT            (PAE_ERR_BASE-7)
#define PAE_ERR_NOT_STOP            (PAE_ERR_BASE-8)
#define PAE_ERR_CHAN_NOT_CREATED        (PAE_ERR_BASE-9)
#define PAE_ERR_CHAN_HAS_CREATED        (PAE_ERR_BASE-10)
#define PAE_ERR_CHAN_UNREADY            (PAE_ERR_BASE-11)
#define PAE_ERR_PARAM_NOT_INIT          (PAE_ERR_BASE-12)
#define PAE_ERR_CHIP_NOT_SUPPORT        (PAE_ERR_BASE-13)
#define PAE_ERR_QUEUE_FULL          (PAE_ERR_BASE-14)
#define PAE_ERR_QUEUE_EMPTY             (PAE_ERR_BASE-15)
#define PAE_ERR_WAITID_MISS             (PAE_ERR_BASE-16)
#define PAE_ERR_NOT_MATCH               (PAE_ERR_BASE-17)
#define PAE_ERR_STRUCT_NULL         (PAE_ERR_BASE-18)
#define PAE_ERR_CMD_NOT_MATCH       (PAE_ERR_BASE-19)
#define PAE_ERR_ILLEGAL_OPERATION       (PAE_ERR_BASE-20)
#define PAE_ERR_CHN_NOT_DESTORY         (PAE_ERR_BASE-21)
```

5 移动侦测

5.1 功能描述

5.2 API 参考

6 内容叠加

6.1 功能描述

VPU 中的 TEXT 字符叠加模块有下面几点：

1. TXT0 和 TXT1，指三个 TXT 窗口的中二个一行字符显示窗口，最大支持 32*1，由于支持旋转，故也支持 1*32；

2. TXT2，指可显示最大连续 8 行字符的显示窗口，最大支持 32*8，由于支持旋转，故也支持 8*32，由于只能设置窗口位置，8 行字符上下之间是连续的，只可整体设置位置~~，与海康目前的字符叠加方案不一致，与安霸方案类似~~；

3. 每行字符个数最多为 32 个~~，与海康目前最多 44 个字符不一致~~；

4. 点阵字符的叠加有硬件实现，用户只需输入“字符串”和“字库”；这里的字符串有别与 C 代码实现中的机内码字符，指这个字符在所设置字库中的索引。这里的字库也不是 GB2312/GBK 等标准字库，需要根据显示的字体动态生成；

5. TXT 窗口输入的字库类型必须一致，即全部点阵类型等宽等高，现有 ASC 字库 8*16，中文字库 16*16 无法实现，必须把 ASC 字库也转换为 16*16，这样会造成中英文结合的字符串中，英文字符前后会有一个 ASC 字符的间隙；

6. 实现方式上考虑，时间显示行的设计上需根据不同语言类型设计一个包含全年、月、日、星期、数字等字体的特色字库，时间的更新变成 TXT 中“字符串”索引的更新（也可以设计更新特殊字库）。字符叠加的实现，变成了固定“字符串”索引，动态修改和生成特殊字库。

6.2 API 参考

OSD 功能。

该功能模块提供以下 API：

- FH_VI_OSD_SysInit:初始化 OSD。
- FH_VI_OSD_SysDeInit:
- FH_VI_OSD_SetFontLib
- FH_VI_OSD_SetText
- FH_VI_OSD_ClearText
- FH_VI_OSD_SetLogo
- FH_VI_OSD_ClearLogo
- FH_VI_OSD_SetMask
- FH_VI_OSD_ClearMask

```
FH_SINT32 FH_VI_OSD_SysInit(FH_UINT32 enc_width, FH_UINT32 enc_height);

FH_SINT32 FH_VI_OSD_SysDeInit();

FH_SINT32 FH_VI_OSD_SetFontLib(FH_UINT8 *asc, FH_UINT8 *charset);

FH_SINT32 FH_VI_OSD_SetText(FH_OSD_CONFIG *pOsdInit);

FH_SINT32 FH_VI_OSD_ClearText();

FH_SINT32 FH_VI_OSD_SetLogo(FH_LOGO_PARAM    *plogoParam, FH_UINT8 *graph);

FH_SINT32 FH_VI_OSD_ClearLogo();

FH_SINT32 FH_VI_OSD_SetMask(FH_UINT32 index, FH_MASK_CONFIG    *maskParam);

FH_SINT32 FH_VI_OSD_ClearMask(FH_UINT32 index);
```

7 视频输出

7.1 功能描述

VOU 是一个视频输出设备，在 CPU 的启动下可以从 DDR 中获得图像数据，输出 BT.656 数字视频，通过 TVE 和 DAC 形成 CVBS 模拟信号。

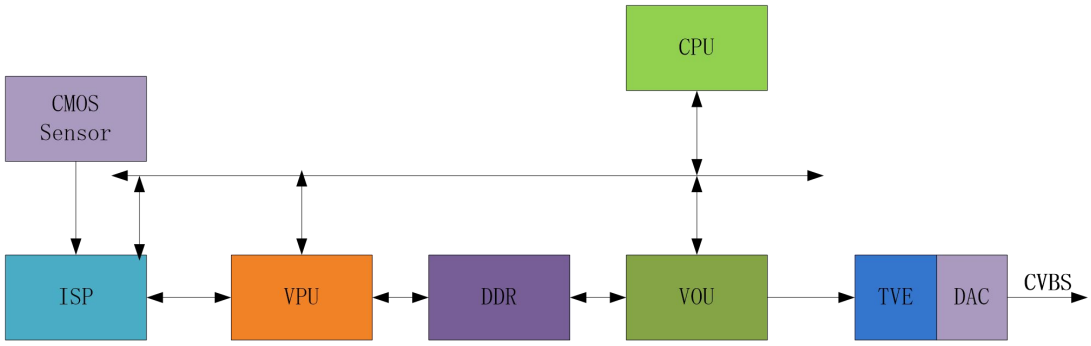


图 3-1 VOU 模块在系统中的位置

视频输出操作，当 CPU 通过 AHB 对 VOU 的寄存器进行配置后，VOU 即开始工作，VOU 启动后，通过 AXI 总线从 DDR 中读取图像数据，读取的数据首先是存放在一个同步 FIFO（该 FIFO 的写数据和读数据位宽可调）中，内部 memory 再从 FIFO 中读取图像数据(内部 memory 最多能缓存 2 行的图像数据)并最终输出 BT656 格式的图像数据，下面是基本框图：

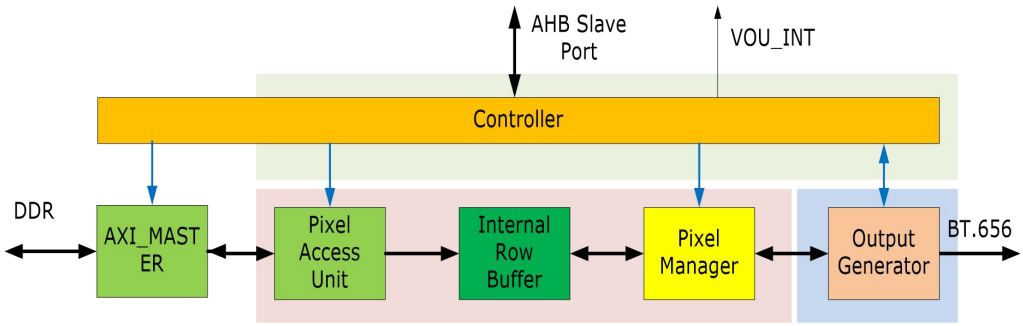


图 3-2 VOU 基本框架图

7.2 API 参考

该功能模块提供以下 MPI：

设备操作：

- FH_VOU_Enable：启用视频输出设备。
- FH_VOU_Disable：禁用视频输出设备。
- FH_VOU_SetConfig：设置视频输出设备的配置属性。
- FH_VOU_GetConfig：获取视频输出设备的配置属性。

- FH_VOU_SendFrame：用户直接发送图片信息到视频输出。
- fh_vou_init：初始化视频输出设备。
- fh_vou_close：关闭视频输出设备。

1. FH_VOU_Enable

■ 功能说明

启用视频输出设备。

■ 函数定义

FH_SINT32　FH_VOU_Enable()

■ 输入参数

无。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

2. FH_VOU_Disable

■ 功能说明

禁用视频输出设备。

■ 函数定义

FH_SINT32　FH_VOU_Disable()

■ 输入参数

无。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

3. FH_VOU_SetConfig

- 功能说明

  设置视频输出设备的配置属性。

- 函数定义

  FH_SINT32 FH_VOU_SetConfig(const FH_VOU_PIC_CONFIG * pstVouconfig)

- 输入参数

  pstVouconfig：配置参数指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

4. FH_VOU_GetConfig

- 功能说明

  获取视频输出设备的配置属性。

- 函数定义

  FH_SINT32 FH_VOU_GetConfig(FH_VOU_PIC_SIZE * pstVouconfig)

- 输入参数

  无。

- 输出参数

  pstVouconfig：配置参数指针。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

5. FH_VOU_SendFrame

- 功能说明

  支持用户直接发送图片信息到视频输出设备，当视频输出与视频处理单元通道绑定后，

此函数无效。

- 函数定义

  FH_SINT32 FH_VOU_SendFrame(const FH_VOU_PIC_INFO * pstVouconfig)

- 输入参数

  pic_info：视频输出帧信息。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

7.3 数据结构

FH_VOU_PIC_CONFIG *

- 功能说明
- 定义

```
typedef struct
{
    FH_UINT32        mode; /**< 表示 VOU 输出模式 PAL 制 720 = 0 ，NTSC 制 = 1,960 暂不支持 */

} FH_VOU_PIC_CONFIG;
```

FH_VOU_PIC_SIZE *

- 功能说明
- 定义

```
typedef struct
{
    FH_SIZE     vou_size;
}FH_VOU_PIC_SIZE;
```

FH_VOU_PIC_INFO

- 功能说明

■ 定义

```
typedef struct
{
    FH_ADDR             yaddr;
    FH_ADDR             caddr;
    FH_UINT32           ystride;
    FH_UINT32           cstride;
    FH_SIZE             vou_size;
}FH_VOU_PIC_INFO;
```

7.4 错误码

```
#define VOU_ERR_BASE                    (-300)
#define VOU_ERR_STRUCT_NULL             (VOU_ERR_BASE-0)
#define VOU_ERR_CMD_NOT_MATCH           (VOU_ERR_BASE-1)
#define VOU_ERR_INVALID_PARAM           (VOU_ERR_BASE-2)
#define VOU_ERR_INVALID_OPERATION       (VOU_ERR_BASE-3)
```

8 JPEG 抓图

8.1 功能描述

抓图是 JPEG 模块针对视频处理输出的 YUV 进行压缩的过程。本平台最大支持 720P 抓图，暂时不支持 MJPEG 实时编码。有 JPEG Wrapper 模块完成 JPEG 编码器的数据输入输出以及配置接口的适配。

其主要功能包括：

- 从视频处理输出帧缓冲读取数据，并将其转为适用于 JPEG 编码器的输入格式。
- 将来自 JPEG 编码器的码流数据转换 AXI 接口，并将其发送写入 DDR 内的缓冲
- 将来自 AHB 的配置寄存器访问请求转换为适合 JPEG 编码器的访问格式。

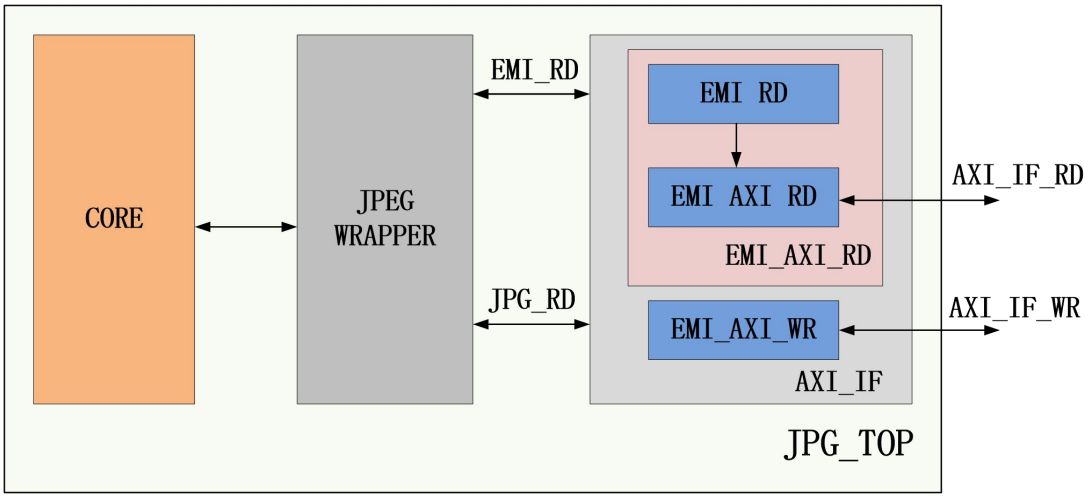


图 2-1 JPEG 模块框架

JPEG 功能操作流程（参考 FH81：JPEG Wrapper 规格定义）：

JPEG wrapper 允许在 JPEG 编码器不工作时自动关闭其时钟以降低功耗。此时所有的配置信息都被保存在 wrapper 中。

当对命令寄存器写入时，wrapper 将按以下次序工作：

- 启动编码器时钟
- 复位编码器
- 将配置载入编码器
- 启动编码器
- 待编码完成后切断时钟
- 对输出缓冲区执行 flush 操作
- 输出中断表示编码操作结束

当在编码中途发出编码停止的操作，则 wrapper 将会按以下次序工作：

- 发出停止命令
- 等待编码器报告停止
- 切断编码器时钟
- 对输出缓冲区执行 flush 操作

以上操作均为 wrapper 自动执行，无需外界干预。

流程图见下图，图中 sleep 表示编码器时钟使能信号。

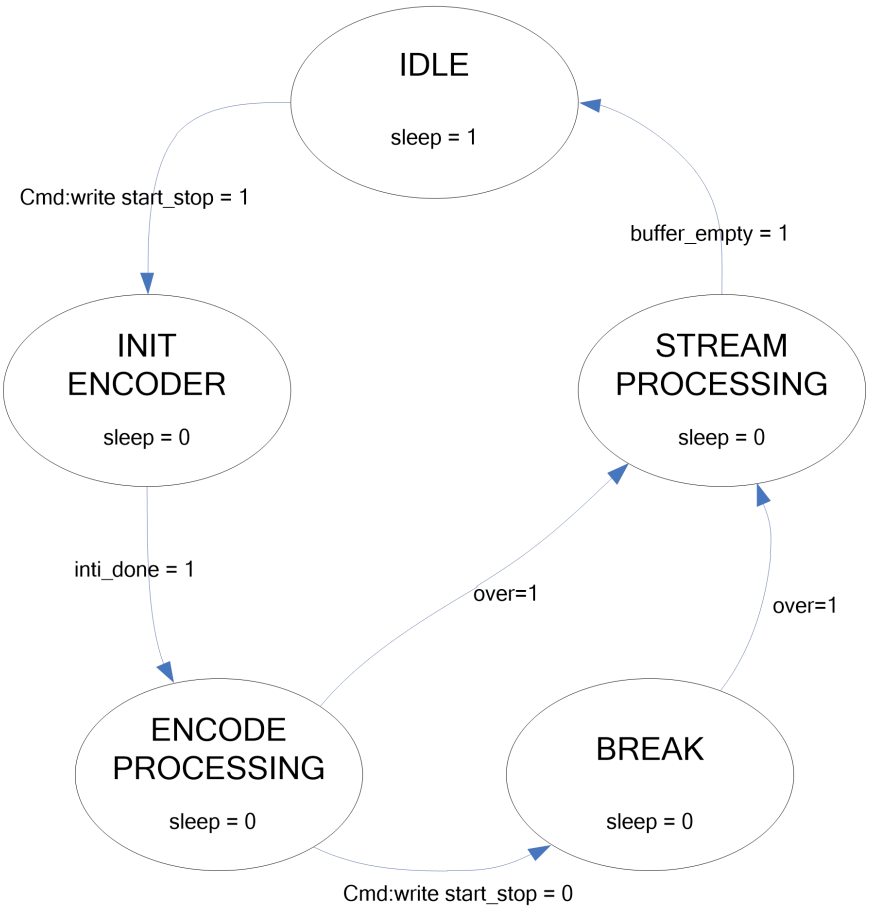


图 2-2 JPEG 编码流程框架

8.2 API 参考

该功能模块提供以下 MPI：

- FH_JPEG_InitMem： 配置 JPEG 模块需要的内存。
- FH_JPEG_Setconfig：设置 JPEG 的相关配置。
- FH_JPEG_Getconfig：获取 JPEG 的相关配置。
- FH_JPEG_Setqp：设置 JPEG 的 QP 值。
- FH_JPEG_Getqp：获取 JPEG 设置的 QP 值。
- FH_JPEG_Setstream：设置 JPEG 编码的码流信息。
- FH_JPEG_Getstream：获取 JPEG 编码的码流信息。
- fh_jpeg_init：初始化 JPEG 驱动模块。
- fh_jpeg_close：关闭 JPEG 驱动模块。

6. FH_JPEG_InitMem

■ 功能说明

配置 JPEG 模块需要的内存。

- 函数定义

  FH_SINT32　FH_JPEG_InitMem(FH_UINT32 Jpegwidth,FH_UINT32 Jpeghight);

- 输入参数

  Jpegwidth：JPEG 编码的宽度。

  Jpeghight：JPEG 编码的高度。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

7. FH_JPEG_Setconfig

- 功能说明

  设置 JPEG 配置参数。

- 函数定义

  FH_SINT32　FH_JPEG_Setconfig(const FH_JPEG_CONFIG *pstJpegconfig)

- 输入参数

  pstJpegconfig：JPEG 编码配置参数。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

8. FH_JPEG_Getconfig

- 功能说明

  获取 JPEG 配置参数。

- 函数定义

  FH_SINT32　FH_JPEG_Getconfig(FH_JPEG_CONFIG *pstJpegconfig)

- 输入参数

  无。

- 输出参数

  pstJpegconfig：JPEG 编码配置参数。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9. FH_JPEG_Setqp

- 功能说明

  设置 JPEG 编码的 QP 值，动态调整。

- 函数定义

  FH_SINT32　FH_JPEG_Setqp(FH_UINT32 QP)

- 输入参数

  QP：　JPEG 编码的 QP 值。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

10. FH_JPEG_Getqp

- 功能说明

  获取 JPEG 编码的 QP 值。

- 函数定义

  FH_SINT32　FH_JPEG_Getqp(FH_UINT32 *QP)

- 输入参数

  无。

- 输出参数

QP：JPEG 编码的 QP 值。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

11. FH_JPEG_Setstream

- 功能说明

提交一帧图像给 JPEG 编码。

- 函数定义

FH_SINT32　FH_JPEG_Setstream(const FH_JPEG_FRAME_INFO *pstJpegframe)

- 输入参数

pstJpegframe：编码的码流信息。

- 输出参数

无。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

12. FH_JPEG_Getstream

- 功能说明

获取 JPEG 编码的码流信息。

- 函数定义

FH_SINT32　FH_JPEG_Getstream(FH_JPEG_STREAM_INFO　*pstJpegframe)

- 输入参数

无。

- 输出参数

pstJpegconfig：编码的码流信息。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

8.3 2.3 数据结构

FH_JPEG_CONFIG *

- 功能说明

- **定义**

```
typedef struct
{
    FH_SINT32           QP; //0-98
    FH_SINT32           rate;//JPEG  完成速度控制  28bit
}FH_JPEG_CONFIG;
```

FH_JPEG_FRAME_INFO *

- 功能说明
- 定义

```
typedef struct
{
    FH_SINT32           frame_id;
    FH_ADDR             YADDR;
    FH_ADDR             CADDR;
    FH_SIZE             jpeg_size;
}FH_JPEG_FRAME_INFO;
```

FH_JPEG_STREAM_INFO *

- 功能说明
- 定义

```
typedef struct
{
    FH_SINT32       frame_id;
    FH_SIZE         size;
    FH_ADDR_INFO    stream;
}FH_JPEG_STREAM_INFO;
```

8.4 2.4 错误码

```
#define JPEG_ERR_BASE                   (-100)
#define JPEG_ERR_PTR_NULL               (JPEG_ERR_BASE-0)
#define JPEG_ERR_STRUCT_NULL            (JPEG_ERR_BASE-1)
#define JPEG_ERR_CMD_NOT_MATCH          (JPEG_ERR_BASE-2)
#define JPEG_ERR_MEM_NOT_ENOUGH         (JPEG_ERR_BASE-3)
#define JPEG_ERR_INVALID_PARAM          (JPEG_ERR_BASE-4)
#define JPEG_ERR_SYS_NOT_CREATED        (JPEG_ERR_BASE-5)
#define JPEG_ERR_SYS_NOT_INIT           (JPEG_ERR_BASE-6)
#define JPEG_ERR_BUSY_NOW               (JPEG_ERR_BASE-7)
#define JPEG_ERR_STREAM_EMPTY           (JPEG_ERR_BASE-8)
```

9 音 频

音频输入输出接口分为SIO（Sonic Input/Output）接口和AIO（Audio Input/Output）接口两种类型，均用于和 Audio Codec 对接，完成声音的录制和播放。

9.1 API 参考

- FH_AC_Init：音频初始化。
- FH_AC_DeInit：音频资源释放。
- FH_AC_Set_Config：设置 AI AO 设备属性。
- FH_AC_AI_Enable：启用 AI 设备。
- FH_AC_AI_Disable：禁用 AI 设备。
- FH_AC_AI_GetFrame：获取音频帧。
- FH_AC_AI_Pause：暂停 AI 设备运行。
- FH_AC_AI_Resume：恢复 AI 设备运行。
- FH_AC_AI_SetVol：设置 AI 设备音量。
- FH_AC_AO_Enable：启用 AO 设备。
- FH_AC_AO_Disable：禁用 AO 设备。
- FH_AC_AO_SendFrame：发送 AO 音频帧。
- FH_AC_AO_Pause：暂停 AO 设备运行。
- FH_AC_AO_Resume：恢复 AO 设备运行。

9.1.1 FH_SINT32 FH_AC_Init()

■ 功能说明

初始化音频设备。

■ 输入参数

无。

■ 输出参数

无。

■ 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

9.1.2 FH_SINT32 FH_AC_DeInit()

■ 功能说明

释放音频设备所用资源。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.3 FH_SINT32 FH_AC_Set_Config(FH_AC_CONFIG *pstConfig)

- 功能说明

  设置 AI、AO 设备参数。

- 输入参数

  pstConfig：配置信息结构体指针。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.4 FH_SINT32 FH_AC_AI_Enable()

- 功能说明

  使能 AI 设备。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.5 FH_SINT32 FH_AC_AI_Disable()

- 功能说明

  禁用 AI 设备。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.6 FH_SINT32 FH_AC_AO_Enable()

- 功能说明

  使能 AO 设备。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.7 FH_SINT32 FH_AC_AO_Disable()

- 功能说明

  禁用 AO 设备。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

RETURN_OK：函数调用成功。

其他：失败，函数调用失败的原因见出错信息。

9.1.8 FH_SINT32 FH_AC_AI_Pause()

- 功能说明

  暂停 AI 设备运行。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.9 FH_SINT32 FH_AC_AI_Resume()

- 功能说明

  恢复 AI 设备运行。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.10 FH_SINT32 FH_AC_AO_Pause()

- 功能说明

  暂停 AO 设备运行。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.11 FH_SINT32 FH_AC_AO_Resume()

- 功能说明

  恢复 AO 设备运行。

- 输入参数

  无。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.12 FH_SINT32 FH_AC_AI_SetVol(FH_SINT32 volume)

- 功能说明

  设置 AI 设备音量。

- 输入参数

  volume：音量大小，必须为 0~100。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.13 FH_SINT32 FH_AC_AI_GetFrame(FH_AC_FRAME_S *pstFrame)

- 功能说明

获取 AI 设备音频数据。

- 输入参数

  无。

- 输出参数

  pstFrame：音频帧结构体指针，包含数据地址和长度。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.1.14 FH_SINT32 FH_AC_AO_SendFrame(FH_AC_FRAME_S *pstFrame)

- 功能说明

  向 AO 设备发送音频数据。

- 输入参数

  pstFrame：音频帧结构体指针，包含数据地址和长度。

- 输出参数

  无。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。

9.2 数据结构

9.2.1 FH_AC_SAMPLE_RATE_E

```
typedef enum{
    AC_SR_8K     = 8000,
    AC_SR_16K    = 16000,
    AC_SR_32K    = 32000,
    AC_SR_441K = 44100,
    AC_SR_48K    = 48000,
} FH_AC_SAMPLE_RATE_E;
```

9.2.2 FH_AC_BIT_WIDTH_E

```
typedef enum{
```

```
    AC_BW_8    = 8,    //8bit/sample
    AC_BW_16 = 16, //16bit/sample
    AC_BW_24 = 24, //24bit/sample
} FH_AC_BIT_WIDTH_E;
```

9.2.3 FH_AC_FRAME_S

```
typedef struct{
    FH_UINT32 len;
    FH_UINT8 *data;
}FH_AC_FRAME_S;
```

9.2.4 FH_AC_IO_TYPE_E

```
typedef enum{
    FH_AC_MIC_IN = 0,
    FH_AC_LINE_IN = 1,
    FH_AC_SPK_OUT = 2,
    FH_AC_LINE_OUT = 3
}FH_AC_IO_TYPE_E;
```

9.2.5 FH_AC_CONFIG

```
typedef struct {
    FH_AC_IO_TYPE_E io_type;
    FH_AC_SAMPLE_RATE_E sample_rate;
    FH_AC_BIT_WIDTH_E bit_width;
    FH_UINT32 channels;
    FH_UINT32 period_size;
    FH_UINT32 volume;
} FH_AC_CONFIG;
```

9.3 错误码

```
#define OPEN_DEVICE_ERR                    (-1)
#define MEM_NULL                           (-9)
#define PARAM_ERR                          (-5)
```

标准 C 库 errno

10 调试信息

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。

/proc/umap

【文件清单】

文件名称	描述
Sys	记录当前 SYS 模块的使用情况。
vi	视频输入模块信息。
vpu	图像处理单元信息。
pae	视频编码器信息。
vou	视频输出模块信息。
jpeg	JPEG 编码过程中，各通道的编码属性、状态以及历史信息统计

【信息查看方法】

在控制台上可以使用 cat 命令查看信息，例如cat /proc/umap/enc；也可以使用其他常用的文件操作命令。

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。

10.1 API 参考

13. FH_SINT32 FH_VPU_GetPkginfo(FH_PKG_INFO *pstVpupkginfo)

- 功能说明

  获取 VPU PKG 模式下寄存器的配置值。

- 输入参数

  无。

- 输出参数

  pstVpupkginfo：PKG 模块下寄存器配置值。

- 返回值

  RETURN_OK：函数调用成功。

  其他：失败，函数调用失败的原因见出错信息。