版本: **3.1.0 2014** 年 **08** 月

图像库接口说明

声　　明

联 系 方 式

安凯（广州）微电子技术有限公司

地址：广州科学城科学大道 182 号创新大厦 C1 区 3 楼
电话: (86)-20-3221 9000

传真: (86)-20-3221 9258

邮编: 510663

销售热线:

(86)-20-3221 9499

电子邮箱:

sales@anyka.com

主页:

http: //www. anyka.com

版 本 变 更 说 明

以下表格对于本文档的版本变更做一个简要的说明。版本变更仅限于技术内容的变更，不包括版式、格式、句法等的变更。

版本	说明	完成日期
V2.12.00	正式发布	2012 年 03 月
V3.0.0	增加 PNG 解码并任意缩小接口	2012 年 04 月
V3.1.0	增加 AK37xxC 芯片支持	2014 年 08 月

本接口说明与图像编码库对应关系

Version	Corresponding media record lib
V3.1.0	V2.12.02
V3.0.0	V2.12.01
V2.12.00	V2.12.00
V2.11.09	V2.11.09
V2.10.06	V2.10.6
V2.10.02	V2.10.02
V2.10.00	V2.10.00
V2.9.01	V2.9.01
V2.8.00	V2.8.00

目录

1 模块介绍

1.1 功能概述

图像库为上层应用层提供 WBMP、JPEG、PNG、GIF 图片解码功能，JPEG 编码功能。目前 JPEG 解码支持 Sequence DCT 格式与 Progressive DCT 格式；GIF 支持 87a 和 89a 格式；PNG 支持 1.2 版。

1.2 与其他模块关系

依赖于内存管理模块，在编解码的 API 函数调用之前，必须通过函数 Img_SetCallbackFuns 指定内存分配和释放函数。

2 使用说明

无。

3 接口说明

3.1 数据结构/格式

3.1.1 BMP文件信息头

单字节对齐

```
typedef struct tagBITMAPFILEHEADER
{
    T_U16  bfType;
    T_U32  bfSize;
    T_U16  bfReserved1;
    T_U16  bfReserved2;
    T_U32  bfOffBits;
} BITMAPFILEHEADER;
```

具体定义参见 BMP 标准格式定义的 Specication 文档，以上结构各成员的含义与 BMP 标准格式一致。

3.1.2 BMP图像信息头

单字节对齐

```
typedef struct tagBITMAPINFOHEADER
{
    T_U32  biSize;
    T_S32   biWidth;
    T_S32   biHeight;
    T_U16  biPlanes;
    T_U16  biBitCount;
    T_U32  biCompression;
    T_U32  biSizeImage;
    T_S32   biXPelsPerMeter;
    T_S32   biYPelsPerMeter;
```

```
    T_U32  biClrUsed;
    T_U32  biClrImportant;
} BITMAPINFOHEADER;
```

具体定义参见 BMP 标准格式定义的 Specication 文档,，以上结构各成员的含义与 BMP 标准格式一致。

3.1.3 图片类型

```
typedef enum tagIMAGETYPE
{
    IMG_IMAGE_UNKNOWN,
    IMG_IMAGE_BMP,              //BMP 格式图片
    IMG_IMAGE_WBMP,             //WBMP 格式图片
    IMG_IMAGE_JPG,              //JPG 格式图片
    IMG_IMAGE_GIF,              //GIF 格式图片
    IMG_IMAGE_PNG               //PNG 格式图片
}T_IMAGE_TYPE;
```

T_IMAGE_TYPE 定义图片解码库所能支持的图片类型，IMG_IMAGE_UNKNOWN 值表示该图片是图片解码库不能支持的图片类型。该枚举主要用于 Img_ImageType 函数。

3.1.4 GIF图片解码HANDLE

```
typedef T_S32 T_hGIFENUM;
#define IMG_INVALID_HANDLE 0        //无效的 handle
```

注: 对 GIF 图片进行各帧的遍历解码时, 必须先创建该 HANDLE 进行帧的遍历。

3.1.5 各种回调函数指针的类型定义

1、读文件

```
typedef T_U32 (*CALLBACK_FUN_FREAD)
```

(unsigned int hFile, void *pBuffer, unsigned int count);

2、写文件

typedef T_U32 (*CALLBACK_FUN_FWRITE)

(unsigned int hFile, void *pBuffer, unsigned int count);

3、文件定位

typedef T_U32 (*CALLBACK_FUN_FSEEK)

(unsigned int hFile, int offset, unsigned charorigin);

4、获取文件长度

typedef T_U32 (*CALLBACK_FUN_FGETLEN)(unsigned int hFile);

5、获取当前文件位置

typedef T_U32 (*CALLBACK_FUN_FTELL)(unsigned int hFile);

6、资源

```
typedef struct {
   T_U16  ResourceID;
   T_U8    *Buff;    //The Pointer to save the resource.
          // (Application should allocate the memory for it)
   T_U32  Resource_len;
} T_LOADRESOURCE_CB_PARA;
```

7、读取资源

typedef void

(*CALLBACK_FUN_LOADRESOURCE)(T_LOADRESOURCE_CB_PARA *pPara);

8、释放资源

typedef void (*CALLBACK_FUN_RELEASERESOURCE)(T_U8 *Buff);

9、分配内存

typedef void* (*CALLBACK_FUN_MALLOC)
(T_U32 size, char *filename, T_U32 line);

10、释放内存

typedef void* (*CALLBACK_FUN_FREE)(void* mem);

11、重新分配内存

typedef void* (*CALLBACK_FUN_REMALLOC)(void* mem, T_U32 size);

12、DMA 方式复制内容

typedef void* (*CALLBACK_FUN_DMAMEMCOPY)
(void* dst, void* src, T_U32 count);

13、播放时显示一帧 YUV 图像

typedef void (*CALLBACK_FUN_SHOWFRAME)
(void* srcImg, T_U32 src_width, T_U32 src_height);

14、录像时显示一帧 YUV 图像

typedef void (*CALLBACK_FUN_CAMERASHOWFRAME)
(void* srcImg, T_U32 src_width, T_U32 src_height);

15、启动 Camera 进行采集

typedef void (*CALLBACK_FUN_CAPSTART)(void);

16、查询 Camera 采集是否完成

typedef T_BOOL (*CALLBACK_FUN_CAPCOMPLETE)(void);

17、返回采集完成的 YUV 数据

typedef void* (*CALLBACK_FUN_CAPGETDATA)(void);

18、查询系统时钟，单位为毫秒

typedef T_U32 (*CALLBACK_FUN_GETTICKCOUNT)(void);

19、信息打印

typedef T_S32 (*CALLBACK_FUN_PRINTF)(T_pCSTR format, ...);

20、寄存器修改接口

typedef IMG_T_BOOL (*CALLBACK_FUN_REGMODIFY)(IMG_T_U32 addr, IMG_T_U32 value, IMG_T_U32 mask);

21、回调函数指针的数据结构定义

```
typedef struct{
    CALLBACK_FUN_FREAD          fread;
    CALLBACK_FUN_FWRITE         fwrite;
    CALLBACK_FUN_FSEEK          fseek;
    CALLBACK_FUN_FGETLEN        fgetlen;
    CALLBACK_FUN_FTELL          ftell;
    CALLBACK_FUN_LOADRESOURCE   LoadResource;
    CALLBACK_FUN_RELEASERESOURCE ReleaseResource;
    CALLBACK_FUN_MALLOC         malloc;
    CALLBACK_FUN_FREE           free;
    CALLBACK_FUN_REMALLOC       remalloc;
    CALLBACK_FUN_DMAMEMCOPY     DMAMemcpy;
    CALLBACK_FUN_SHOWFRAME      ShowFrame;
    CALLBACK_FUN_CAMERASHOWFRAME CameraShowFrame;
    CALLBACK_FUN_CAPSTART       CapStart;
    CALLBACK_FUN_CAPCOMPLETE    CapComplete;
    CALLBACK_FUN_CAPGETDATA     CapGetData;
    CALLBACK_FUN_GETTICKCOUNT   GetTickCount;
CALLBACK_FUN_PRINTF         printf;
CALLBACK_FUN_REGMODIFY regModify;
```

} CB_FUNS;

22、设置 FlushCache 函数指针

强制将 Cache 里面数据送到内存函数。

typedef IMG_T_VOID (*FlushCacheFunc)(IMG_T_VOID);

23、设置 JPEG 并行工作函数指针

typedef IMG_T_VOID (*JPEGTaskFunc)(IMG_T_VOID);

3.1.6 GIF解码模式

typedef enum

{

MIN_SPACE,

MAX_SPEED

}GIF_DEC_MODE;

3.1.7 编解码错误代码

typedef enum

{

IMG_NO_ERROR,

IMG_INPUT_NULL_POINTER,

IMG_PARAMETER_ERROR,

IMG_STREAM_ERROR,

IMG_NOT_ENOUGH_MOMORY,

IMG_NOT_SUPPORT_FORMAT

}T_IMAGE_ERROR;

3.1.8 JPEG硬件解码版本

typedef enum

{

JPEG_HW_VER1,

```
    JPEG_HW_VER2,
    JPEG_HW_VER3,
    JPEG_HW_VER4,
    JPEG_HW_VER5,
    JPEG_HW_VER6
} JPEG_HW_VERSION;

#define   AK3221_JPEG_VER          JPEG_HW_VER1
#define   AK3223_JPEG_VER          JPEG_HW_VER1
#define   AK3224_JPEG_VER          JPEG_HW_VER1
#define   AK3225_JPEG_VER          JPEG_HW_VER1
#define   AK3610_JPEG_VER          JPEG_HW_VER1
#define   AK3620_JPEG_VER          JPEG_HW_VER1
#define   AK322L_JPEG_VER          JPEG_HW_VER2
#define   AK3225L_JPEG_VER         JPEG_HW_VER2
#define   AK3631L_JPEG_VER         JPEG_HW_VER2
#define   AK3671_JPEG_VER          JPEG_HW_VER2
#define   AK3810_JPEG_VER          JPEG_HW_VER3
#define   AK7801_JPEG_VER          JPEG_HW_VER3
#define   AK7802_JPEG_VER          JPEG_HW_VER3
#define   AK980x_JPEG_VER          JPEG_HW_VER4
#define   AK37xx_JPEG_VER          JPEG_HW_VER5
#define   AK37xC_JPEG_VER          JPEG_HW_VER6
```


3.1.9 JPEG编码时嵌入的EXIF信息

```
typedef struct
{
   T_U8   ImageDescription[80];
   T_U8   Make[30];
   T_U8   Model[20];
   T_U8   Artist [20];
```

```
    T_U8    Software[40];
    T_U8    DateTime[20];          // "YYYY:MM:DD hh:mm:ss"
    T_BOOL  bThumbnail;
} T_ExifInfo;
```

3.1.10 JPEG编码时水印OSD信息

```
typedef struct
{
  T_U16 osdWidth;          // OSD 宽度，必须是 16 的倍数
  T_U16 osdHeight;         // OSD 高度，必须是 16 的倍数
  T_U16 osd_H_offset;// OSD 相对于目标背景图的水平方向偏移，必须是 16 的倍数
  T_U16 osd_V_offset;// OSD 相对于目标背景图的垂直方向偏移，必须是 16 的倍数
  T_U16 *osdRGB565;// OSD 的 RGB565 数据，存储顺序(R,G,B)；其中 0 值表示透明
                   即背景色
  T_U8  alpha;     // OSD 和背景色混合度，取值范围[0,16]。0：不透明；16：全透明
}J_OSD_Info;

// for VERRSION 6, AK37C
typedef struct
{
  IMG_T_U16 stampWidth;              // stamp 宽度，必须是 16 的倍数
  IMG_T_U16 stampHeight;             // stamp 高度，必须是 16 的倍数
  IMG_T_U16 stamp_H_offset;  // stamp 相对于目标背景图的水平方向偏移，必须是
                             16 的倍数
  IMG_T_U16 stamp_V_offset;  // stamp 相对于目标背景图的垂直方向偏移，必须是
                             16 的倍数
  IMG_T_U8 *stampYptr;       // stamp 的 Y 数据指针 YUV 三个值全 0 即透明
  IMG_T_U8 *stampUptr;       // stamp 的 U 数据指针 YUV 三个值全 0 即透明
  IMG_T_U8 *stampVptr;       // stamp 的 V 数据指针 YUV 三个值全 0 即透明
  }J_STAMP_Info;
```

3.2 JPEG解码

3.2.1 Img_JpegInfo

原　　型	T_BOOL Img_JpegInfo(const T_U8 *jpegData, T_U32 size, T_U16 *width, T_U16 *height, T_U8 *y_h_samp, T_U8 *y_v_samp)	
功能概述	取得对应 JPEG 图片的宽度和高度信息及亮度的垂直和水平采样率	
参数说明	JpegData	JPEG 的源数据指针
	size	JPEG 数据长度
	width	输出，返回 JPEG 源数据对应图片的宽度
	height	输出，返回 JPEG 源数据对应图片的高度
	y_h_sam	输出，返回亮度的水平采样率
	y_v_samp	输出，返回亮度的垂直采样率
返回值说明	AK_TRUE：获取成功，宽度和高度信息返回在 width 和 height 中，水平和垂直采样率返回在 y_h_sam 和 y_v_samp 中。 AK_FALSE：获取失败，width、height、y_h_sam、y_v_sam 中的值无效	
注意事项	调用该函数时应确保传入的 jpg_width、jpg_height 指针有效，如果不使用 y_h_samp，y_v_samp 变量，要将它们设为 AK_NULL	

调用示例	T_U8 jpegData[MAX_JPEG_LEN];
	T_U16 jpgWidth = 0;
	T_U16 jpgHeight = 0;
	T_U8 y_h_samp = 0;
	T_U8 y_v_samp = 0;
	
	if((ReadFromFile(“a:/image/example.jpg”, jpegData, MAX_JPEG_LEN))
	{
	　Img_JpegInfo(jpegData, &jpgWidth, &jpgHeight, &y_h_samp, &y_v_samp);
	}	

3.2.2 Img_Jpeg2BMP

原　型	T_U8 *Img_Jpeg2BMP(const T_U8 *jpegData, T_U32 *size);	
功能概述	将 Jpeg 图片解码成 BMP 格式	
参数说明	jpegData	Jpeg 的源数据指针
	size	输入：Jpeg 数据长度；输出：返回解码后的 BMP 数据长度
返回值说明	AK_NULL：解码失败，size 中的返回值无效 其它：解码成功，返回结果 BMP 数据缓冲区指针，长度返回在 size 中	
注意事项	解码成功得到的 BMP 数据指针在不用时必须将其释放，否则会有内存泄漏 该函数使用硬件解码	

调用示例	T_U32 bmpLen = 0;
	T_U8 jpegData[MAX_JPEG_LEN];
	T_U8 *bmpData = AK_NULL;
	if((ReadFromFile(“a:/image/example.jpg”, jpegData, MAX_JPEG_LEN))
	{
	bmpData = Img_Jpeg2BMP(jpegData, &bmpLen);
	if(bmpData)
	{
	Eng_DrawBMP(LCD_0, bmpData, 0, 0, AK_NULL);
	Img_Free(bmpData);
	bmpData = AK_NULL;
	}
	}	

3.2.3 Img_Jpeg2BMPSoft

原　型	T_U8 *Img_Jpeg2BMPSoft(const T_U8 *jpegData, T_U32 *size);	
功能概述	纯软件解码, 其它和 Img_Jpeg2BMP 一致	
参数说明	jpegData	JPEG 的源数据指针
	size	输入：JPEG 数据长度；输出：返回解码后的 BMP 数据长度

原　型	T_U8 *Img_Jpeg2BMPSoft(const T_U8 *jpegData, T_U32 *size);
返回值说明	AK_NULL：解码失败，size 中的返回值无效 其它：解码成功，返回结果 BMP 数据缓冲区指针，长度返回在 size 中
注意事项	参数和返回均与 Img_Jpeg2BMP 一致，只是用软件解码 解码成功后，指针在不用时必须将其释放, 否则会有内存泄漏
调用示例	See Img_Jpeg2BMP

3.2.4 Img_Jpeg2BMPEx

原　型	T_U8 *Img_Jpeg2BMPEx(const T_U8 *jpegData, T_U16 dstWidth, T_U16 dstHeight, T_U32 *size);	
功能概述	将 JPEG 图片解码后按给定的尺寸进行缩小，生成成 BMP 格式图片	
参数说明	jpegData	JPEG 的源数码指针
	dstWidth	指定解码后的 BMP 图片宽度
	dstHeight	指定解码后的 BMP 图片高度
	size	输入：JPEG 数据长度；输出：返回解码后的 BMP 数据长度
返回值说明	AK_NULL：解码失败，size 中的返回值无效 其它：解码成功，返回结果 BMP 数据缓冲区指针，长度返回在 size 中	
注意事项	原图片尺寸必须比指定的尺寸大，否则解码后图像大小不变。 解码成功得到的 BMP 数据指针在不用时必须用 Img_Free 将其释放，否则会有内存泄漏	

调用示例	#define BMP_HEIGHT 50
	#define BMP_WIDTH 50
	T_U32 bmpLen = 0;
	T_U8 jpegData[MAX_JPEG_LEN];
	T_U8 *bmpData = AK_NULL;
	if((ReadFromFile(“a:/image/example.jpg”, jpegData, MAX_JPEG_LEN))
	{
	bmpData = Img_Jpeg2BMPEx(jpegData, BMP_HEIGHT, BMP_WIDTH, &bmpLen);
	if(bmpData)
	{
	Eng_DrawBMP(LCD_0, bmpData, 0, 0, AK_NULL);
	Img_Free(bmpData);
	bmpData = AK_NULL;
	}
	}	

3.2.5 Img_Jpeg2BMPExSoft

原　型	T_U8 *Img_Jpeg2BMPExSoft(const T_U8 *jpegData, T_U16 dstWidth, T_U16 dstHeight, T_U32 *size);	
功能概述	通过软解码将 JPEG 图片按给定的尺寸解码成 BMP 格式图片	
参数说明	jpegData	JPEG 的源数码指针
	dstWidth	解码后的 BMP 图片宽度
	dstHeight	解码后的 BMP 图片高度
	size	输入：JPEG 数据长度；输出：返回解码后的 BMP 数据长度
返回值说明	AK_NULL：解码失败，size 中的返回值无效 其它：解码成功，返回结果 BMP 数据缓冲区指针，长度返回在 size 中	
注意事项	原图片尺寸必须比指定的尺寸大，否则解码后图像大小不变。 解码成功得到的 BMP 数据指针在不用时必须用 Img_Free 将其释放，否则会有内存泄漏	
调用示例	同 Img_Jpeg2BmpEx	

3.2.6 Img_Jpeg2YUV

原　型	T_BOOL Img_JPEG2YUV(T_U8 *srcJPEG, T_U32 size, T_U8 *dstYUV, T_S32 *width, T_S32*height)	
功能概述	将 JPEG 图片数据解码成 YUV 数据，并获得图片宽高	
参数说明	srcJPEG	JPEG 的源数据指针
	size	JPEG 数据长度
	dstYUV	解码后的 YUV 数据地址
	width	JPEG 图片宽度
	height	JPEG 图片高度
返回值说明	是否成功	
注意事项	支持 JPEG 格式 422、420、444	
调用示例	-	

3.2.7 Img_Jpeg2YUVSoft

原　型	T_BOOL Img_JPEG2YUVSoft(T_U8 *srcJPEG, T_U32 size, T_U8 *dstYUV, T_S32 *width, T_S32*height)	
功能概述	通过软解码，将 JPEG 图片数据解码成 YUV 数据，并获得图片宽高	
参数说明	srcJPEG	JPEG 的源数据指针
	size	JPEG 数据长度
	dstYUV	解码后的 YUV 数据地址
	width	JPEG 图片宽度
	height	JPEG 图片高度
返回值说明	是否成功	
注意事项	支持 JPEG 格式 422、420、444	
调用示例	-	

3.2.8 Img_Jpeg2YUV4x

原　型	T_BOOL Img_JPEG2YUV4x(T_U8 *srcJPEG, T_U32 size, T_U8 *dstYUV, T_S32 *width, T_S32*height)	
功能概述	将 JPEG 图片数据解码成 YUV 数据，缩小 1/4（宽高各缩小 1/2），并获得缩小后的图片宽高	
参数说明	srcJPEG	JPEG 的源数据指针
	size	JPEG 数据长度
	dstYUV	解码后的 YUV 数据地址
	width	JPEG 图片宽度
	height	JPEG 图片高度
返回值说明	是否成功	
注意事项	支持 JPEG 格式 422、420、444	
调用示例	-	

3.2.9 Img_Jpeg2YUVEx

原　型	T_BOOL Img_JPEG2YUVEx(T_U8 *srcJPEG, T_U32 size, T_U8 *dstYUV, T_U16 dstWidth, T_U16 dstHeight);	
功能概述	将 JPEG 图片数据解码成 YUV 数据，任意比例缩放，并获得缩小后的图片宽高	
参数说明	srcJPEG	JPEG 的源数据指针
	size	JPEG 数据长度
	dstYUV	解码后的 YUV 数据地址
	dstWidth	指定解码后的 JPEG 图片宽度
	dstHeight	指定解码后的 JPEG 图片高度
返回值说明	是否成功	
注意事项	支持 JPEG 格式 422、420、444	
调用示例	-	

3.2.10 Img_VideoJPEG2YUV

原　型	T_BOOL Img_VideoJPEG2YUV(T_U8 *srcJPEG, T_U32 size, T_U8 *dstYUV, T_S32 *width, T_S32*height)	
功能概述	将 MJPEG 码流中的 JPEG 数据解码成 YUV 数据，并获得图片宽高	
参数说明	srcJPEG	JPEG 的源数据指针
	size	JPEG 数据长度
	dstYUV	解码后的 YUV 数据地址
	width	JPEG 图片宽度
	height	JPEG 图片高度
返回值说明	是否成功	
注意事项	支持 JPEG 格式 422、420、444	
调用示例	-	

3.2.11 Img_Jpeg2RGB

原　型	T_BOOL Img_Jpeg2RGB(const T_U8 *srcJPEG, T_U8 *dstRGB, T_U32 *width, T_U32 *height, T_U32 size);	
功能概述	JPEG 图片数据解码成 RGB，并获得图像的宽度和高度	
参数说明	srcJPEG	JPEG 的源数据指针
	dstRGB	解码后的 RGB 数据地址
	width	返回的 JPEG 图片宽度指针
	height	返回的 JPEG 图片高度指针
	size	JPEG 数据长度
返回值说明	是否成功	
注意事项	支持 jpeg 格式 422、420、444； 计算 RGB 缓冲的大小公式：(width*24+31)/32*4*height	
调用示例	-	

3.2.12 Img_JpegThumbnail

原　型	T_U8 *Img_JpegThumbnail(const T_U8 *jpegData, T_U32 *size, T_U16 *width, T_U16 *height)	
功能概述	解码 JPEG 文件中的缩略图	
参数说明	jpegData	JPEG 的源数据指针
	size	输入：JPEG 数据长度；输出：返回解码后的 BMP 缩略图的数据长度
	width	缩略图的宽度
	height	缩略图的高度
返回值说明	AK_NULL：解码失败，size 中的返回值无效 其它：解码成功，返回结果 BMP 数据缓冲区指针，长度返回在 size 中	
注意事项	解码成功得到的 BMP 数据指针在不用时必须将其释放，否则会有内存泄漏 该函数搜索 APP1 段中的 EXIF 信息以及 APP13 段中的 Photoshop 8BIM 信息中的缩略图，并取先搜索到的缩略图数据进行硬件解码	
调用示例	-	

3.2.13 Img_EmbedThumbnail

原　型	T_U8 *Img_EmbedThumbnail(const T_U8 *jpegData, T_U32 *size);	
功能概述	往 JPEG 文件中嵌入缩略图	
参数说明	jpegData	JPEG 的源数据指针
	size	输入：JPEG 数据长度；输出：嵌入缩略图后的 JPEG 数据长度
返回值说明	AK_NULL：失败，size 中的返回值无效 其它：成功，返回结果 JPEG 数据缓冲区指针，长度返回在 size 中	
注意事项	解码成功得到的 JPEG 数据指针在不用时必须将其释放，否则会有内存泄漏	
调用示例	-	

3.3 JPEG编码

3.3.1 Img_YUV2JPEG

原　型	T_BOOL Img_YUV2JPEG(T_U8 *srcY, T_U8 *srcU, T_U8 *srcV, T_U8 *dstJPEG, T_U32 *size, T_U32 width, T_U32 height, T_U8 quality)	
功能概述	将 YUV 数据编码成 JPEG 图片数据	
参数说明	srcY	Y 数据指针
	srcU	U 数据指针
	srcV	V 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：JPEG 长度
	width	图像宽度
	height	图像高度
	quality	编码质量，取值范围 0-200
返回值说明	是否成功	
注意事项	输入 YUV 格式：H211（AK3810、AK78xx 芯片则为 YUV 420）； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小	
调用示例	-	

3.3.2 Img_YUV2JPEGSoft

原　型	T_BOOL Img_YUV2JPEGSoft(T_U8 *srcY, T_U8 *srcU, T_U8 *srcV, T_U8 *dstJPEG, T_U32 *size, T_U32 width, T_U32 height, T_U8 quality)	
功能概述	通过软编码，将 YUV 数据编码成 JPEG 图片数据	
参数说明	srcY	Y 数据指针
	srcU	U 数据指针
	srcV	V 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：Jpeg 长度
	width	图像宽度
	height	图像高度
	quality	编码质量，取值范围 0-200
返回值说明	是否成功	
注意事项	输入 YUV 格式：H211（AK3810、AK78xx 芯片则为 YUV 420）； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小	
调用示例	-	

3.3.3 Img_YUV2JPEG4x

原　型	T_BOOL Img_YUV2JPEG4x(T_U8 *srcYUV, T_U8 *dstJPEG, T_U32 *size, T_U32 width, T_U32 height, T_U8 quality)	
功能概述	将 YUV 数据编码成 JPEG 图片数据，并放大 4 倍（水平垂直方向各两倍）	
参数说明	srcYUV	YUV 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：JPEG 长度
	srcWidth	源图像宽度
	srcHeight	源图像高度
	Quality	质量，取值范围 0-200
返回值说明	是否成功	

原　型	T_BOOL Img_YUV2JPEG4x(T_U8 *srcYUV, T_U8 *dstJPEG, T_U32 *size, T_U32 width, T_U32 height, T_U8 quality)
注意事项	输入 YUV 格式：H211（AK3810、AK78xx 芯片则为 YUV 420）； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小
调用示例	-

3.3.4 Img_YUV2JPEGEx

原　型	T_BOOL Img_YUV2JPEGEx(T_U8 *srcYUV, T_U8 *dstJPEG, T_U32 *size, T_U32 srcWidth, T_U32 srcHeight, T_U32 dstWidth, T_U32 dstHeight, T_U8 quality)	
功能概述	将 YUV 数据编码成 JPEG 图片数据，任意比例缩放	
参数说明	srcYUV	YUV 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：JPEG 长度
	srcWidth	源图像宽度
	srcHeight	源图像高度
	dstWidth	目标图像宽度
	dstHeight	目标图像高度
	Quality	质量，取值范围 0-200
返回值说明	是否成功	
注意事项	输入 YUV 格式：H211（AK3810、AK78xx 芯片则为 YUV 420）； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小	
调用示例	-	

3.3.5 Img_YUV2JPEGExs

原　型	T_BOOL Img_YUV2JPEGExs(T_U8 *srcYUV, T_U8 *srcU, T_U8 *srcV, T_U8 *dstJPEG, T_U32 *size, T_U32 srcWidth, T_U32 srcHeight, T_U32 dstWidth, T_U32 dstHeight, T_U8 quality)	
功能概述	将 YUV 数据编码成 JPEG 图片数据，任意比例缩放	
参数说明	srcY	Y 数据指针

原　型	T_BOOL Img_YUV2JPEGExs(T_U8 *srcYUV, T_U8 *srcU, T_U8 *srcV, T_U8 *dstJPEG, T_U32 *size, T_U32 srcWidth, T_U32 srcHeight, T_U32 dstWidth, T_U32 dstHeight, T_U8 quality)	
	srcU	U 数据指针
	srcV	V 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：JPEG 长度
	srcWidth	源图像宽度
	srcHeight	源图像高度
	dstWidth	目标图像宽度
	dstHeight	目标图像高度
	Quality	质量，取值范围 0-200
返回值说明	是否成功	
注意事项	输入 YUV 格式：H211（AK3810、AK78xx 芯片则为 YUV 420）； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小	
调用示例	-	

3.3.6　Img_YUV2JPEG_OSD

原　型	IMG_T_BOOL　Img_YUV2JPEG_OSD(const T_U8 *srcY, const T_U8 *srcU, const T_U8 *srcV, T_U8 *dstJPEG, T_U32 *size, T_U32 srcWidth, T_U32 srcHeight, T_U32 dstWidth, T_U32 dstHeight, T_U8 quality,　J_OSD_Info *osdinfo);	
功能概述	将 YUV 数据编码成 JPEG 图片数据，支持水印功能。10 倍内任意放大，2 倍内任意缩小。硬件完成缩放。	
参数说明	srcY	Y 数据指针
	srcU	U 数据指针
	srcV	V 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：JPEG 长度
	srcWidth	源图像宽度

	srcHeight	源图像高度
	dstWidth	目标图像宽度
	dstHeight	目标图像高度
	Quality	质量，取值范围 0-200
	osdinfo	水印信息。见 3.1.10 节。如果不需要水印，该参数设置 NULL 即可。
返回值说明	是否成功	
注意事项	输入 YUV 格式： YUV 420； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小	
调用示例	-	

3.3.7 Img_YUV2JPEG_Stamp

原 型	IMG_T_BOOL Img_YUV2JPEG_Stamp(const IMG_T_U8 *srcY, const IMG_T_U8 *srcU, const IMG_T_U8 *srcV, IMG_T_U8 *dstJPEG, IMG_T_U32 *size, IMG_T_U32 width, IMG_T_U32 height, IMG_T_U8 quality, J_STAMP_Info *stampinfo);	
功能概述	将 YUV 数据编码成 JPEG 图片数据，支持时间戳功能。	
参数说明	srcY	Y 数据指针
	srcU	U 数据指针
	srcV	V 数据指针
	dstJPEG	JPEG 数据指针
	size	输入：dstJPEG 缓冲大小；输出：JPEG 长度
	width	目标图像宽度
	height	目标图像高度
	quality	质量，取值范围 0-200
	stampinfo	时间戳信息。见相关数据结构定义。
返回值说明	是否成功	
注意事项	输入 YUV 格式： YUV 420； 调用前需要设置 size 的最大值，通常为应用程序已经分配的 dstJPEG 缓冲大小	

原　型	IMG_T_BOOL Img_YUV2JPEG_Stamp(const IMG_T_U8 *srcY, const IMG_T_U8 *srcU, const IMG_T_U8 *srcV, IMG_T_U8 *dstJPEG, IMG_T_U32 *size, IMG_T_U32 width, IMG_T_U32 height, IMG_T_U8 quality, J_STAMP_Info *stampinfo);
调用示例	-

3.3.8 Img_EnableExif

原　型	T_VOID Img_EnableExif(T_BOOL enable);	
功能概述	使能 / 禁止 JPEG 编码时嵌入 EXIF 信息	
参数说明	enable	JPEG 编码时是否加入 EXIF 信息
	-	-
返回值说明	无	
注意事项	-	
调用示例	-	

3.3.9 Img_GetExifInfo

原　型	T_VOID Img_GetExifInfo(T_ExifInfo* exifInfo);	
功能概述	获取 JPEG 编码时嵌入的 EXIF 信息	
参数说明	exifInfo	指向 T_ExifInfo 类型的结构体的指针
返回值说明	无	
注意事项	结构体类型 T_ExifInfo 参见 3.1.9 节	
调用示例	T_ExifInfo exifInfo; Img_GetExifInfo(&exifInfo);	

3.3.10 Img_SetExifInfo

原　型	T_VOID Img_SetExifInfo(T_ExifInfo* exifInfo);	
功能概述	设置 JPEG 编码时嵌入的 EXIF 信息	
参数说明	exifInfo	指向 T_ExifInfo 类型的结构体的指针

原　　型	T_VOID Img_SetExifInfo(T_ExifInfo* exifInfo);
返回值说明	无
注意事项	结构体类型 T_ExifInfo 参见 3.1.9 节

调用示例	T_ExifInfo exifInfo;
	char MyDateTime[20] = "2008:10:22 16:18:54";
	
	Img_EnableExif(AK_TRUE);
	Img_GetExifInfo(&exifInfo);
	
	exifInfo.ImageDescription[0] = '\0'; // 禁止嵌入该信息项
	strcpy((char*)exifInfo.Make, "Anyka"); // 注意字符串不要超长
	strcpy((char*)exifInfo.Model, "AK3224M");
	// 不修改原有的 Artist 和 Software 信息项
	strcpy((char*)exifInfo.DateTime, MyDateTime);
	exifInfo.bThumbnail = AK_TRUE;　　// 允许嵌入缩略图
	Img_SetExifInfo(&exifInfo);

3.3.11 Img_SetJPEGTaskFunc

原　　型	IMG_T_VOID Img_SetJPEGTaskFunc(const JPEGTaskFunc func)
功能概述	设置 JPEG 并行工作任务函数
参数说明	JPEGTaskFunc | 与 JPEG 模块并行工作的函数指针
返回值说明	无
注意事项	JPEGTaskFunc 将会在每次执行 JPEG 编解码操作时被循环调用。应用程序在调用完 JPEG 接口后应该将该函数指针重新设置成 NULL，以避免影响其他应用。

调用示例	extern void foo();
	Img_SetJPEGTaskFunc(foo);
	Img_YUV2JPEG(srcY, srcU, srcV, dstBuffer, &size, 640, 480, 75);//foo 函数将在执行本函数时被调用。
	Img_SetJPEGTaskFunc(NULL);

3.4 WBMP解码

3.4.1 Img_WBMPInfo

原　　型	T_BOOL Img_WBMPInfo(const T_U8 *buffer, T_U16 *width, T_U16 *height);	
功能概述	取得 WBMP 格式图片的宽度和高度信息	
参数说明	buffer	WBMP 格式图片的缓冲区指针
	width	输出,WBMP 格式图片宽度
	height	输出,WBMP 格式图片高度
返回值说明	AK_TRUE: 取 WBMP 图片信息成功, width 和 height 中返回图片宽度和高度值有效 AK_FALSE: 取 WBMP 图片信息失败, buffer 指向缓冲区不是 WBMP 格式数据	
注意事项	-	
调用示例	-	

3.4.2 Img_WBMP2BMP

原　　型	T_U8 *Img_WBMP2BMP(const T_U8 *buffer, T_U32 length, T_U32 *outLen)	
功能概述	将 WBMP 格式图片解码成 BMP 格式图片	
参数说明	buffer	WBMP 格式的源数据缓冲区指针
	length	WBMP 格式的源数据字节长度
	outLen	输出, 解码成功返回结果 BMP 数据字节长度
返回值说明	AK_NULL：解码失败, outLen 中的返回值无效 其它：解码成功，返回结果 BMP 数据缓冲区指针, 长度在 outLen 中	
注意事项		
调用示例		

3.5 PNG解码

3.5.1 Img_PNGInfo

原　　型	T_BOOL Img_PNGInfo(const T_U8 *pngbuf, T_U16 *width, T_U16 *height, T_U8 *bitCount);	
功能概述	取得 PNG 格式图片的宽度、高度和颜色深度信息	
参数说明	pngbuf	PNG 格式的源数据缓冲区指针
	width	输出，成功则返回 PNG 图片宽度
	height	输出， 成功则返回 PNG 图片高度
	bitCount	输出，成功则返回 PNG 图片颜色深度
返回值说明	AK_TRUE：取 PNG 图片信息成功 AK_FALSE：取 PNG 图片信息失败，pngbuf 指向缓冲区不是 PNG 格式数据	
注意事项	-	
调用示例	-	

3.5.2 Img_PNG2BMP

原　　型	T_U8 *Img_PNG2BMP(const T_U8 *pngbuf, T_U32 *outLen);	
功能概述	将 PNG 格式图片解码成 BMP 格式图片	
参数说明	pngbuf	PNG 格式的源数据缓冲区指针
	outLen	输出, 解码成功返回结果 BMP 数据字节长度
返回值说明	AK_NULL：解码失败, outLen 中的返回值无效 其它：解码成功, 返回结果 BMP 数据缓冲区指针, 长度在 outLen 中	
注意事项	返回的 BMP 数据指针在不用时必须将其释放，否则会有内存泄漏	
调用示例		

3.5.3 Img_PNG2BMPEx

原　　型	T_U8 *Img_PNG2BMPEx(const T_U8 *pngbuf, T_U32 *outLen, T_U16 dstWidth, T_U16 dstHeight);
功能概述	将 PNG 格式图片按指定的尺寸进行缩小，生成成 BMP 格式图片

原　　型	T_U8 *Img_PNG2BMPEx(const T_U8 *pngbuf, T_U32 *outLen, T_U16 dstWidth, T_U16 dstHeight);	
参数说明	pngbuf	PNG 格式的源数据缓冲区指针
	outLen	输出, 解码成功返回结果 BMP 数据字节长度
	dstWidth	指定解码后的 BMP 图片宽度
	dstHeight	指定解码后的 BMP 图片高度
返回值说明	AK_NULL：解码失败, outLen 中的返回值无效 其它：解码成功, 返回结果 BMP 数据缓冲区指针, 长度在 outLen 中	
注意事项	返回的 BMP 数据指针在不用时必须将其释放，否则会有内存泄漏	
调用示例	-	

3.6　PNG编码

Img_BMP2PNG

原　　型	T_BOOL Img_BMP2PNG(IMG_T_U8 *png_buff, IMG_T_U32 *png_len, const IMG_T_U8 *bmp_buff);	
功能概述	将 BMP 格式图片解码成 PNG 格式图片	
参数说明	png_buff	PNG 格式的数据缓冲区指针
	png_len	设置和返回的 PNG 缓冲大小指针
	bmp_buff	BMP 格式的源数据缓冲指针
返回值说明	解码成功返回 AK_TRUE，失败返回 AK_FALSE	
注意事项	调用前需要设置 png_len 的最大值，通常为应用程序已经分配的 png_buff 缓冲大小	
调用示例	Unsigned long png_len = 0x100000 Unsigned char *png_buff = malloc(png_len); Img_BMP2PNG(png_buff, &png_len, bmp_buff);	

3.7 GIF解码

3.7.1 GIFEnum_SetDecMode

原　　型	T_VOID GIFEnum_SetDecMode(GIF_DEC_MODE mode);	
功能概述	设置 GIF 解码模式，有两种：最大速度、最小空间	
参数说明	mode	设置的 GIF 解码模式
返回值说明	无	
注意事项	如果没调用该函数，则取默认值：最大速度	
调用示例	GIFEnum_SetDecMode(MAX_SPEED);	

3.7.2 Img_GIFInfo

原　　型	T_BOOL Img_GIFInfo(const T_U8 *GIFbuf, T_U16 *width, T_U16 *height, T_U8 *bitCount);	
功能概述	获取 GIF 信息	
参数说明	GIFbuf	GIF 图像数据
	width	返回的 GIF 图像宽度
	height	返回的 GIF 图像高度
	bitCount	GIF 图像颜色位数
返回值说明	如果数据是 GIF 格式，返回 AK_TRUE，否则返回 AK_FALSE	
注意事项	-	
调用示例	-	

3.7.3 GIFEnum_New

原　　型	T_hGIFENUM GIFEnum_New(const T_U8 *GIFbuf, T_S32 buflength);	
功能概述	创建 GIF 图片解码 HANDLE	
参数说明	GIFbuf	GIF 格式图片的源缓冲区指针
	buflength	GIF 格式图片的源缓冲区长度
返回值说明	IMG_INVALID_HANDLE: 创建 HANDLE 失败, 返回无效 HANDLE, 解码失败 其它: 有效的 HANDLE	

原　　型	T_hGIFENUM GIFEnum_New(const T_U8 *GIFbuf, T_S32 buflength);
注意事项	创建 HANDLE 成功, GIFbuf 所指向内存会一直被 HANDLE 引用直到 HANDLE 被 Close,因此在 HANDLE 被 Close 之前 GIFbuf 所指向的内存不能被释放 首次创建 GIF 图片解码 HANDLE 时, 会解码 GIF 中的第一帧图片
调用示例	-

3.7.4 GIFEnum_Close

原　　型	T_VOID GIFEnum_Close(T_hGIFENUM gifEnum);	
功能概述	关闭 GIF 解码 HANDLE	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
返回值说明	-	
注意事项	-	
调用示例	-	

3.7.5 GIFEnum_GetFrameCount

原　　型	T_U16 GIFEnum_GetFrameCount(T_hGIFENUM gifEnum);	
功能概述	取得 HANDLE 对应 GIF 图片的总帧数	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
返回值说明	GIF 图片的总帧数	
注意事项	必须确保传入的 gifEnum 是有效的 HANDLE	
调用示例	-	

3.7.6 GIFEnum_FirstFrame

原　　型	T_BOOL GIFEnum_FirstFrame(T_hGIFENUM gifEnum);	
功能概述	解码 GIF 第一帧图片	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE

原　　型	T_BOOL GIFEnum_FirstFrame(T_hGIFENUM gifEnum);	
返回值说明	AK_TRUE: 解码第一帧图片成功 AK_FALSE: 解码第一帧图片失败	
注意事项	必须确保传入的 gifEnum 是有效的 HANDLE	
调用示例	-	

3.7.7 GIFEnum_NextFrame

原　　型	T_BOOL GIFEnum_NextFrame(T_hGIFENUM gifEnum);	
功能概述	解码 GIF 下一帧图片	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
返回值说明	AK_TRUE: 解码下一帧图片成功 AK_FALSE: 解码下一帧图片失败	
注意事项	必须确保传入的 gifEnum 是有效的 HANDLE	
调用示例	-	

3.7.8 GIFEnum_GetCurBMP

原　　型	const T_U8 *GIFEnum_GetCurBMP(T_hGIFENUM gifEnum, T_U32 *dataLen, T_U8 *bitsPerPix);	
功能概述	取得 GIF 解码 HANDLE 当前帧的 BMP	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
	dataLen	输出, 返回当前帧的 BMP 数据长度
	bitsPerPix	输出, 返回当前帧的 BMP 色深值
返回值说明	AK_NULL: 解码 HANDLE 没有当前帧 BMP 数据, dataLen, bitsPerPix 中的值无效 其它: 当前帧 BMP 数据指针	
注意事项	必须确保传入的 gifEnum 是有效的 HANDLE	
调用示例	-	

3.7.9 GIFEnum_GetFrameBMP

原　　型	const T_U8 *GIFEnum_GetFrameBMP(T_hGIFENUM gifEnum, T_U16 packetIdx, T_U32 *outLen, T_U8 *bitsPerPix);	
功能概述	解码 HANDLE 对应 GIF 数据指定帧的 BMP 并返回	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
	packetIdx	指定的帧号
	outLen	输出, 返回解码输出 BMP 的数据长度
	bitsPerPix	输出, 返回返回解码输出 BMP 色深值
返回值说明	AK_NULL: 解码失败, outLen, bitsPerPix 中的值无效 其它: 当前帧 BMP 数据指针	
注意事项	-	
调用示例	-	

3.7.10 GIFEnum_GetCurPacket

原　　型	T_U16 GIFEnum_GetCurPacket(T_hGIFENUM gifEnum);	
功能概述	取得 GIF 解码 HANDLE 当前帧的帧序号	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
返回值说明	返回当前帧的帧序号	
注意事项	-	
调用示例	-	

3.7.11 GIFEnum_GetCurDelay

原　　型	T_S32 GIFEnum_GetCurDelay(T_hGIFENUM gifEnum);	
功能概述	取得 GIF 解码 HANDLE 当前帧的延时时间	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
返回值说明	当前帧在播放 GIF 动画时所要求的延时时间, 单位为 ms	
注意事项	只有当 GIFEnum_GetCurBMP 返回不为 AK_NULL 时, 本函数的返回值才有意义	
调用示例	-	

3.7.12 GIFEnum_SetCurDelay

原　　型	T_VOID GIFEnum_SetCurDelay(T_hGIFENUM gifEnum, T_S32 curDelay);	
功能概述	更改当前帧的延时时间	
参数说明	gifEnum	前面用 GIFEnum_New 创建的 HANDLE
	curDelay	延时时间, 单位为 ms
返回值说明	-	
注意事项	DMP 平台使用该接口来更改 GIF 播放时当前显示帧还应显示的延时时间, 在调用了 GIFEnum_SetCurDelay 之后, GIFEnum_GetCurDelay 得到的值就不再是 GIF 图片中的延时时间原始值. 因此不推荐使用该接口来记录和更改 GIF 播放时当前显示帧还应已显示的延时时间, 显示 GIF 动画的上层程序应自己维护一份当前帧还应显示的延时时羊	
调用示例	-	

3.7.13 GIFEnum_GetTotalSize

原　　型	T_U32 GIFEnum_GetTotalSize(T_hGIFENUM gifEnum);	
功能概述	计算 GIF 所有帧的 BMP 总大小	
参数说明	gifEnum	用 GIFEnum_New 创建的 HANDLE
返回值说明	返回 GIF 所有帧的 BMP 总大小	
注意事项	-	
调用示例	-	

3.8 其它

3.8.1 SetImageChipID

原　　型	IMG_T_VOID SetImageChipID(JPEG_HW_VERSION ver)	
功能概述	设置当前芯片的使用的 JPEG 硬件解码版本	
参数说明	ver	JPEG 硬件解码版本，取值参见 3.1.8 节
	-	-
返回值说明	无	

原　　型	IMG_T_VOID SetImageChipID(JPEG_HW_VERSION ver)
注意事项	使用 JPEG_HW_VER1 的芯片型号：AK3221, AK3223, AK3224, AK3225, AK3610, AK3620；使用 JPEG_HW_VER2 的芯片型号：AK322L, AK3225L, AK3631L
调用示例	SetImageChipID(JPEG_HW_VER1);

3.8.2 Img_SetCallbackFuns

原　　型	void Img_SetCallbackFuns(const CB_FUNS *pCBFuns);	
功能概述	设定回调函数指针	
参数说明	pCBFuns	函数指针结构体的指针
	-	-
返回值说明	无	
注意事项	-	
调用示例	-	

3.8.3 Img_SetFlushCacheFunc

原　　型	void Img_SetFlushCacheFunc(const FlushCacheFunc func)	
功能概述	强制将 Cache 数据送到内存	
参数说明	FlushCacheFunc	FlushCache 函数指针
	-	-
返回值说明	无	
注意事项	-	
调用示例	-	

3.8.4 Img_GetVersionInfo

原　　型	const T_U8 *Img_GetVersionInfo(T_VOID)	
功能概述	获取图像库版本信息	
参数说明	无	

原　　型	const T_U8 *Img_GetVersionInfo(T_VOID)
返回值说明	图像库版本信息字符串
注意事项	-
调用示例	-

3.8.5 Img_GetLastError

原　　型	T_IMAGE_ERROR Img_GetLastError(T_VOID);	
功能概述	获得最后一次出错的出错代码	
参数说明	无	
返回值说明	出错代码的枚举值，参见 T_IMAGE_ERROR 定义	
注意事项	-	
调用示例		

-

3.8.6 Img_ImageType

原　　型	T_IMAGE_TYPE Img_ImageType(const T_U8 *imgBuf);	
功能概述	取得图片数据的图片格式类型	
参数说明	imgBuf	图片数据的缓冲区指针
返回值说明	图片类型枚举值, 参见 T_IMAGE_TYPE 定义	
注意事项	-	
调用示例	-	

3.8.7 Img_ImageToBmp

原　　型	T_U8 *Img_ImageToBmp(const T_U8 *imgBuf, T_U32 bufLen, T_U32 *outLen);	
功能概述	将图片解码成 BMP 格式	
参数说明	imgBuf	图片数据的缓冲区指针
	bufLen	图片数据的缓冲区长度
	outLen	输出, 结果 BMP 格式数据的字节长度

原　　型	T_U8 *Img_ImageToBmp(const T_U8 *imgBuf, T_U32 bufLen, T_U32 *outLen);
返回值说明	解码成功, 返回 BMP 格式图像数据 解码失败, 返回 AK_NULL, outLen 值无效
注意事项	本函数解码功能将自动判断图片类型, 进行支持的图片类型对应的解码 如果源图片是 GIF 动画格式, 将解码输出第一帧图片 如果解码成功, 得到 BMP 格式图像数据指针, 不再使用 BMP 图像数据指针时应用 Img_Free 将其释放掉, 否则会有内存泄漏
调用示例	-

3.8.8 Img_YUV2RGB

原　　型	T_S32 Img_YUV2RGB (T_U8 *srcY, T_U8 *srcU, T_U8 *srcV, T_U8 *RGB, T_S32 srcWidth, T_S32 srcHeight , T_S32 dstWidth, T_S32 dstHeight, T_S32 timeout)	
功能概述	将 YUV 数据转换为 RGB 数据	
参数说明	srcY	Y 数据指针
	srcU	U 数据指针
	srcV	V 数据指针
	RGB	RGB 数据指针
	srcWidth	源图像的宽度
	srcHeight	源图像的高度
	dstWidth	目的图像的宽度
	dstHeight	目的图像的高度
	timeout	设定的超时值（3210 芯片有效）
返回值说明	0：转换正确；负值：转换失败	
注意事项	输入 YUV 格式：H211（AK3810、AK78xx 芯片则为 YUV 420）； 源图像尺寸必须大于目的图像尺寸	
调用示例	-	

4 API调用注意事项

4.1 图像库初始化

在调用图像库的解码和编码功能前，系统需要先调用以下接口对图像库进行初始化设置：

- SetImageChipID()：设置当前芯片的使用的 JPEG 硬件解码版本
- Img_SetCallbackFuns()：设置回调函数指针
- Img_SetFlushCacheFunc()：设置平台 Cache 刷新回调函数指针

4.2 依赖于其他模块

由于图像的解码和编码需要动态分配内存，因此依赖于内存管理模块。用户在使用图像库前需调用 Img_SetCallbackFuns 来指定内存分配和释放函数，函数的参数和返回值请参考 3.1.5 一节。

此外，图像库还需要将 Cache 里面的数据强制送到内存，用户在调用具体编解码函数前需要使用 Img_SetFlushCacheFunc 来指定 FlushCache 函数指针，函数参数类型和返回值请参考 3.1.5 一节。

4.3 图像库的内存需求

API 接口编解码函数中，生成转换成 BMP 的函数都是在库里面分配并返回，因此用户不需要预先分配 bmp 内存，其他的函数都必须在应用程序中预先分配。

图像库在编解码过程中，除了用于显示的 BMP 内存外，还需要分配的内存最大不超过 60KB。

4.4 32 位PNG图片解码的使用方法

32 位 PNG 解码后是 32 位的 BMP，代表每个象素的前三个字节是 RGB 数据，第四个字节是该象素的透明度 alpha。应用可以根据 alpha 值，使用下面公式，将该 BMP 与自定义的背景色相混合，形成应用需要的目标 BMP：

$$X_{dst} = X + (0x100\text{-}alpha)*X_{bk}$$

其中，X代表某个RGB，X_{bk}为背景色，X_{dst}为最终颜色。

背景色可以是每个象素都不同，也可以相同。

4.5 典型调用范例

```
T_VOID main(int argc, char* argv[])
{
  CB_FUNS cb_funs;
  J_OSD_Info osdinfo;
  T_U8 *srcY = NewAddresssrcY;
  T_U8 *srcU = NewAddresssrcU;
  T_U8 *srcV = NewAddresssrcV;
  T_U8 *dstJPEG = NewAddressdstJPEG;

  T_U32 srcWidth = 640;
  T_U32 srcHeight = 480;
  T_U32 dstWidth = 1280;
  T_U32 dstHeight = 960;
  T_U32 size = NewSize;
  T_U16 quality = 50;

  SetImageChipID(AK37xx_JPEG_VER);
  memset(&cb_funs, 0, sizeof(cb_funs));
cb_funs.malloc = NewMalloc;
  cb_funs.free   = NewFree;
  cb_funs.printf = DebugOutput;
          Img_SetCallbackFuns(&cb_funs);
  Img_SetFlushCacheFunc(Cache_FlushFunc);

  memset(&osdinfo, 0, sizeof(osdinfo));
  osdinfo.osdWidth = 128;
  osdinfo.osdHeight = 32;
```

```
    osdinfo.osd_H_offset = 16;
    osdinfo.osd_V_offset = 16;
    osdinfo.alpha = 16;
    osdinfo.osdRGB565 = NewAddressRGB;

    Img_YUV2JPEG_OSD(srcY, srcU,srcV,dstJPEG, &size,srcWidth,  srcHeight,
                                        dstWidth,dstHeight, quality,&osdinfo);

    // other operations
            // ...
}
```