JPEG Encoder/Decoder API Specification (WinCE/WM)

S3C6400

May 15, 2008

REV 4.5

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product

Revision History

Revision No	Description of Change	Refer to	Author(s)	Date
1.0	Initial Version	-	Jiyoung Shin	2007-05-30
2.0	Upgrade for WinCE6.0 and WM6.0	-	Jiyoung Shin	2007-08-13
3.0	Change Buffer Allocation	-	Jiyoung Shin	2007-11-09
4.5	Support 3M pixel	-	Jiyoung Shin	2008-05-15

Contents

SAMSUNG
ELECTRONICS

1 Introduction

1.1 Purpose

The purpose of the document is to describe the JPEG Encoder/Decoder API for easy portability into different platforms by developers.

1.2 Scope

The scope of this document is to describe
- Software architecture of Encoder/Decoder
- Data structures and API used for Encoder/Decoder
- Usage example of Encoder/Decoder

1.3 Intended Audience

Intended Audience	Tick whenever Applicable
Project Manager	Yes
Project Leader	Yes
Project Team Member	Yes
Test Engineer	Yes

1.4 Definitions, Acronyms, and Abbreviations

Abbreviations	Description
JPEG	Joint Photographic Exports Grout
MCU	Minimum Coded Unit
EXIF	Exchangeable Image File Format
API	Application Program Interface

1.5 References

Number	Reference	Description
1	SMDK6400_WinCE6.0_FMD_PortingGuide.doc	OS porting guide
2	SMDK6400_WinMobile6.0_VideoDriver_UserManual.doc	Video Driver API
3	S3C6400_WinCE6.0_WM6.0_JPEG_API_REV4.5_20080515.doc	JPEG API specification

2 Software Architecture

2.1 Overview

Software architecture of JPEG Encoder/Decoder package mainly comprises of two major modules:
- Common JPEG API
- JPEG Encoder/Decoder Device Driver

Common JPEG API provides the same interface to user application even if operating system and Codec(H/W or S/W) is different. Figure 1. shows relationship among user application, API, device driver and codec

Common JPEG API consists of 5 operations
- Initialize : Initialize encoder/decoder. i.e. initialize memory and default variables.

- Buffer Management : Get input/output buffer address
- Execute : Execute encoding/decoding process
- Configuration : Set/Get parameters to execute encoding/decoding
- Finalization : Release encoder/decoder resources
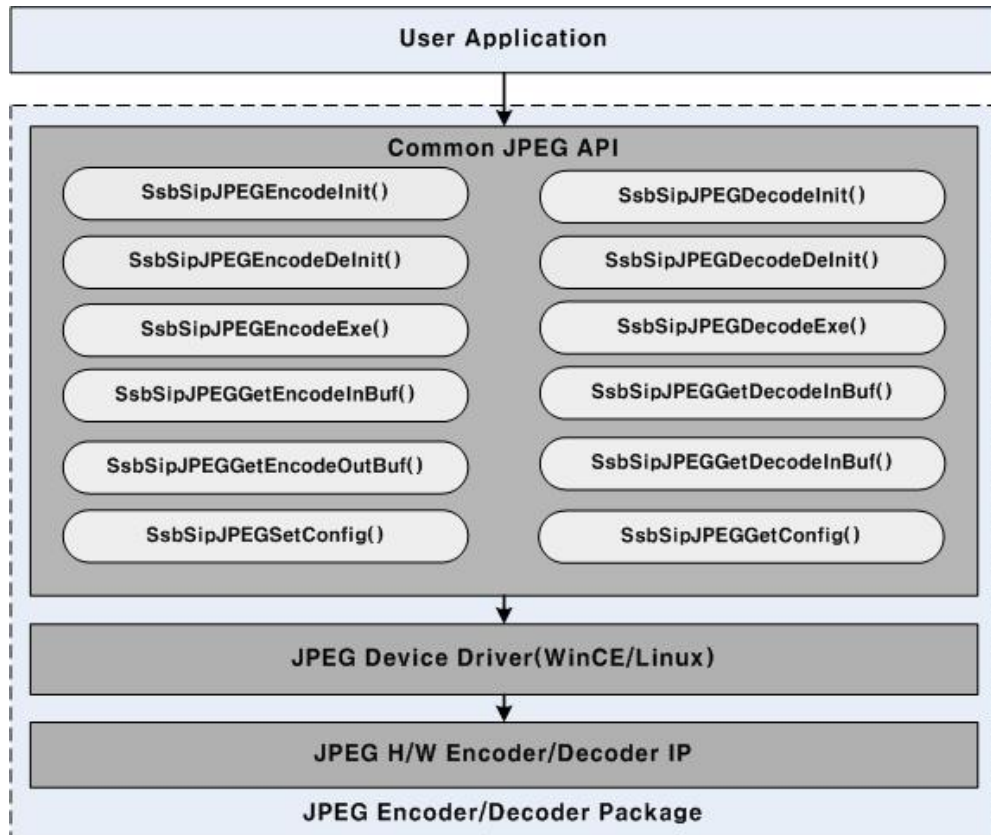
Figure 1. Architecture of JPEG API

2.2 Features

- Compression/decompression up to 2048*1536.

- Encoding format: interleaved YCBYCR (input format of JPEG engine).

- Decoding format: YCbCr4:4:4, YCbCr4:2:2, YCbCr4:2:0, YCbCr4:1:1 or Gray.

- Support Exif file and Thumbnail encoding.

- The resolution of thumbnail is up to 160*120.

- Support 4 different image quality levels during encoding.

- Support of direct compression from the camera output.

- Support of compression of memory data in interleaved YCBYCR.

2.3 Encoding/Decoding Process

Figure 2. shows the process of encoding/decoding. For both encoder and decoder, H/W encoder/decoder engine only support multiple resolution of MCU size. For example, MCU size of YUV420 is 16*16. therefore, if resolution is 200*200, encoding/decoding will fail.

S/W post processor overcome this limitation for decoder. But for encoder, the limitation still exits. Output format of H/W decoder is only interleaved YCBYCR.

SAMSUNG
ELECTRONICS

For camera, Exif file format is supported in Encoder. But, Exif and thumbnail is S/W. If you want to get better performance. You may disable Exif and thumbnail encoding.
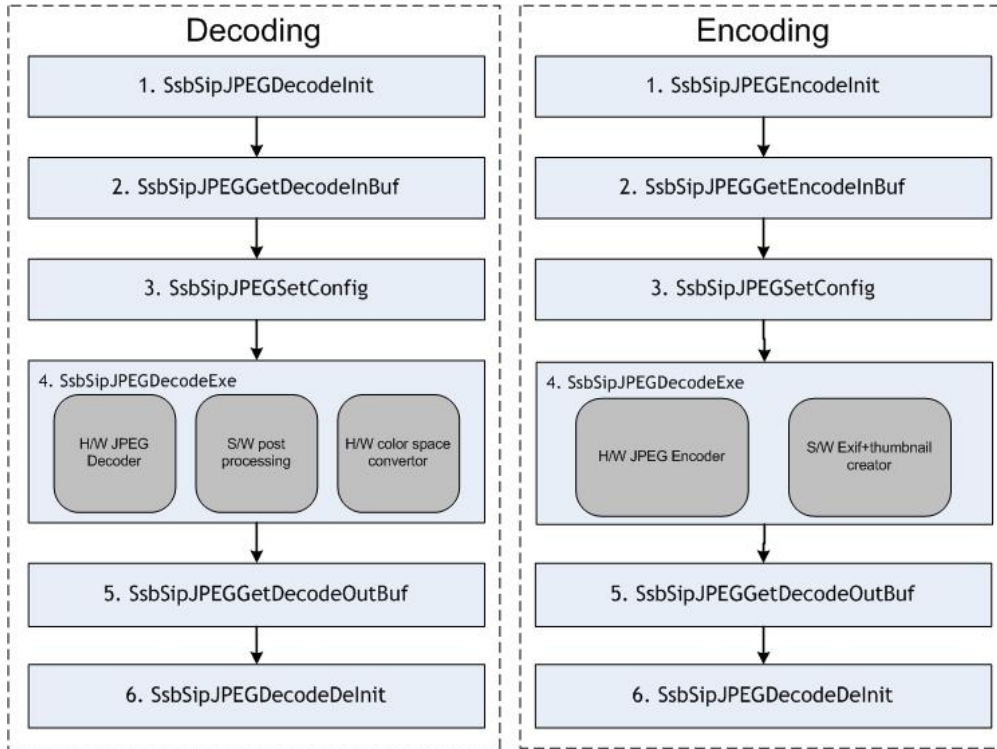
Figure 2. Encoding/Decoding Process

2.4 Memory Management

Operating system reserve memory to assign JPEG H/W IP.
To increase efficiency of buffer usage, SsbSipJPEGxxxxBuf return the logical address of buffer. The logical address is mapped to pre-assigned JPEG H/W IP buffer. Therefore, we reduced memcpy overhead.
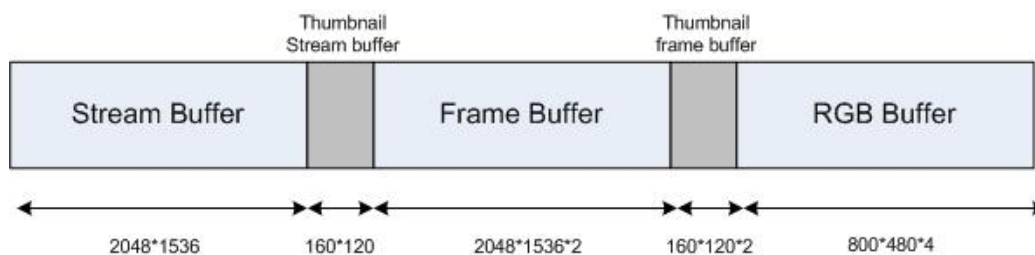
Figure 3. Memory Usage

In figure 3, stream buffer have JPEG file stream. And, Frame buffer have decoded YCBYCR. In the case of encoding, encoding frame(YCBYCR) is put in frame buffer. And, encoded JPEG file is put in stream buffer.

For thumbnail decoding, we assigned thumbnail stream/frame buffer. After encoding frame(YCBYCR) is put frame buffer. It is resized to thumbnail frame buffer with thumbnail resolution. Resized frame is finally encoded to thumbnail stream buffer.

RGB buffer is only used for color space conversion. When you display Jpeg, you may need RGB format. If you use H/W post processor, conversion performance is much faster than s/w conversion performance. For detail usage, refer test.c in test directory.

2.5 Change Memory Size

If you want to change memory size, you must change bellows.

1. In Jpgmem.h, *MAX_JPG_WIDTH, MAX_JPG_HEIGHT*
2. In JpgAPI.h, *MAX_JPG_WIDTH, MAX_JPG_HEIGHT*
3. In Config.bib, *MFC_JPG* size must be `bigger than` *JPG_TOTAL_BUF_SIZE* `in jpgmem.h`
4. `After you change MFC_JPG, you must change` *RAMLEN* `in Config.bib`
5. `In image_cfg.inc, image_cfg.h,` *IMAGE_MFC_BUFFER_SIZE* `must be bigger than` *JPG_TOTAL_BUF_SIZE*.

3 Data Structure

3.1 Exif File Format

ExifFileInfo	
UCHAR Make[32]	Manufacturer name
Char Model[32]	Model name
Char Version[32]	Version number
Char DateTime[32]	Date
Char CopyRight[32]	Copy right
UINT Height	Primary image height
UINT Width	Primary image width
UINT Orientation	Direction for display
UINT ColorSpace	Color space of input frame
UINT Process	JPEG process
UINT Flash	Flash on/off
UINT FocalLengthNum	The length of focal length filed
UINT ExposureTimeNum	Exposure time
UINT ExposureTimeDen	Exposure time
UINT FNumberNum	
UINT FNumberDen	
UINT ApertureFNumber	
Int SubjectDistanceNum	Subject distance
Int SubjectDistanceDen	Subject distance
UINT CCDWidth	
Int ExposureBiasNum	Exposure bias
Int ExposureBiasDen	Exposure bias
Int WhiteBalance	whitebalnce
UINT MeteringMode	Metering mode
Int ExposureProgram	Exposure Program
UINT ISOSpeedRatings[2]	ISO speed ratings
UINT FocalPlaneXResolutionNum	Focal plane X resolution
UINT FocalPlaneXResolutionDen	Focal plane X resolution
UINT FocalPlaneYResolutionNum	Focal plane Y resolution
UINT FocalPlaneYResolutionDen	Focal plane Y resolution
UINT FocalPlaneResolutionUnit	Focal plane resolution unit
UINT XResolutionNum	X resolution
UINT XResolutionDen	X resolution
UINT YResolutionNum	Y resolution
UINT YResolutionDen	Y resolution
UINT RUnit	
INT BrightnessNum	brightness
INT BrightnessDen	brightness
UCHAR UserComments[150]	User comment

4 API

4.1 Initialization

SsbSipJPEGDecodeInit()	
Description	This function is • to initialize codec variables • to assign memory for JPEG H/W codec register
Syntax	void * SsbSipJPEGDecodeInit (void);
Parameters	
Returns	Return handle after JPEG Decoder initialization.

SsbSipJPEGEncodeInit ()	
Description	This function is • to initialize codec variables • to assign memory for JPEG H/W codec register
Syntax	void * SsbSipJPEGEncodeInit (void);
Parameters	
Returns	Return handle after JPEG Encoder initialization.

4.2 Decode JPEG

SsbSipJPEGDecodeExe()	
Description	This function is • to decode JPEG file
Syntax	JPEG_ERRORTYPE SsbSipJPEGDecodeExe (void *openHandle);
Parameters	[IN] openHandle Return value from SsbSipJPEGDecodeInit ().
Returns	JPEG_ERRORTYPE returns error code.

4.3 Encode JPEG

SsbSipJPEGEncodeExe()	
Description	This function is • to encode JPEG

Syntax	JPEG_ERRORTYPE SsbSipJPEGEncodeExe (void *openHandle, ExifFileInfo *Exif);
Parameters	[IN] openHandle Return value from SsbSipJPEGEncodeInit (). [IN] Exif Exif file Parameters. if It is NULL, Exif file is not included in JPEG file.
Returns	JPEG_ERRORTYPE returns error code.

4.4 Memory Access

SsbSipJPEGGetDecodeInBuf ()	
Description	This function is • to get memory address for decoding input buffer
Syntax	Void * SsbSipJPEGGetDecodeInBuf (void *openHandle, long size);
Parameters	[IN] size Allocation size(byte) [IN]openHandle Return value from SsbSipJPEGxxxInit ().
Returns	It returns memory address of decoding input buffer. In H/W codec, physical address of decoding input buffer is statically set during initialization. Size is limited by 2048*1536 *2 byte.

SsbSipJPEGGetDecodeOutBuf ()	
Description	This function is • to get memory address for decoding output buffer
Syntax	Void * SsbSipJPEGGetDecodeOutBuf (void *openHandle, long *size);
Parameters	[IN] size Allocation size(byte) [IN]openHandle Return value from SsbSipJPEGxxxInit ().
Returns	It returns memory address of Frame buffer. In H/W codec, physical address of decoding output buffer is statically set during initialization. Size is limited by 2048*1536 byte.

SsbSipJPEGGetDecodeOutPhyBuf ()	
Description	This function is • to get physical address for decoding output buffer
Syntax	Void * SsbSipJPEGGetDecodeOutPhyBuf (void *openHandle);
Parameters	[IN]openHandle Return value from SsbSipJPEGxxxInit ().
Returns	It returns physical address of Frame buffer. This value is useful for video Driver API

SsbSipJPEGGetEncodeInBuf ()	
Description	This function is • to get memory address for encoding input buffer
Syntax	Void * SsbSipJPEGGetEncodeInBuf (void *openHandle, long size);
Parameters	[IN] size Allocation size(byte) [IN]openHandle Return value from SsbSipJPEGxxxInit ().
Returns	It returns memory address of encoding input buffer. In H/W codec, physical address of encoding input buffer is statically set during initialization. Size is limited by 2048*1536 byte.

SsbSipJPEGGetEncodeOutBuf ()	
Description	This function is • to get memory address for encoding output buffer
Syntax	Void * SsbSipJPEGGetStreamBuf (void *openHandle, long size);
Parameters	[IN] size Allocation size(byte) [IN]openHandle Return value from SsbSipJPEGxxxInit ().
Returns	It returns memory address of encoding output m buffer. In H/W codec, physical address of encoding output buffer is statically set during initialization. Size is limited by 2048*1536 *2 byte.

SsbSipJPEGGetRGBBuf ()	
Description	This function is • to get memory address for RGB buffer
Syntax	Void * SsbSipJPEGGetRGBBuf (void *openHandle, INT32 width, INT32 height)
Parameters	[IN]openHandle Return value from SsbSipJPEGxxxInit (). [IN] width RGB width(byte) [IN] height RGB height(byte)
Returns	It returns memory address of RGB buffer. You can use this after convert YCBYCR to RGB using Video Driver API.

SsbSipJPEGGetDecodeOutPhyBuf ()	
Description	This function is • to get physical address for decoding output buffer
Syntax	Void * SsbSipJPEGGetRGBPhyBuf(void *openHandle, INT32 width, INT32 height)
Parameters	[IN]openHandle Return value from SsbSipJPEGxxxInit (). [IN] width RGB width(byte) [IN] height RGB height(byte)
Returns	It returns physical address of RGB buffer. You can use this after convert YCBYCR to RGB using Video Driver API. This physical address is useful for display

4.5 Configuration

SsbSipJPEGSetConfig ()	
Description	This function is • to set codec variables

SAMSUNG ELECTRONICS

Syntax	JPEG_ERRORTYPE SsbSipJPEGSetConfig (void *openHandle, JPEGConf type, void value);
Parameters	[IN] openHandle Return value from SsbSipJPEGxxxInit (). [IN] type Configuration type defined 5.Defintion and Error codes [IN] value Configuration value.
Returns	JPEG_ERRORTYPE returns error code.

SsbSipJPEGGetConfig ()	
Description	This function is • to get codec variables
Syntax	JPEG_ERRORTYPE SsbSipJPEGGetConfig (void *openHandle, JPEGConf type, void *value);
Parameters	[IN] openHandle Return value from SsbSipJPEGxxxInit (). [IN] type Configuration type defined 5.Defintion and Error codes [OUT] value Configuration value
Returns	JPEG_ERRORTYPE returns error code.

4.6 Finalization

SsbSipJPEGDecodeDeInit ()	
Description	This function is • to release codec resources
Syntax	JPEG_ERRORTYPE SsbSipJPEGDecodeDeInit (void *openHandle);
Parameters	[IN] openHandle Return value after JPEG initialization.
Returns	JPEG_ERRORTYPE returns error code.

SsbSipJPEGEncodeDeInit ()	
Description	This function is • to release codec resources

SAMSUNG
ELECTRONICS

Syntax	JPEG_ERRORTYPE SsbSipJPEGEncodeDeInit (void *openHandle);
Parameters	[IN] openHandle Return value after JPEG initialization.
Returns	JPEG_ERRORTYPE returns error code.

5 Definition and Error Codes

5.1 Configuration

Definition	Description
JPEG_GET_DECODE_WIDTH	To get width of JPEG file.
JPEG_GET_DECODE_HEIGHT	To get height of JPEG file.
JPEG_GET_SAMPING_MODE	To get sampling mode of JPEG file.
JPEG_SET_ENCODE_WIDTH	To set width of JPEG file for encoding.
JPEG_SET_ENCODE_HEIGHT	To set height of JPEG file for encoding.
JPEG_SET_ENCODE_QUALITY	To set image quality level.
JPEG_SET_SAMPING_MODE	To set sampling mode for encoding
JPEG_SET_ENCODE_THUMBNAIL	To encode thumbnail picture when encoding.
JPEG_SET _THUMBNAIL_WIDTH	To set width of thumbnail picture
JPEG_SET _THUMBNAIL_HEIGHT	To set height of thumbnail picture

5.2 Sampling mode

Name	Description
JPG_444	YUV444
JPG_422	YUV422
JPG_420	YUV420
JPG_411	YUV411
JPG_400	YUV400
JPG_UNKNOWN	Unknown type

5.3 Image Quality

Name	Description
JPG_QUALITY_LEVEL_1	Compression ratio is about 30%
JPG_QUALITY_LEVEL_2	Compression ratio is about 24%
JPG_QUALITY_LEVEL_3	Compression ratio is about 20%
JPG_QUALITY_LEVEL_4	Compression ratio is about 14%

SAMSUNG
ELECTRONICS

5.4 Error Codes

Name	Description
JPEG_FAIL	General failure
JPEG_OK	General success
JPEG_ENCODE_FAIL	Encoding failure
JPEG_ENCODE_OK	Encoding success
JPEG_DECODE_FAIL	Decoding failure
JPEG_DECODE_OK	Decoding success
JPEG_HEADER_PARSE_FAIL	Header parse error during decoding
JPEG_HEADER_PARSE_OK	Successful header paring during decoding

6 Test Parameters

6.1 Decoder

1. input
   - input file: JPEG format. For example, input.jpg
2. output
   - output file: YCBYCR format. For example output.yuv
   - width : width of jpg file. use JPEG_GET_DECODE_WIDTH
   - height : height of jpg file. use JPEG_GET_DECODE_HEIGHT
   - sampling mode : sampling mode of jpg file. use JPEG_GET_SAMPLING_MODE

6.2 Encoder

1. input
   - input file : YCBYCR format. For example input.yuv
   - Width : width of YUV. Use JPEG_SET_ENCODE_WIDTH.
   - height : height of YUV. Use JPEG_SET_ENCODE_HEIGHT.
   - sampling mode : sampling mode of jpg file. Use JPEG_SET_SAMPLING_MODE
   - image quality : image quality of jpg file. Use JPEG_SET_ENCODE_QUALITY
   - thumbnail flag : whether thumbnail is included or not. Use JPEG_SET_ENCODE_THUMBNAIL
   - thumbnail width: width of thumbnail. Use JPEG_SET _THUMBNAIL_WIDTH
   - thumbnail height: height of thumbnail. Use JPEG_SET _THUMBNAIL_HEIGHT
   - Exif info: Exif information. Pass parameters via SsbSipJPEGEncodeExe()
2. output
   - output file: JPEG format. For example, output.jpg

Annex A (JPEG Decoder Usage)

This is an example program to give information how to use API.
We will not accept responsibility for any errors about example program.

```
void TestDecoder()
{
        char *InBuf = NULL;
        char *OutBuf = NULL;
        char *OutPhyBuf = NULL;
        char *OutRGBBuf = NULL;
        char *OutRGBPhyBuf = NULL;
        FILE *fp;
        FILE *CTRfp;
        UINT32 fileSize;
        UINT32 streamSize;
        void *handle;
        INT32 width, height, samplemode;
        JPEG_ERRORTYPE ret;
        char outFilename[128];
        char inFilename[128];
        BOOL result = TRUE;
#if (FPS == 1)
        INT32   decodeTime;
#endif
        DWORD startTime;


        printf("-----------------------Decoder Test Start --------------------\n");

        /////////////////////////////////////////////////////////////
        // 0. Get input/output file name                    //
        /////////////////////////////////////////////////////////////
        CTRfp = fopen(CTRL_FILE_NAME, "rb");
        if(CTRfp == NULL){
                printf("file open error : %s\n", CTRL_FILE_NAME);
                return;
        }

        do{
                memset(outFilename, 0x00, sizeof(outFilename));
                memset(inFilename, 0x00, sizeof(inFilename));

                fscanf(CTRfp, "%s", inFilename);

                if(inFilename[0] == '#'){
                        printf("-----------------------Decoder Test Done --------------------\n");
                        fclose(CTRfp);
                        return;
                }

                fscanf(CTRfp, "%s", outFilename);

                if(inFilename == NULL || outFilename == NULL){
                        printf("read file error\n");
                        printf("-----------------------Decoder Test Done --------------------\n");
                        fclose(CTRfp);
                        return;
```

```
        }

        printf("inFilename : %s \noutFilename : %s\n", inFilename, outFilename);
        /////////////////////////////////////////////////////////////
        // 1. handle Init                                          //
        /////////////////////////////////////////////////////////////
        #if (FPS == 1)
                decodeTime = GetTickCount();
        #endif
        handle = SsbSipJPEGDecodeInit();
        #if (FPS == 1)
                decodeTime = GetTickCount() - decodeTime;
                printf( "Initialization Time : %d \n", decodeTime);
        #endif
        if(handle == NULL){
                printf("Decoder Init failed\n");
                break;
        }

        /////////////////////////////////////////////////////////////
        // 2. open JPEG file to decode                             //
        /////////////////////////////////////////////////////////////
        fp = fopen(inFilename, "rb");
        if(fp == NULL){
                result = FALSE;
                printf("file open error : %s\n", inFilename);
                break;
        }
        fseek(fp, 0, SEEK_END);
        fileSize = ftell(fp);
        fseek(fp, 0, SEEK_SET);

        printD("filesize : %d\n", fileSize);

        /////////////////////////////////////////////////////////////
        // 3. get Input buffer address                             //
        /////////////////////////////////////////////////////////////
        InBuf = SsbSipJPEGGetDecodeInBuf(handle, fileSize);
        if(InBuf == NULL){
                printf("Input buffer is NULL\n");
                result = FALSE;
                break;
        }
        printD("inBuf : 0x%x\n", InBuf);

        /////////////////////////////////////////////////////////////
        // 4. put JPEG frame to Input buffer                       //
        /////////////////////////////////////////////////////////////
        fread(InBuf, 1, fileSize, fp);
        fclose(fp);


        /////////////////////////////////////////////////////////////
        // 5. Decode JPEG frame                                    //
        /////////////////////////////////////////////////////////////
        #if (FPS == 1)
                decodeTime = GetTickCount();
        #endif

        ret = SsbSipJPEGDecodeExe(handle);
```

```
                    #if (FPS == 1)
                            decodeTime = GetTickCount() - decodeTime;
                            printf( "decodeTime : %d \n", decodeTime);
                    #endif

                    if(ret != JPEG_OK){
                            printf("Decoding failed\n");
                            result = FALSE;
                            break;
                    }

                    //////////////////////////////////////////////////////////
                    // 6. get Output buffer address                    //
                    //////////////////////////////////////////////////////////
                    OutBuf = SsbSipJPEGGetDecodeOutBuf(handle, &streamSize);
                    if(OutBuf == NULL){
                            printf("Output buffer is NULL\n");
                            result = FALSE;
                            break;
                    }
                    printD("OutBuf : 0x%x streamsize : %d\n", OutBuf, streamSize);

                    //////////////////////////////////////////////////////////
                    // 7. get decode config.                           //
                    //////////////////////////////////////////////////////////
                    SsbSipJPEGGetConfig(JPEG_GET_DECODE_WIDTH, &width);
                    SsbSipJPEGGetConfig(JPEG_GET_DECODE_HEIGHT, &height);
                    SsbSipJPEGGetConfig(JPEG_GET_SAMPING_MODE, &samplemode);

                    printf("width : %d height : %d samplemode : %d\n", width, height, samplemode);

                    //////////////////////////////////////////////////////////
                    // 8. wirte output file                            //
                    //////////////////////////////////////////////////////////
#if (TEST_DECODE_OUTPUT_YCBYCR == 1)
                    DecodeFileOutYCBYCR(OutBuf, streamSize, outFilename);        // YCBYCR
interleaved
#elif (TEST_DECODE_OUTPUT_YUV422 == 1)
                    DecodeFileOutYUV422(OutBuf, streamSize, outFilename);        // yuv422 non-
interleaved
#elif (TEST_DECODE_OUTPUT_RGB16 == 1)
        //RGB16
                    OutPhyBuf = SsbSipJPEGGetDecodeOutPhyBuf(handle);
                    OutRGBPhyBuf = SsbSipJPEGGetRGBPhyBuf(handle, LCD_X, LCD_Y);
                    if(ConvertYCBYCRToRGB((int)OutPhyBuf, width, height,
                                                    POST_SRC_YUV422_CRYCBY,
                                                    (int)OutRGBPhyBuf, LCD_X, LCD_Y,
                                                    POST_DST_RGB16) == FALSE){
                            printf("ConvertYCBYCRToRGB error\n");
                            result = FALSE;
                            break;
                    }
                    OutRGBBuf = SsbSipJPEGGetRGBBuf(handle, LCD_X, LCD_Y);
                    DecodeFileOutRGB16ToPPM(OutRGBBuf, LCD_X, LCD_Y, outFilename);
#elif(TEST_DECODE_OUTPUT_LCD == 1)
                    OutPhyBuf = SsbSipJPEGGetDecodeOutPhyBuf(handle);
                    OutRGBPhyBuf = SsbSipJPEGGetRGBPhyBuf(handle, LCD_X, LCD_Y);
                    startTime = GetTickCount();
```

```
                if(ConvertYCBYCRToRGB((int)OutPhyBuf, width, height,
                                              POST_SRC_YUV422_CRYCBY,
                                              (int)OutRGBPhyBuf, LCD_X, LCD_Y,
                                              POST_DST_RGB16) == FALSE){
                        printf("ConvertYCBYCRToRGB error\n");
                        result = FALSE;
                        break;
                }
                printf("converting time : %d\n", GetTickCount() - startTime);
                printf("\n\n This image will be disappeared after %d seconds......\n\n",
DISPLAY_TIME);
                DisplayJPEG((int)OutRGBPhyBuf, LCD_X, LCD_Y, LCD_X, LCD_Y, DISPLAY_TIME);
#endif

                //////////////////////////////////////////////////////////
                // 9. finalize handle                       //
                //////////////////////////////////////////////////////////
                SsbSipJPEGDecodeDeInit(handle);
                Sleep(5);
        }while(1);

        if(result == FALSE){
                SsbSipJPEGDecodeDeInit(handle);
        }

        fclose(CTRfp);
        printf("-----------------------Decoder Test Done --------------------\n");
}
```

Annex B (JPEG Encoder Usage)

```
void TestEncoder()
{
        char *InBuf = NULL;
        char *InThumbBuf = NULL;
        char *OutBuf = NULL;
        FILE *fp;
        FILE *CTRfp;
        JPEG_ERRORTYPE ret;
        UINT32 fileSize;
        UINT32 frameSize;
        void *handle;
        ExifFileInfo *ExifInfo;
        char outFilename[128];
        char inFilename[128];
```

```
        char widthstr[8], heightstr[8];
        INT32 width, height;
        BOOL result = TRUE;
#if (FPS == 1)
        INT32   encodeTime;
#endif

        printf("-----------------------Encoder Test start--------------------\n");
        /////////////////////////////////////////////////////////////////
        // 0. Get input/output file name                      //
        /////////////////////////////////////////////////////////////////
        CTRfp = fopen(CTRL_FILE_NAME, "rb");
        if(CTRfp == NULL){
                printf("file open error : %s\n", CTRL_FILE_NAME);
                return;
        }

        do{
                memset(outFilename, 0x00, sizeof(outFilename));
                memset(inFilename, 0x00, sizeof(inFilename));
                memset(widthstr, 0x00, sizeof(widthstr));
                memset(heightstr, 0x00, sizeof(heightstr));

                fscanf(CTRfp, "%s", inFilename);
                if(inFilename[0] == '#'){
                        printf("-----------------------Encoder Test Done--------------------\n");
                        fclose(CTRfp);
                        return;
                }

                fscanf(CTRfp, "%s", outFilename);
                fscanf(CTRfp, "%s", widthstr);
                fscanf(CTRfp, "%s", heightstr);
                width = (INT32)atoi(widthstr);
                height = (INT32)atoi(heightstr);

                if(inFilename == NULL || outFilename == NULL){
                        printf("read file error\n");
                        printf("-----------------------Encoder Test Done--------------------\n");
                        fclose(CTRfp);
                        return;
                }

                printf("inFilename : %s \noutFilename : %s width : %d height : %d\n",
                            inFilename, outFilename, width, height);
                /////////////////////////////////////////////////////////////////
                // 1. handle Init                               //
                /////////////////////////////////////////////////////////////////
                handle = SsbSipJPEGEncodeInit();
                if(handle == NULL)
                        break;

                /////////////////////////////////////////////////////////////////
                // 2. set decode config.                          //
                /////////////////////////////////////////////////////////////////
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_SAMPING_MODE, JPG_420)) !=
JPEG_OK){
                        result = FALSE;
                        break;
                }
```

```
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_ENCODE_WIDTH, width)) != JPEG_OK){
                        result = FALSE;
                        break;
                }
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_ENCODE_HEIGHT, height)) != JPEG_OK){
                        result = FALSE;
                        break;
                }
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_ENCODE_QUALITY,
JPG_QUALITY_LEVEL_2)) != JPEG_OK){
                        result = FALSE;
                        break;
                }
#if (TEST_ENCODE_WITH_THUMBNAIL == 1)
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_ENCODE_THUMBNAIL, TRUE)) !=
JPEG_OK){
                        result = FALSE;
                        break;
                }
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_THUMBNAIL_WIDTH, 64)) != JPEG_OK){
                        result = FALSE;
                        break;
                }
                if((ret = SsbSipJPEGSetConfig(JPEG_SET_THUMBNAIL_HEIGHT, 48)) != JPEG_OK){
                        result = FALSE;
                        break;
                }
#endif

                //////////////////////////////////////////////////////////
                // 3. open JPEG file to decode                  //
                //////////////////////////////////////////////////////////
                fp = fopen(inFilename, "rb");
                if(fp == NULL){
                        printf("file open error : %s\n", inFilename);
                        result = FALSE;
                        break;
                }
                fseek(fp, 0, SEEK_END);
                fileSize = ftell(fp);
                fseek(fp, 0, SEEK_SET);

                //////////////////////////////////////////////////////////
                // 4. get Input buffer address                  //
                //////////////////////////////////////////////////////////
                printD("filesize : %d\n", fileSize);
                InBuf = SsbSipJPEGGetEncodeInBuf(handle, fileSize);
                if(InBuf == NULL){
                        result = FALSE;
                        break;
                }
                printD("inBuf : 0x%x\n", InBuf);

                //////////////////////////////////////////////////////////
                // 5. put YUV stream to Input buffer             //
                //////////////////////////////////////////////////////////
                fread(InBuf, 1, fileSize, fp);
                fclose(fp);
```

```
///////////////////////////////////////////////////////////
// 6. Make Exif info parameters                    //
///////////////////////////////////////////////////////////
ExifInfo = (ExifFileInfo *)malloc(sizeof(ExifFileInfo));
memset(ExifInfo, 0x00, sizeof(ExifFileInfo));
makeExifParam(ExifInfo);


///////////////////////////////////////////////////////////
// 7. Encode YUV stream                            //
///////////////////////////////////////////////////////////
#if (FPS == 1)
        encodeTime = GetTickCount();
#endif

#if (TEST_ENCODE_WITH_EXIF == 1)
ret = SsbSipJPEGEncodeExe(handle, ExifInfo);    //with Exif
#else
ret = SsbSipJPEGEncodeExe(handle, NULL);                 //No Exif
#endif

#if (FPS == 1)
        encodeTime = GetTickCount() - encodeTime;
        printf( "encodeTime : %d \n", encodeTime);
#endif
if(ret != JPEG_OK){
        result = FALSE;
        break;
}

///////////////////////////////////////////////////////////
// 8. get output buffer address                    //
///////////////////////////////////////////////////////////
OutBuf = SsbSipJPEGGetEncodeOutBuf(handle, &frameSize);
if(OutBuf == NULL){
        result = FALSE;
        break;
}

printD("OutBuf : 0x%x freamsize : %d\n", OutBuf, frameSize);

///////////////////////////////////////////////////////////
// 9. write JPEG result file                       //
///////////////////////////////////////////////////////////
fp = fopen(outFilename, "wb");
fwrite(OutBuf, 1, frameSize, fp);
fclose(fp);

///////////////////////////////////////////////////////////
// 10. finalize handle                             //
///////////////////////////////////////////////////////////
SsbSipJPEGEncodeDeInit(handle);
free(ExifInfo);
Sleep(5);
}while(1);

if(result == FALSE){
        SsbSipJPEGEncodeDeInit(handle);
        if(ExifInfo != NULL)
                free(ExifInfo);
}
```

```
        fclose(CTRfp);
        printf("----------------------Encoder Test Done--------------------\n");
}
```