Application Programming Interface

JPEG Image Encoder on ARM Platforms

Version: 1.2

Release Date: Dec 02nd, 2005

Emuzed, Inc.

Revision History

Version	Date	Description of Change(s)	Authors(s)
0.1	25-11-2004	Initial Draft	G. Nageswara Rao
1.0	29-11-2004	Review comments incorporated	G. Nageswara Rao
1.1	03-01-2005	Support for Writing Configuration data into a given buffer is added.	G. Nageswara Rao
1.2	02-12-2005	Support for YUV420 Planner and YUV422 Planner color formats for input image is added.	G. Nageswara Rao

1 Introduction

This documents explains the Application Programming Interface (API) to use the Emuzed's JPEG encoder on ARM architecture based processors. JPEG encoder library exposes the functions to create, encode and delete operations for the application. The generated images are compliant to JPEG standard [1]. Section 2 explains about the API for the encoder.

2 JPEG Encoder API

This section describes about the Emuzed JPEG encoding API and each of the function services.

2.1 Base Object Definition

2.1.1 tBaseImageEncoder

This structure is base structure for JPEG image encoder. It contains the function pointers for different functionalities provided by JPEG image encoder.

```
typedef struct tBaseImageEncoder
{
      int32 (*vEncodeFrame) (tBaseImageEncoder  *BaseJpegEnc, unsigned char
      *FrameBuffer, unsigned char *outBuf, uint32 *numBytes);
      int32 (*vSetParam) (tBaseImageEncoder *base, uint32 flag, uint32 val);
      int32 (*vGetParam) (tBaseImageEncoder *base, uint32 flag, uint32 *val);
      int32 (*vDelete) (tBaseImageEncoder *base);
      int32 (*vPutCongfigData) (tBaseImageEncoder *base, uint8 *outBuf, uint32
               *numBytes);
}
```

Following table gives details of each member in the base object structure.

Table 1 : Description of Base Object Structure Members (tBaseVideoEncoder)

Member Name	Description
vEncodeFrame	This is a virtual function to encode a raw image or partial image if streaming mode (Refer Sec 3.1) is enabled to JPEG bit-stream
vSetParam	This is a virtual function to set a parameter value of the encoder instance
vGetParam	This is a virtual function to get a parameter value of the encoder instance
vDelete	This is a virtual function to delete the encoder instance
vPutCongfigData	This is a virtual function to write configuration (header data) into the given buffer

2.2 Functions

2.2.1 Create JPEG Encoder Object Instance

Proto Type

```
int gCreateJpegImageEncoder(  tBaseImageEncoder       **base,
```

```
                                tJpegImageEncParam      *imageEncParam)
```

Description

Creates the JPEG encoder instance and returns the handle to JPEG encoder.

Parameters

- `Base` : `[OUT]` Handle to JPEG encoder instance.
- `ImageEncParam` : `[IN]` JPEG encoder parameters are passed through pointer to the Structure tJpegImageEncParam

Return Value

Returns `E_SUCCESS` if create is successful otherwise return error value corresponding to the errors given in the following table.

Table 2 Error code description

Error Code	Error Value	Error Description
E_OUT_OF_MEMORY	-2	Failure of memory allocation
E_INVALID_SIZE	-13	Invalid image width and height values.
E_INVALID_MODE	-12	Invalid streaming mode parameter.
E_INVALID_FORMAT	-11	Invalid input color format

2.2.2 Encode One Raw image to JPEG Bit-stream

Proto Type

```
int vEncodeFrame     (tBaseImageEncoder   *BaseJpegEnc,
                      unsigned char       *FrameBuffer,
                      unsigned char       *outBuf,
                      unsigned int        *numBytes)
```

Description

Encodes one raw image or partial image (if streaming mode flag is enabled) in FrameBuffer to outBuf. The raw video frame must start from *FrameBuffer[0]* and *outBuf* must be big enough to hold all the encoded bytes. If the streaming mode flag is enabled outBuf must be big enough to hold at least number of bytes required to encode one MCU row. *numBytes* parameter carries the number of bytes allocated for the outBuf, and holds the number of encoded bytes on return of the function.

If the streaming mode flag is enabled vEncodeFrame function need to be called multiple times as long as the function returns E_NOT_COMPLETE flag indicating that encoding is not completed. Every time the partial image is encoded the outpBuff contents need to be captured by the application. After each call the encoded bytes (signaled in numBytes parameter) are overwritten into outBuff.

Parameters

- `BaseJpegEnc` : `[IN]` Pointer to JPEG encoder object.
- `FrameBuffer` : `[IN]` Pointer to input buffer of input raw image.

- `OutBuf` : `[OUT]` Buffer in which the encoded stream needs to be written.[1,2] . The size of the buffer that needs to be allocated can be obtained using the gGetParamJpegImageEncoder API as explained in Section 2.1.4.
- `NumBytes` : `[IN/OUT]` Number of bytes that are allocated for output buffer / Number of bytes generated after encoding an image(or part of the image).

Return Value

Returns `E_SUCCESS` if encoding is completed and successful or returns E_NOT_COMPLETE to indicate encoding is not completed and required to invoke same function until E_SUCCESS is returned. Following table lists the possible return values and corresponding description.

`E_SUCCESS`	0	One raw image is encoded successfully
`E_FAILURE`	-1	Encoder failed to encode the given raw image.
`E_NOT_COMPLETE`	-9	Encoding is not completed, required to invoke the vEncodeFrame function again to complete encoding.
`E_INSUFFICIENT_OUTBUFF`	-10	Output buffer is insufficient for encoding the image or one MCU row (if streaming mode is enabled).

2.2.3 Delete the JPEG Encoder Object Instance

Proto Type

```
Int vDelete (tBaseImageEncoder * BaseJpegEnc)
```

Description:

Deletes the JPEG encoder object.

Parameters:

- `BaseJpegEnc` : `[IN]` Handle to JPEG encoder object.

Return Value

`E_SUCCESS` : The encoder object is deleted successfully.

`E_FAILURE` : Encoder object deletion failed.

2.2.4 Writing configuration data into a buffer

Proto Type

```
int32 (*vPutCongfigData) (tBaseImageEncoder *base, uint8 *outBuf, uint32
    *numBytes)
```

Description:

Writes the configuration data (JFIF header information) into the given buffer.

[1] Enough buffer to hold the encoded stream corresponding to raw image should be given.

[2] The buffer should be aligned on 4-byte boundary.

Parameters:

- `BaseJpegEnc` : `[IN]` Handle to JPEG encoder object.
- `uint8` : `[IN]` Pointer to output buffer into which data should be written. The size of the buffer should me enough to hold the header data.
- `uint32` : `[IN/OUT]` Number of bytes that are allocated for configuration data buffer / Number of bytes generated after writing configuration data into the buffer.

Return Value

Returns `E_SUCCESS`, if generation of configuration data is successful. Following table lists the possible return values and corresponding description.

`E_SUCCESS`	0	One raw image is encoded successfully
`E_FAILURE`	-1	Encoder failed to generate configuration data into given buffer.
`E_INSUFFICIENT_OUTBUFF`	-10	Configuration data buffer given is insufficient for writing the header data.

2.2.5 Retrieve JPEG Encoding Parameter

Proto Type

```
Int32 vGetParam ( tBaseImageEncoder *BaseJpegEnc,
                  Unsigned int      flag,
                  Unsigned int      *val)
```

Description

Get the parameter value from the encoder.

Parameters

- `BaseJpegEnc` : `[IN]` Handle to JPEG encoder object.
- `flag` : `[IN]` Parameter flag to be read. Supported flags are listed in the following table with appropriate description.

`GET_MAX_OUTBUF_SIZE`	10	To return the maximum size of the output bit-stream buffer needed for encoding the given input raw image or one row of MCUs if streaming mode is enabled. The buffer size is returned in number of bytes.
`GET_CONFIG_DATA_SIZE`	11	To return the maximum size of the buffer needed for generating the header into the buffer. The buffer size is returned in number of bytes.

- `val` : `[OUT]` Pointer to return the parameter value

Return Value

`E_SUCCESS` : The encoder parameter retrieved successfully.

`E_FAILURE` : The encoder could not retrieve the parameter value.

3 Data Types

3.1 tJpegImageEncParam

This structure is used to store and pass the user configuration parameters for the encoding process at the time of encoder object creation.

```
typedef struct
{
      int               maxXDimension;
      int               maxYDimension;
      int               QualityFactor;
      int               InputColorFormat;
      int               StreamingMode;
} tJpegImageEncParam;
```

maxXDimension: Maximum dimension value in horizontal direction among all the components of the input raw image. MaxXDimension should be integer multiples of 16.

maxYDimension: Maximum dimension value in vertical direction among all the components of the input raw image. MaxYDimension should be integer multiples of 16.

QualityFactor: This parameter is used to control the quality of encoding. It can take values from 1 to 100. Quality factor 1 produces least quality and 100 gives best quality. If the value of input QualityFactor is out of the range (1, 100) it's value shall be forced to 50 internally.

InputColorFormat: This parameter is to signal the color format of the input image to the encoder. The supported color formats are YUV 420Planner (1), YUV422 Planner (2) and YUV 422 Interleaved in YUYV color order (3).

StreamingMode: This flag is to indicate whether encoding needs to be done in one shot (if the flag is set to 'E_OFF') for whole image or in multiple times (if the flag is set 'E_ON') based on the available output buffer size. However the output buffer size must be at least enough to hold bytes to encode one MCU row.

4 References

[1] JPEG Standard, "Information Technology- Digital Compression and Coding of Continuous-Tone Still Images- Requirements and Guidelines," Recommendation T.81, ITU, September 1992.