Hantro

Application Programming Interface 1.0

7280 JPEG Encoder

User Manual Version 1.1

Copyright Information

Hantro Products Oy has used their best efforts in preparing this document. Hantro Products Oy makes no warranty, representation, or guarantee of any kind, expressed or implied, with regards to this document. Hantro Products Oy does not assume any and all liability arising out of this document including without limitation consequential or incidental damages.

Hantro Products Oy reserves the right to make changes and improvements to any of the products described in this document without prior notice.

Hantro Products Oy
Kiviharjunlenkki 1
90220 Oulu
FINLAND
www.hantro.com

Glossary

API	Application Programming Interface
CIF	Common Interchange Format (352x288 pixels)
fps	Frames per second
H.263	Video Coding Standard for low bit rate communication (ITU-T)
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group, also a common term for still images according to [1]
MCU	Minimum Coded Unit (16x16 pixels in this case)
MPEG-4	Motion Picture Experts Group standard 4 (ISO / IEC 14496-2)
QCIF	Quarter CIF (176x144 pixels)
QVGA	Quarter Video Graphics Array (320x240 pixels)
QQVGA	Quarter QVGA (160x120 pixels)
sub-QCIF	Another video resolution (128x96 pixels)
VGA	Video Graphics Array (resolution 640x480 pixels)
YCbCr	Luminance (Y) and chrominance (Cb and Cr) color space
YUV	Common term for YCbCr
THUMBNAIL	A small, low resolution version of a larger image

Table of Contents

1 Introduction

This document presents the Application Programming Interface (API) of the Hantro 7280 JPEG hardware based encoder. The encoder is able to encode JPEG standard [1] baseline profile compatible streams.

The encoder conforms to the JPEG Baseline profile and can encode streams up to a maximum picture size of 4672x3504 (~16 Megapixels).

The usage of the API is described in chapter 2. Chapter 3 gives guidelines to frame storage format. The detailed function interfaces and data types are introduced in chapter 4. Chapter 5 gives application examples. References are presented in the end of the document.

In the document all functions, parameters, data types and code are described in `Courier new (syntax style)` font. Notes and filenames are written in *italic* expression.

This document assumes that the reader understands the fundamentals of C-language and possesses knowledge about JPEG concepts and standards.

2 API Version History

Table 1 describes the released API versions, any changes introduced.

TABLE 1 API VERSION HISTORY

API version	Changes/Comments
1.0	Original version

3 Encoder Usage

The block diagram of a typical encoding process is depicted in FIGURE 1. The figure shows the main steps of the encoder usage: the initialization, configuration, the stream producing and finally the release of the encoder.

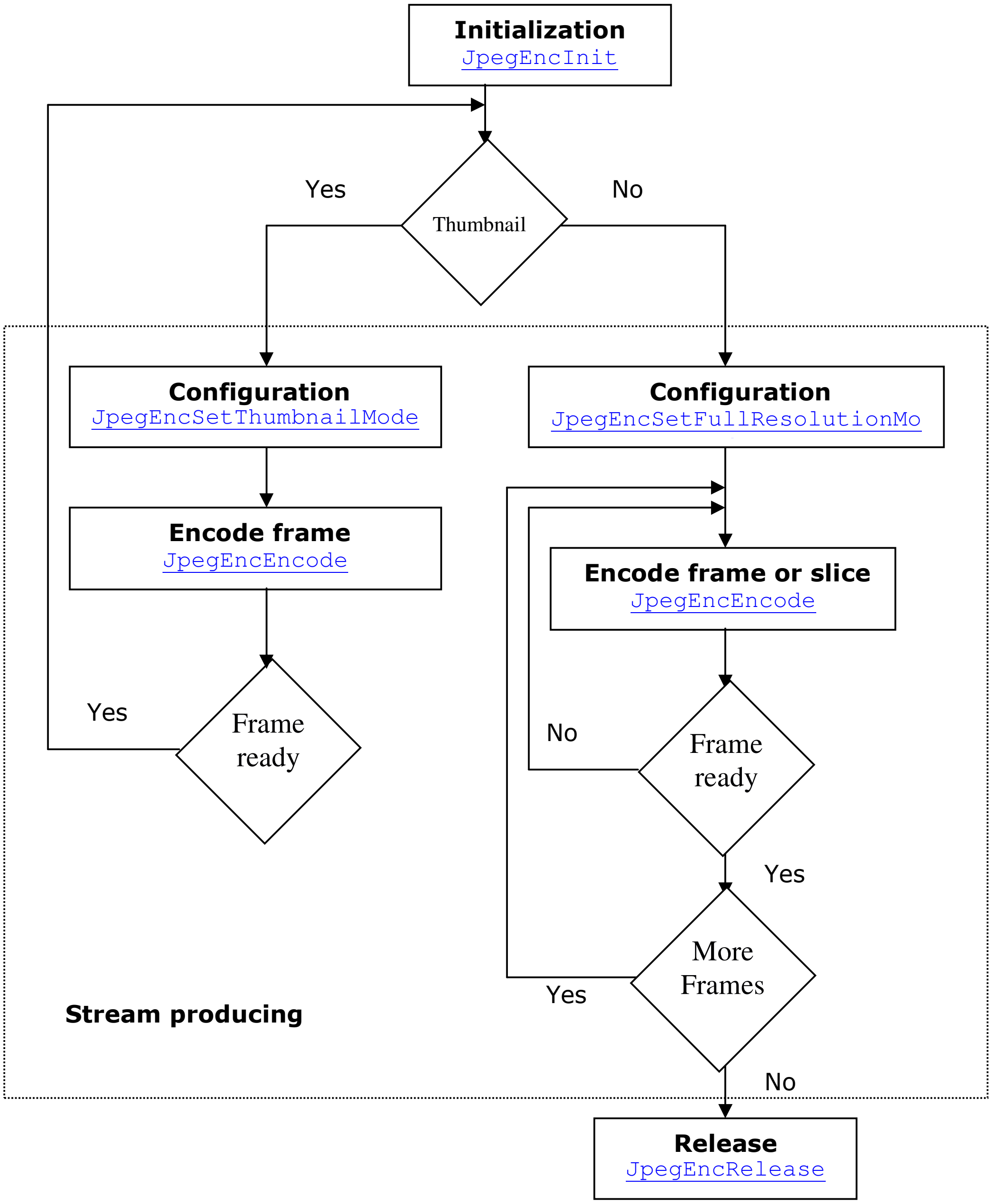

FIGURE 1 ENCODING PROCESS BLOCK DIAGRAM

3.1 Initialization and release of the encoder

In order to be able to use the encoder, it has to be properly initialized first. The initialization is done by calling JpegEncInit. The call will allocate all the resources needed by the encoder. If successful, the initialization call will return a new instance of the encoder, which will be used as an identifier for all the subsequent encoder operations.

The initialization call will return JPEGENC_OK for a successful initialization. The error code returned in case of a failure will give hints about what went wrong during the initialization process.

Encoder release

At the end of the encoding process the encoder has to be released. Use JpegEncRelease to perform a safe release of all the resources allocated when the encoder was initialized. If a release fails most probably the encoder instance was corrupted and there is no safe way to assure that all the encoder’s resources were freed.

3.2 Configuration of the encoder

The encoder needs to be configured separately for thumbnail and full resolution encoding mode. There are several encoding parameters that need to be updated after the encoder has been initialized. Function JpegEncSetThumbnailMode will configure encoder for thumbnail encoding. After the thumbnail encoding is ready, function JpegEncSetFullResolutionMode is called for configure encoder for full resolution frame encoding.

Note! The thumbnail image has to be encoded with JpegEncEncode before the full resolution mode is configured!

The encoder uses a certain number of parameters for configuration. These parameters are the most significant ones, parameters that can’t be changed during the encoding process of mode.

Encoded picture size

The width and height in pixels of the encoded picture is specified during configuration an encoder mode. Notice the difference between the input picture size as captured by a camera and the final encoded image size. The input image size can be larger than the encoded one if before the encoding process the input image is cropped (see details in Picture cropping).

Note the following implementation specific limitations in force: the encoded picture width has to be a multiple of 4 and the height a multiple of 2; the smallest encoded picture size is 64x16; in full resolution mode the maximum input picture size is 8192x8192 and the maximum encoded picture size is 4672x3504 (or 3504x4672 when rotated); in thumbnail mode the maximum input picture size is 256x256 and the maximum encoded picture size is 240x240.

Even though the encoded picture width can be a 4 multiple, the horizontal scanline (stride) has to be a 16 multiple, meaning that the memory offset form the start of a pixel row to the beginning of the next pixel row is always a 16 multiple. This assumption can be seen in the initial value of the pre-processor parameters, where the input source image width is the rounded up 16 multiple of the encoded width. There is no such limitation on the height of the picture. Figure 2 shows the described limitation regarding the input picture horizontal scanline. Check also chapter 4.1 Frame size limitations.
The encoded size cannot be altered after the initialization.

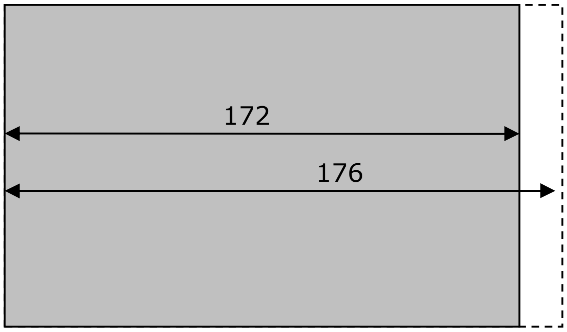

FIGURE 2 ENCODED PICTURE SIZE AND ASSUMED VIRTUAL INPUT PICTURE SIZE

When the encoded picture size is obtained by rotating the original input picture, the size set at the initialization phase reflects the final rotated pictures dimensions.

Coding type indication

The frame coding type can be chosen for full resolution mode encoding. A single frame can be encoded as a whole or in several slices. Each slice is of equal size with the width of the original image and height multiple of 16. The height of a slice is defined in MCU rows with restart interval. Slice coding saves memory and allows simultaneous operation of a camera module and the encoder. The coding types are illustrated in Figure 2.

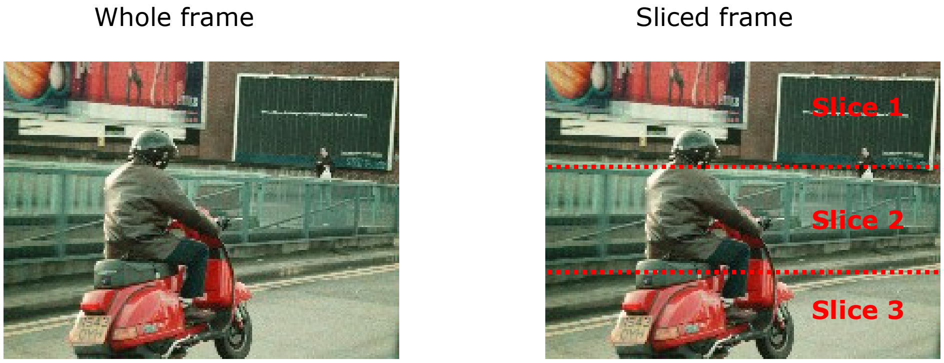

FIGURE 3 DIFFERENCES BETWEEN CODING TYPES

Tasks when encoding whole frame at a time:

1) Get frame
2) Encode frame

Tasks when encoding a frame a slice at a time:

1) Get first slice
2) Encode first slice
3) Get second slice
4) Encode second slice
5) Get third slice
6) Encode third slice

In whole frame encoding the frame must be retrieved and stored in memory before the encoding can begin. In sliced frame encoding each slice is retrieved and encoded separately. It is possible to process the tasks 2) and 3) at the same time, as well as tasks 4) and 5). On the other hand it is possible to use a single slice buffer for all of the slices if the tasks are done sequentially.

3.3 Pre-processor usage

The encoder includes three pre-processing blocks.

1. YCbYCr 4:2:2 to YCbCr 4:2:0 conversion
2. Picture rotation
3. Picture cropping

By default the encoder input is considered to be in planar YUV 4:2:0 format, cropping and rotations are disabled.

3.3.1 Color conversion

The conversion block reads the input picture in interleaved YCbCr 4:2:2 format. The picture is converted into YCbCr 4:2:0 format by sub-sampling the chrominance samples vertically. The conversion is specified at the initialization phase and it will affect every encoded picture.

3.3.2 Picture rotation

The rotation block rotates the picture 90 degrees clockwise or counter-clockwise. The rotation is configured at the initialization phase and it will affect every encoded picture. Figure 4 illustrates the picture dimensions when rotating the input.

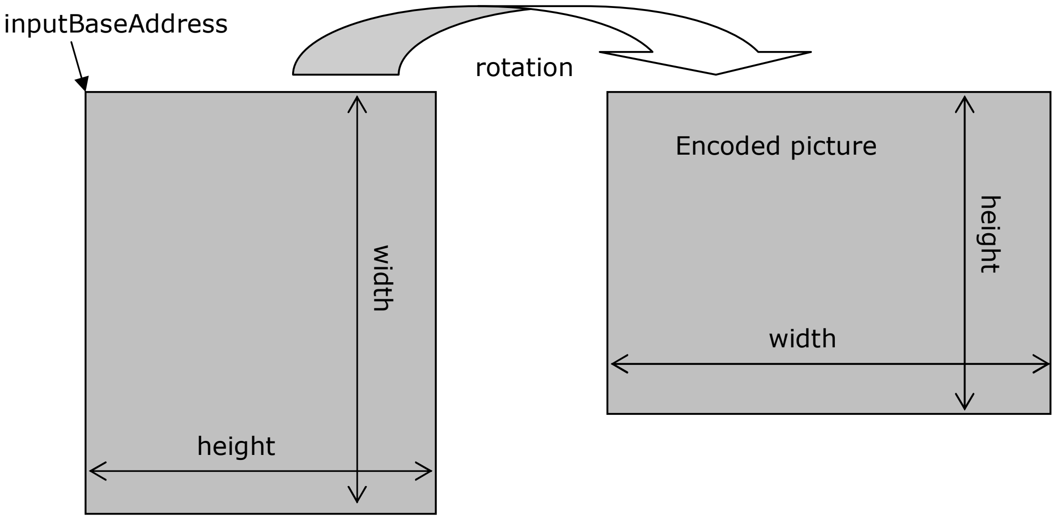

FIGURE 4 PICTURE DIMESIONS WITH ROTATION

NOTE! Picture rotation is only supported in whole frame encoding mode.

3.3.3 Picture cropping

The frame cropping provides a way to read and encode a smaller selected part of a bigger picture. The usage of cropping block is set in configuration phase of the encoder (see `JpegEncSetThumbnailMode` and `JpegEncSetFullResolutionMode`).

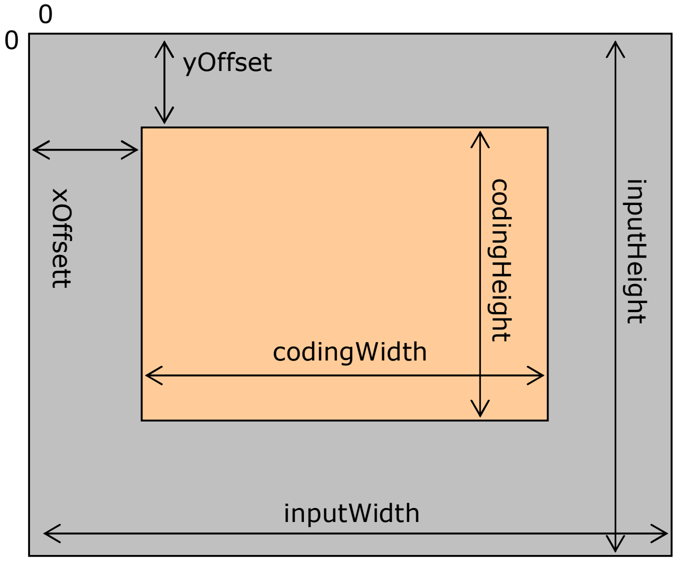

FIGURE 5 RELATIONS BETWEEN THE INPUT IMAGE'S AND THE ENCODED IMAGE'S SIZES

Figure 5 represents the different picture parameters (grey picture is the input picture as captured by a camera and the magenta picture is the one encoded by the encoder). Below are the conditions that have to be satisfied in order to enable the cropping block:

$$inputWidth >= (codingWidth + 15)\ \&\ (\sim 15)$$
$$inputWidth - xOffset - codingWidth \geq 0$$
$$inputHeight - yOffset - codingHeight \geq 0$$

The input source picture width has to take always 16 multiple values. The source height has to be 2 multiple.

3.4 Encoding frames

After a successful initialization and an optional configuration, the encoder is ready to produce a JPEG/JFIF compliant stream.

The supported formats of the input are: planar YCbCr 4:2:0, semiplanar YCbCr 4:2:0 or interleaved YCbYCr 4:2:2. The format in use has been selected during the initialization. Independently of what input format is used the memory area where the buffers are located has to be 64-bit aligned, linear and hardware accessible. With other words the addresses, which are passed as parameters to the JpegEncEncode function, are bus addresses and will be given to the hardware unmodified and they will be used to read the input frame.

The output buffer has to be specified both for software and hardware. A virtual address pointer will specify the buffer for software access and a bus address for the hardware access. The size of the output buffer has to be correlated with the image size and quantization level. If the encoder reaches the end of the output buffer the encoding of the current frame cannot continue and it will be lost (i.e. JpegEncEncode returns

JPEGENC_OUTPUT_BUFFER_OVERFLOW). If the sliced coding type is used the JpegEncEncode function will encode one slice at a time and return JPEGENC_RESTART_INTERVAL.

Encoding a JPEG image that contains a thumbnail has to be done in the following sequence:

1. call JpegEncSetThumbnailMode to configure the thumbnail.
2. allocate big enough output buffer to fit the final full resolution encoded image together with the thumbnail.
3. call JpegEncEncode to encode the thumbnail image. Do not alter the output buffer parameters because the full resolution jpeg data will be appended after the thumbnail data.
4. call JpegEncSetFullResolutionMode to configure the full resolution image.
5. call JpegEncEncode to encode the full resolution image. When JPEGENC_FRAME_READY is returned the full JPEG image is ready.

3.5 Quantization level vs. output size

An application can define several quantization levels (see JpegEncInit) to be used in the encoding. A small value generates lower image quality and smaller JFIF size and a large value will generate higher image quality and larger JFIF size. Table 2 presents an example of file sizes for different quantization levels and image resolutions ("motorcycle" image seen in Figure 3).

TABLE 2 EXAMPLE OF THE QUANTIZATION LEVEL'S INFLUENCE ON JFIF STREAM'S SIZE

Quantization Level	CIF 352x288 [kb]	VGA 640x480 [kb]	1Mpixel 1280x864 [kb]	2Mpixel 1600x1232 [kb]	5Mpixel 2560x1968 [kb]
0	4	9	26	42	97
1	6	13	34	52	115
2	7	16	42	62	134
3	9	19	47	70	150
4	10	21	54	78	166
5	12	25	62	88	186
6	13	29	70	105	229
7	17	38	92	145	326
8	24	53	158	258	595
9	42	94	561	823	1665

4 Frame Storage Format

The input picture format of the encoder is planar YCbCr 4:2:0, semiplanar YCbCr 4:2:0 or interleaved YCbCr 4:2:2. Figure 6 describes how the planar YCbCr 4:2:0 picture is stored in external memory. The luminance and both chrominance components are stored in three buffers and they must be located in a linear and physically contiguous memory block.

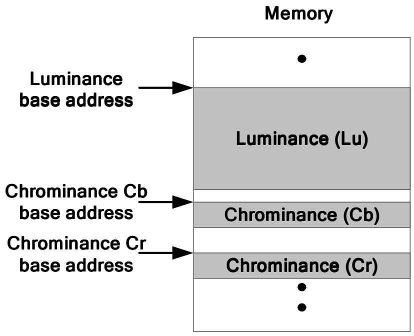

FIGURE 6 PLANAR YCBCR 4:2:0 PICTURE STORAGE IN EXTERNAL MEMORY

Figure 7 describes the format of semiplanar YCbCr 4:2:0 picture. The luminance is equal to planar format but the Cb and Cr chrominance components are interleaved together.

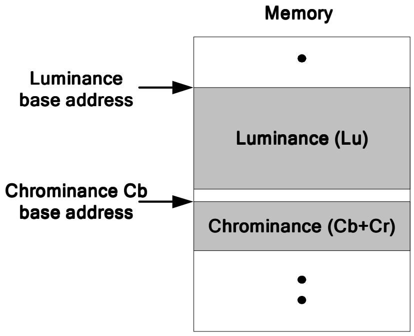

FIGURE 7 SEMIPLANAR YCBCR 4:2:0 PICTURE STORAGE IN EXTERNAL MEMORY

Figure 8 describes the format of interleaved YCbCr 4:2:2 picture where all the three components are interleaved. The sample order is selected at initialization to be either Y-Cb-Y-Cr or Cb-Y-Cr-Y.

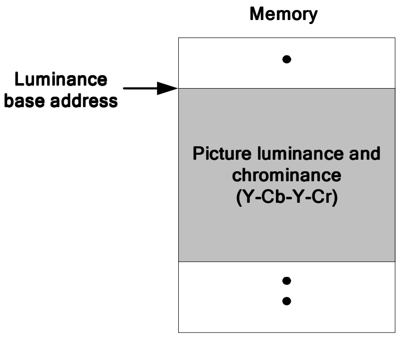

FIGURE 8 INTERLEAVED YCBCR 4:2:2 PICTURE STORAGE IN EXTERNAL MEMORY

The input picture data endianess is set when the encoder software is compiled and can't be changed during run-time. The sizes of the different components are based on the picture's dimensions and the input format:

Planar YCbCr 4:2:0	Y = width x height [bytes]
	Cb = width/2 x height/2 [bytes]
	Cr = width/2 x height/2 [bytes]

Semiplanar YCbCr 4:2:0	Y = width x height [bytes] Cb+Cr = width x height/2 [bytes]
Interleaved YCbCr 4:2:2	Y+Cr+Cr = width x height x 2 [bytes]

4.1 Frame size limitations

YCbCr 4:2:0 limitations regarding the input picture size and the encoded picture size:

encoded_width % 4 = 0
encoded_height % 2 = 0
input_width >= (encoded_width + 15) & (~15)
input_width % 16 = 0
input_height >= encoded_height
input_height % 2 = 0

YCbCr 4:2:2 limitations regarding the input picture size and the encoded picture size:

encoded_width % 4 = 0
encoded_height % 2 = 0
input_width >= (encoded_width + 15) & (~15)
input_width % 8 = 0
input_height >= encoded_height
input_height % 2 = 0

Where:

%	- remainder operator
&	- bitwise AND operator
~	- bitwise NOT operator

5 Interface Functions

This chapter describes the API declared in *jpegencapi.h*.

Common numeric data types are declared in *basetype.h*:
u8 – unsigned 8 bits integer value
i8 – signed 8 bits integer value
u16 – unsigned 16 bits integer value
i16 – signed 16 bits integer value
u32 – unsigned 32 bits value
i32 – signed 32 bits value

5.1 JpegEncGetApiVersion

Syntax

```
JpegEncApiVersion JpegEncGetApiVersion(void)
```

Purpose

Returns the encoder’s API version information. Does not require encoder initialization.

Parameters

None.

Return value

`JpegEncApiVersion`

A structure containing the API’s major and minor version number.

Description

```
typedef struct
{
    u32 major;
    u32 minor;
} JpegEncApiVersion;
```

`major`

The major version number.

`minor`

The minor version number.

5.2 JpegEncGetBuild

Syntax

JpegEncBuild `JpegEncGetBuild(`**void**`)`

Purpose

Returns the encoder's hardware and software build information. Does not require the encoder initialization.

Parameters

None.

Return value

JpegEncBuild

A structure containing the encoder's builds information.

```
typedef struct
{
    u32 swBuild;
    u32 hwBuild;
} JpegEncBuild;
```

`swBuild`

The internal release version of the 7280 JPEG software.

`hwBuild`

The internal release version of the 7280 hardware.

5.3 JpegEncInit

Syntax

JpegEncRet `JpegEncInit(` **JpegEncCfg** `*pEncCfg,` **JpegEncInst** `*instAddr )`

Purpose

Call this function to init the encoder and get an instance of it.

Parameters

JpegEncCfg `*pEncCfg`

Points to a structure that contains the encoder's initial configuration parameters.

JpegEncInst `*instAddr`

Points to a space where the new encoder instance will be stored.

Return value

JPEGENC_OK – initialization successful
JPEGENC_NULL_ARGUMENT – error, a pointer argument had a null value
JPEGENC_INVALID_ARGUMENT – error, one of the arguments was invalid
JPEGENC_MEMORY_ERROR – error, the encoder failed to allocate memory
JPEGENC_EWL_ERROR – error, the encoder's system interface failed to initialize

Description

```
typedef struct
{
    u32 inputWidth;
    u32 inputHeight;
    u32 xOffset;
    u32 yOffset;
    u32 codingWidth;
    u32 codingHeight;
    u32 restartInterval;
    u32 qLevel;
    JpegEncFrameType frameType;
    JpegEncPictureRotation rotation;
    JpegEncCodingType codingType;
    JpegEncAppUnitsType unitsType;
    JpegEncTableMarkerType markerType;
    u32 xDensity;
    u32 yDensity;
    u32 comLength;
    u8 *pCom;
    u32 thumbnail;
    JpegEncCfgThumb cfgThumb;
} JpegEncCfg;
```

Only the following fields of the structure are used by the 7280 encoder:

qLevel

The quantization level. Defines the amount of quantization used in the encoding process. Small value = low image quality = small JFIF size. Large value = high image quality = large JFIF size.
Valid value range: [0, 9]

frameType

Specifies the input YUV picture format. The formats are described in Frame Storage Format chapter. The possible values are:
JPEGENC_YUV420_PLANAR
JPEGENC_YUV420_SEMIPLANAR
JPEGENC_YUV422_INTERLEAVED_YUYV
JPEGENC_YUV422_INTERLEAVED_UYVY

rotation

Specifies the YUV picture rotation before encoding. The possible values are:
JPEGENC_ROTATE_0 – no rotation.
JPEGENC_ROTATE_90R – rotates the picture clockwise by 90 degrees before the encoding.

JPEGENC_ROTATE_90L – rotates the picture counter-clockwise by 90 degrees before the encoding. Picture rotation is only supported in whole frame mode!

unitsType

Specifies unit type of xDensity and yDensity in APP0 header. The possible values are:
JPEGENC_NO_UNITS – No units, X and Y specify the pixel aspect ratio.
JPEGENC_DOTS_PER_INCH – X and Y are dots per inch.
JPEGENC_DOTS_PER_CM – X and Y are dots per cm.

markerType

Specifies the type of the DQT and DHT headers. The possible values are:
JPEGENC_SINGLE_MARKER – Y and CbCr tables are written behind own marker.
JPEGENC_MULTI_MARKER – Y and CbCr tables are written behind same marker.
See [1] for description of the header fields.

xDensity

Horizontal pixel density to APP0 header. Used only when JPEGENC_DOTS_PER_INCH or JPEGENC_DOTS_PER_CM is specified. For JPEGENC_NO_UNITS defaults to 1.
Value range: [1, 65535]

yDensity

Vertical pixel density to APP0 Header. Horizontal pixel density to APP0 header. Used only when JPEGENC_DOTS_PER_INCH or JPEGENC_DOTS_PER_CM is specified. For JPEGENC_NO_UNITS defaults to 1.
Value range: [1, 65535]

comLength

The length of comment header data.
Valid value range: [0, 65535]

pCom

Pointer to comment header data of size comLength.

thumbnail

Indicates if thumbnail is added to stream. When enabled, with a non-zero value, the thumbnail parameters have to be specified with JpegEncSetThumbnailMode.
Valid value range: [0, 1]

5.4 JpegEncRelease

Syntax

```
JpegEncRet JpegEncRelease( JpegEncInst inst )
```

Purpose

Releases an encoder instance. This will free all the resources allocated at the encoder initialization phase.

Parameters

JpegEncInst inst

The encoder instance to be released. This instance was created earlier with a JpegEncInit call.

Return value

JPEGENC_OK – release successful
JPEGENC_NULL_ARGUMENT – error, a pointer argument had a null value
JPEGENC_INSTANCE_ERROR – error, the encoder instance was invalid

5.5 JpegEncSetThumbnailMode

Syntax

```
JpegEncRet JpegEncSetThumbnailMode( JpegEncInst inst,
                                    JpegEncCfg *pEncCfg )
```

Purpose

Sets the encoder's coding parameters for thumbnail.

Parameters

JpegEncInst inst

The instance that defines the encoder in use.

JpegEncCfg *pEncCfg

Points to a structure that contains the encoder's initial configuration parameters.

Return value

JPEGENC_OK – the setting was successful
JPEGENC_INVALID_ARGUMENT – error, one of the arguments was invalid
JPEGENC_INSTANCE_ERROR – error, the encoder instance was invalid
JPEGENC_NULL_ARGUMENT – error, a pointer argument had a null value
JPEGENC_EWL_MEMORY_ERROR – error, the encoder's system interface failed to allocate memory

Description

```
typedef struct
{
    u32 inputWidth;
    u32 inputHeight;
    u32 xOffset;
    u32 yOffset;
    u32 codingWidth;
    u32 codingHeight;
    u32 restartInterval;
    u32 qLevel;
    JpegEncFrameType frameType;
    JpegEncPictureRotation rotation;
    JpegEncCodingType codingType;
    JpegEncAppUnitsType unitsType;
    JpegEncTableMarkerType markerType;
    u32 xDensity;
    u32 yDensity;
    u32 comLength;
    u8 *pCom;
    u32 thumbnail;
    JpegEncCfgThumb cfgThumb;
} JpegEncCfg;
```

Only the following fields of the structure are used by this function:

```
typedef struct
{
    u32 inputWidth;
    u32 inputHeight;
    u32 xOffset;
    u32 yOffset;
    u32 codingWidth;
    u32 codingHeight;
} JpegEncCfgThumb;
```

`inputWidth`

The width of the input image in pixels. This value has to be multiple of 16.
Valid value range: [64, 256]

`inputHeight`

The height of the input image in pixels. This value has to be multiple of 2.
Valid value range: [16, 256]

`xOffset`

The horizontal offset from the top-left corner of the input image to the top-left corner of the encoded image in pixels.
Valid value range: [0, 192]

`yOffset`

The vertical offset from the top-left corner of the input image to the top-left corner of the encoded image in pixels.
Valid value range: [0, 240]

`codingWidth`

The width of the encoded image in pixels. This value has to be multiple of 4.
Valid value range: [64, 240]

`codingHeight`

The height of the encoded image in pixels. This value has to be multiple of 2.
Valid value range: [16, 240]

The thumbnail frame cannot be encoded in slices, i.e. `codingType = ` **`JPEGENC_WHOLE_FRAME`** is assumed.

5.6 JpegEncSetFullResolutionMode

Syntax

```
JpegEncRet JpegEncSetFullResolutionMode ( JpegEncInst inst,
                                          JpegEncCfg *pEncCfg )
```

Purpose

Sets the encoder's coding parameters for full resolution frame.

Parameters

JpegEncInst `inst`

The instance that defines the encoder in use.

JpegEncCfg `*pEncCfg`

Points to a structure that contains the encoder's initial configuration parameters.

Return value

`JPEGENC_OK` – the setting was successful

`JPEGENC_INVALID_ARGUMENT` – error, one of the arguments was invalid

`JPEGENC_INSTANCE_ERROR` – error, the encoder instance was invalid

`JPEGENC_NULL_ARGUMENT` – error, a pointer argument had a null value

`JPEGENC_EWL_MEMORY_ERROR` – error, the encoder's system interface failed to allocate memory

Description

```
typedef struct
{
    u32 inputWidth;
    u32 inputHeight;
    u32 xOffset;
    u32 yOffset;
    u32 codingWidth;
    u32 codingHeight;
    u32 restartInterval;
    u32 qLevel;
    JpegEncFrameType frameType;
    JpegEncPictureRotation rotation;
    JpegEncCodingType codingType;
    JpegEncAppUnitsType unitsType;
    JpegEncTableMarkerType markerType;
    u32 xDensity;
    u32 yDensity;
    u32 comLength;
    u8 *pCom;
    u32 thumbnail;
    JpegEncCfgThumb cfgThumb;
} JpegEncCfg;
```

Only the following fields of the structure are used by this function:

`inputWidth`

The width of the input image in pixels. This value has to be multiple of 16.
Valid value range: [64, 8192]

`inputHeight`

The height of the input image in pixels. This value has to be multiple of 2.
Valid value range: [16, 8192]

`xOffset`

The horizontal offset from the top-left corner of the input image to the top-left corner of the encoded image in pixels.
Valid value range: [0, 8128]

`yOffset`

The vertical offset from the top-left corner of the input image to the top-left corner of the encoded image in pixels. In sliced encoding mode this has to be multiple of the slice height (see the description of restartInterval).
Valid value range: [0, 8176]

`codingWidth`

The width of the encoded image in pixels. This value has to be multiple of 4.
Valid value range: [64, 4672]

`codingHeight`

The height of the encoded image in pixels. This value has to be multiple of 2.
Valid value range: [16, 3504]

`restartInterval`

The restart interval in MCU rows. One MCU row equals 16 pixel rows in the encoded image. After restart interval a restart marker is added to the encoded frame. Also defines the height of a slice when using sliced coding type. Value zero disables restart intervals.
Valid value range: [0, 255]

`codingType`

Specifies the coding type of the input frame. The possible values are:

`JPEGENC_WHOLE_FRAME` **–** The whole input frame is in the memory.

`JPEGENC_SLICED_FRAME` **–** The input frame is divided into slices. Each slice is encoded at a time and a restart interval is added after the slice.

5.7 JpegEncEncode

Syntax

```
JpegEncRet JpegEncEncode (    JpegEncInst inst,
                              JpegEncIn *pEncIn,
                              JpegEncOut *pEncOut )
```

Purpose

Encodes a JPEG frame.

Parameters

JpegEncInst `inst`

The instance that defines the encoder in use.

JpegEncIn `*pEncIn`

Pointer to a structure where the input parameters are provided.

JpegEncOut `*pEncOut`

Pointer to a structure where the output parameters will be saved.

Return value

`JPEGENC_FRAME_READY` – a frame encoding was finished
`JPEGENC_RESTART_INTERVAL` – a restart interval was finished
`JPEGENC_NULL_ARGUMENT` **–** error, a pointer argument had a null value
`JPEGENC_INVALID_ARGUMENT` – error, one of the arguments was invalid
`JPEGENC_INSTANCE_ERROR` – error, the encoder instance was invalid
`JPEGENC_OUTPUT_BUFFER_OVERFLOW` – error, the output buffer's size was too small to fit the generated stream. The whole frame is lost. New frame encoding has to be started.
`JPEGENC_HW_TIMEOUT` **–** error, the wait for a hardware finish has timed out. The current frame is lost.
`JPEGENC_HW_BUS_ERROR` **–** error. This can be caused by invalid bus addresses, addresses that push the encoder to access an invalid memory area. New frame encoding has to be started.
`JPEGENC_SYSTEM_ERROR` **–** error, a fatal system error was caught. The encoding can't continue. The encoder instance has to be released.
`JPEGENC_HW_RESET` – error. Hardware was reset by external means. The whole frame is lost.
`JPEGENC_HW_RESERVED` – Failed to reserve HW for current encoder instance. The current frame is lost. New frame encoding has to be started.

Description

```
typedef struct
{
    u32 frameHeader;
    u32 busLum;
    u32 busCb;
    u32 busCr;
    u8 *pLum;
    u8 *pCb;
    u8 *pCr;
    u8 *pOutBuf;
    u32 busOutBuf;
    u32 outBufSize;
} JpegEncIn;
```

Only the following fields of the structure are used by this function:

`frameHeader`

Controls the creation of JPEG frame headers. See [1] for description of the header fields. The possible values are:
0 – Insert only APP0 and SOS headers. This can be used for motion JPEG when all headers are not needed for successive frames.
1 – Insert all headers (SOI, APP0, DQT, SOF0, DRI, DHT, SOS)

`busLum`

The bus address of the buffer where the input picture's luminance component is located. Has to be 8 bytes aligned, DMA capable linear memory buffer.

`busCb`

The bus address of the buffer where the input picture's first chrominance component (Cb or U) is located. For YCbCr 4:2:0 semiplanar input, this specifies the base of the interleaved chrominance components. Not needed when input format is YCbCr 4:2:2. Has to be 8 bytes aligned, DMA capable linear memory buffer.

`busCr`

The bus address of the buffer where the input picture's second chrominance component (Cr or V) is located. Not needed when input format is YCbCr 4:2:2 or YCbCr 4:2:0 semiplanar. Has to be 8 bytes aligned, DMA capable linear memory buffer.

`pOutBuf`

Pointer to the output buffer where the generated JPEG/JFIF stream is written. Software uses this.

`busOutBuf`

The bus address of the output buffer where the generated JPEG/JFIF stream is written. Has to be 8 bytes aligned, DMA capable linear memory buffer.

`outBufSize`

This is the size in bytes of the stream buffer described above.
Minimum value is 1024.

```
typedef struct
{
    u32 jfifSize;
} JpegEncOut;
```

`jfifSize`

The size in bytes of the generated stream is returned here. If the call was unsuccessful, this value is not relevant at all and it should be ignored.

6 Application Example

The source code given in this chapter is just an example and it is not guaranteed to compile as such.

6.1 Example: Encode VGA image

```
#include "JpegEncApi.h"
#include <stdlib.h>

JpegEncRet ret;
JpegEncInst encoder;
JpegEncCfg cfg;
JpegEncIn encIn;
JpegEncOut encOut;

/* Output buffer size: For most images 1 byte/pixel is enough
   for high quality JPEG */
i32 outbuf_size = 640*480;
u32 pict_bus_address = 0;
u32 *outbuf_virt_address = NULL;
u32 outputBusAdderss = 0;

/* Step 1: Initialize an encoder instance */

cfg.qLevel = 5;
cfg.frameType = JPEGENC_YUV420_PLANAR;
cfg.rotation = JPEGENC_ROTATE_0;
cfg.markerType  = JPEGENC_SINGLE_MARKER;
cfg.unitsType = JPEGENC_DOTS_PER_INCH;
cfg.xDensity = 72;
cfg.YDensity = 72;
cfg.thumbnail = 0; /* no thumbnail */

if((ret = JpegEncInit(&cfg, &encoder)) != JPEGENC_OK)
{
    /* Handle here the error situation */
    goto end;
}

/* Step 2: Configuration of encoder (to mode) */

/* VGA resolution, no cropping */
cfg.inputWidth  = 640;
cfg.codingWidth = 640;
cfg.inputHeight  = 480;
cfg.codingHeight = 480;
cfg.xOffset = 0;
cfg.yOffset = 0;
cfg.codingType = JPEGENC_WHOLE_FRAME;

/* Restart interval 5 MCU rows (= 80 pixel rows), average quantization,
   encode whole frame at once */
cfg.restartInterval = 5;

JpegEncSetFullResolutionMode(encoder, &cfg);

/* Step 3: Allocate output buffer */
```

```
    encIn.pOutBuf           = malloc(outbuf_size);
    encIn.outBufSize        = outbuf_size; /* bytes */

    if (encIn.pOutBuf == NULL)
        goto close;


    /* Step 4: Get the picture and setup input */

    /* Allocate linear memory and read the picture */
    ReadFrame(&pict_bus_address);

    /* Enable all frame headers */
    encIn.frameHeader = 1;

    /* We use one linear buffer for the input image: YYYY...UU...VV... */
    encIn.busLum = pict_bus_address;
    encIn.busCb  = encIn.busLum + cfg.inputWidth * cfg.inputHeight;
    encIn.busCr  = encIn.busCr + (cfg.inputWidth/2) * (cfg.inputHeight/2);


    /* Step 5: Encode the picture */

    /* Loop until the frame is ready */
    do
    {
        ret = JpegEncEncode(encoder, &encIn, &encOut);

        switch (ret)
        {
            case JPEGENC_RESTART_INTERVAL:
            case JPEGENC_FRAME_READY:
                SaveStream(encIn.pOutBuf, encOut.jfifSize);
                break;

            default:
                /* All the others are error codes */
                break;
        }
    }
    while (ret == JPEGENC_RESTART_INTERVAL);

    /* Free output buffer */
    free(encIn.pOutBuf);


close:

    /* Last Step: Release the encoder instance */
    if((ret = JpegEncRelease(encoder)) != JPEGENC_OK)
    {
        /*
           There is not much to do for an error here,
           just a notification message. There has to be a fault
           somewhere else that caused a failure of the
           encoder release
        */
    }

end: /* nothing else to do */
```

References

[1] Information technology – Digital compression and coding of continuous-tone still images, International Standard ISO/IEC IS 10918-1, Oct 20, 1999

VISIBLY BETTER

Headquarters:
Hantro Products Oy
Kiviharjunlenkki 1, FI-90220 Oulu, Finland
Tel: +358 207 425100, Fax: +358 207 425299

Hantro Finland
Lars Sonckin kaari 14
FI-02600
Espoo
Finland
Tel: +358 207 425100
Fax: +358 207 425298

Hantro Germany
Ismaninger Str. 17-19
81675
Munich
Germany
Tel: +49 89 4130 0645
Fax: +49 89 4130 0663

Hantro USA
1762 Technology Drive
Suite 202. San Jose
CA 95110,
USA
Tel: +1 408 451 9170
Fax: +1 408 451 9672

Hantro Japan
Yurakucho Building 11F
1-10-1, Yurakucho
Chiyoda-ku, Tokyo
100-0006, Japan
Tel: +81 3 5219 3638
Fax: +81 3 5219 3639

Hantro Taiwan
6F, No. 331
Fu-Hsing N. Rd.
Taipei
Taiwan
Tel: +886 2 2717 2092
Fax: +886 2 2717 2097

Hantro Korea
#518 ho, Dongburoot, 16-2
Sunaedong, Bundanggu,
Seongnamsi, Kyunggido,
463-825 Korea
Tel: +82 31 718 2506
Fax: +82 31 718 2505

Email: sales@hantro.com
WWW.HANTRO.COM