MJPEG AVI Library Developer's Guide

DEVIM1-002-D
October 2005

Analog Devices Inc.

www.analog.com

Table of Contents

1. Introduction 6
1.1. Scope 6
1.2. Target platform 6
1.3. Organisation of this Guide 6
1.4. Version Information 6
1.5. System Requirements 6
1.6. References 6
1.7. Additional Information 7

2. Specifications 7
2.1. Input formats 7
2.2. Output formats 7
2.3. MIPS Performance 7
2.4. Memory Requirements 7

3. Usage 8
3.1. Tools 8
3.2. Software Integration 8
3.2.1. Encoding 8
3.2.1.1. JPEG encoder library 8
3.2.1.2. MJPEG AVI encoder library 9
3.2.1.2.1. Temporary storage 9
3.2.2. Decoding 9
3.2.2.1. JPEG decoder library 9
3.2.2.2. MJPEG AVI decoding 9
3.3. Programming Example 9
3.3.1. MJPEG AVI Encoder 9
3.3.2. MJPEG AVI Decoder 11

4. Programmer’s Reference 12
4.1. MJPEG AVI Encoder Library 13
4.1.1. MJPEG_AVI_OpenFileWrite 13
4.1.2. MJPEG_AVI_CloseFileWrite 13
4.1.3. MJPEG_AVI_OpenStreamWrite 13
4.1.4. MJPEG_AVI_CloseStreamWrite 14
4.1.5. MJPEG_AVI_WriteStream 14
4.2. MJPEG AVI Decoder Library 14
4.2.1. MJPEG_AVI_OpenFileRead 14

4.2.2. MJPEG_AVI_CloseFileRead 15
4.2.3. MJPEG_AVI_OpenStreamRead 15
4.2.4. MJPEG_AVI_CloseStreamRead 16
4.2.5. MJPEG_AVI_RewindToFirstFrame 16
4.2.6. MJPEG_AVI_ReadNextFrame 17

List of Tables

Table 1: Tool versions 6
Table 2: Memory Requirements of MJPEG AVI libraries 8

1. Introduction

This document describes how the developer can use the MJPEG AVI Library in an application.

1.1. Scope

This guide is intended for C-language developers proficient in the application of the Analog Devices Blackfin family of processors and related software tools. It is assumed that the reader is familiar with all aspects of these processors.

1.2. Target platform

The MJPEG AVI library was designed for the Analog Devices Blackfin processors.

1.3. Organisation of this Guide

Information supplied in this guide is organised in the following way. Section 2 provides the specifications of the implemented MJPEG AVI library. In Section 3, notes on how to use and integrate the library are given. Finally, Section 3.3 describes the high-level software structure of the library.

1.4. Version Information

This document refers to the following software versions:
MJPEG AVI Library: ADI Version 4.0.0.

Any information that accompanies the software libraries (in the form of 'readme' files, etc.) should be assumed to supersede the information contained in this document.

1.5. System Requirements

The MJPEG AVI Encoder and Decoder library modules were created and tested with versions of tools listed in Table 1. To ensure the maximum level of compatibility, please use these versions of the tools (listed tools correspond to the September 2005 update for VDSP++ 4.0).

Tool Name	Tool Description	Tool Version
ccblkfn	C/C++ compiler	7.1.3.6
easmblkfn	BlackFin assembler	2.6.9.8
elfar	ELF Librarian/Archive Utility	4.5.1.3
linker	Linker	3.5.8.1

Table 1: Tool versions

1.6. References

[1] ITU-T, “Information Technology- Digital Compression and Coding of Continuous-Tone Still Images- Requirements and Guidelines”, Recommendation T.81 (09/92), September 1992.

[2] Analog Devices Inc., “JPEG Encoder Library Developer's Guide”, DEVIMG-002, October 2005.

[3] Analog Devices Inc., “JPEG Decoder Library Developer's Guide”, DEVIMG-001, October 2005.

1.7. Additional Information

For more information on the latest ADI processors, silicon errata, code examples, development tools, system services and devices drivers, technical support and any other additional information, please visit our website at www.analog.com/processors.

2. Specifications

The MJPEG AVI Library implements motion picture compression and decompression to/from the AVI format following standard [1] for individual image frames. The JPEG encoder library must be installed and working correctly before using the MJPEG AVI encoder functions. Likewise, the JPEG decoder library must be installed and working correctly before using the MJPEG AVI decoding functions. See the references for more information about the JPEG encoder and decoder libraries.

2.1. Input formats

Encoder: A single input format is used for an entire MJPEG AVI stream. This format can be one of YUV4:2:2, YUV4:2:0, or Y (YUV4:0:0, i.e. monochrome). In all formats, the components are not interleaved. Refer to [2] for more information about available JPEG input formats. Input to the MJPEG AVI encoder consists of concatenated input image frames.

Decoder: The MJPEG AVI Decoder takes as input an AVI stream file. All reading from the stream file occurs through the file system (i.e. calls to stdio.h) so a suitable file system is required for decoder operation.

2.2. Output formats

Encoder: The MJPEG AVI Encoder creates an AVI file, composed of individual JPEG-encoded frames. The sequential (baseline) JPEG mode is recommended, where each image component is encoded in a single left-to-right, top-to-bottom raster scan. All writing to the stream file occurs through the file system (i.e. calls to stdio.h) so a suitable file system is required for operation.

Decoder: The MJPEG AVI Decoder returns the bitstream corresponding to each frame in the specified memory buffer, which then needs to be decoded using a JPEG Decoder and subsequently the decoded frame may be written to a file to create a concatenated image file (raw video sequence), or sent to a video display.

2.3. MIPS Performance

A complete specification is not currently available. There are many factors that affect performance, some of which are data dependent. The MIPS overhead is low however, as there is little numerical processing performed in the MJPEG AVI module. The MIPS requirements for the JPEG Encoder and Decoder libraries are described in [2][3].

2.4. Memory Requirements

This section covers memory requirements for the MJPEG AVI library. The memory requirements for the JPEG Encoder and Decoder libraries are described elsewhere [2][3].

The memory requirements and memory section names for the MJPEG libraries are shown in Table 2.

Encoding Mode	Program Memory (kB)	Data Memory (kB)	Heap Maximum (kB)
Encoder	(MJPEGENC_P0) 1.2	0.4	0.2+0.016N
Decoder	(MJPEGDEC_P0) 2.1	0.1	0.2

Table 2: Memory Requirements of MJPEG AVI libraries

Here, N represents the number of frames written to the AVI file.
Data memory section names MJPEGENC_D0 and MJPEGDEC_D0, data1, constdata and bsz are used by the MJPEG AVI libraries.

3. Usage

3.1. Tools

The VisualDSP++ 4.0 integrated development environment for Blackfin is used to build the library and example applications on a Windows PC. The required versions of the tools in this environment are stated in Section 1.4 of this document.

3.2. Software Integration

This section details how to integrate the MJPEG functionality into a test application ("the application").

3.2.1. Encoding

3.2.1.1. JPEG encoder library

The application must link against the JPEG encoder library, so the library file **`jpeg_enc_lib_XXXX.dlb`** must be included in the application makefile (where **XXXX** identifies the library version). If the VisualDSP graphical interface is used, this file can be added to the project under Project Options→Link→Additional Options by:

- typing in the full filename of the library,
- specifying the path that points to this library under Project Options→Link→Search Directories, and
- specifying the path that points to the library header files under Project Options→Compile→Preprocessor→Additional Include Directories.

Alternatively, one can drag the library file from an open Windows Explorer window to the Project Window in VisualDSP++.

The following JPEG header files must be #included in the application:
`jpeg_api_encoder.h`
and the JPEG export directory must be specified as an included in the application project.

3.2.1.2. MJPEG AVI encoder library

The application must link against the MJPEG AVI encoder library file **mjpeg_enc_lib_XXXX.dlb**, following a similar procedure to that in Section 3.2.1.1 above.

The following header files must be #included into the application:
MJPEG_AVI_FileWriter.h
and the MJPEG AVI export directory must be specified as an included in the application project.

The MJPEG AVI Encoder is dependent on the JPEG Encoder library. JPEG header file JPEG_api_common.h is #included by MJPEG_AVI_FileWriter.h.

3.2.1.2.1. Temporary storage

The MJPEG AVI Encoder library creates a temporary file in the current working directory, called **AVIWRITE_TEMP.tmp.** This file may be deleted after the encoding operation concludes.

3.2.2. Decoding

3.2.2.1. JPEG decoder library

The application must link against the JPEG decoder library, so the library file **jpeg_dec_lib_XXXX.dlb** must be included in the application makefile, following a similar procedure to that in Section 3.2.1.1 above.

The following JPEG header files must be #included in the application:
jpeg_api_decoder.h
and the JPEG export directory must be specified as an included in the application project.

3.2.2.2. MJPEG AVI decoding

The application must link against the MJPEG AVI decoder library file **mjpeg_dec_lib_XXXX.dlb**, following a similar procedure to that in Section 3.2.1.1 above.
The following header files must be #included into the application:
MJPEG_AVI_FileReader.h
and the MJPEG export directory must be specified as an included in the application project.

The MJPEG AVI Decoder is dependent on the JPEG Decoder library. JPEG header file JPEG_api_common.h is #included by MJPEG_AVI_FileReader.h.

3.3. Programming Example

3.3.1. MJPEG AVI Encoder

The simplest method to encode an MJPEG AVI motion picture is to call the library functions in the following order:

i. Create an instance of the JPEG encoder. See Reference [2] for information about instantiating the JPEG encoder.
ii. Call **MJPEG_AVI_OpenFileWrite** to create the output AVI file
iii. Call **MJPEG_AVI_OpenStreamWrite** to create the output stream
iv. While input data is available, call **JPEG_EncodeImage** to encode each frame, and write the coded frame to the output file by calling **MJPEG_AVI_WriteStream**

v. Close the stream and file with **MJPEG_AVI_CloseStreamWrite** and **MJPEG_AVI_CloseFileWrite** respectively

The following code example uses these functions to encode a sample MJPEG AVI input stream. See the respective entries in the JPEG Encoder Programmer's Reference [2] for information about the JPEG functions called here.

```
/***************************/
/* SETUP JPEG ENCODER         */
/***************************/
/**/
/* Configure the JPEG Encoder to suit input frames and output parameters        */
JPEG_Param_CONFIG(&lImageParam, JPEG_MAX_X_DIMENSION, lMaxXDimn);
JPEG_Param_CONFIG(&lImageParam, JPEG_MAX_Y_DIMENSION, lMaxYDimn);
JPEG_Param_CONFIG(&lImageParam, JPEG_IMAGEFORMAT, lInterLeaveFormat);
JPEG_Param_CONFIG(&lImageParam, JPEG_QUALITYFACTOR, lQualityFactor);
JPEG_Param_CONFIG(&lImageParam, JPEG_ENCODINGMODE, SEQUENTIAL);
JPEG_Param_CONFIG(&lImageParam, JPEG_THRESHOLD, 0);

/* Allocate Input and Output Buffers         */
JPEG_Param_CONFIG(&lImageParam, JPEG_POINTER_INPUT, (int)lFrameBuffer);
JPEG_Param_CONFIG(&lImageParam, JPEG_POINTER_OUTPUT, (int)lStreamBuffer);

/**/
/* Create Instance of JPEG Encoder */
lJpegEnc = JPEG_Encoder_NEW(&lImageParam);


/***************************/
/* SETUP MJPEG ENCODER        */
/***************************/
/* Create the AVI output file, with the required file */
lResult = MJPEG_AVI_OpenFileWrite(&lStreamFileOutHandle, lStreamFileOutName, &lImageParam, FrameRate);

/**/
/* Create the stream */
lResult = MJPEG_AVI_OpenStreamWrite(lStreamFileOutHandle, &lStreamHandle, &lStreamInfo);

/***************************/
/* ENCODE EACH FRAME          */
/***************************/

/**/
/* Main loop for each frame */
fprintf(TestReportFile, "Main loop\n");

/**/
/* ... Read frame from file into lFrameBuffer here ...*/
while (isInputDataAvailable)
{
        /**/
        /* Encode the image frame */
        lResult = JPEG_EncodeImage(lJpegEnc, &NumBytes);

        /* Write the encoded frame to the stream */
        lResult = MJPEG_AVI_WriteStream(lStreamHandle, lStreamBuffer, NumBytes, 0);

        /**/
        /* ... Read the next frame from file into lFrameBuffer here ...*/

} /* end while main loop */

/*********************************/
/* FINALISE AND DESTROY ENCODER             */
/*********************************/

/**/
/* Close the AVI stream*/
lResult = MJPEG_AVI_CloseStreamWrite(lStreamHandle);

/**/
/* Close the output file*/
lResult = MJPEG_AVI_CloseFileWrite(lStreamFileOutHandle);
```

```
/**/
/* Free and destroy used resources */
JPEG_Encoder_DELETE(lJpegEnc);
```

3.3.2. MJPEG AVI Decoder

The simplest method to decode an MJPEG AVI stream is to call the library functions in the following order:

i. Call **`MJPEG_AVI_OpenFileRead`** to open the input AVI file

ii. Call **`MJPEG_AVI_OpenStreamRead`** to open the input stream and to read the stream header

iii. Call **`MJPEG_AVI_RewindToFirstFrame`** to move file pointer to the start of the first frame

iv. Call **`JPEG_MemAlloc_NEW`** to allocate JPEG decoder input buffer

v. Call **`MJPEG_AVI_ReadNextFrame`** to read the first frame

vi. Call **`JPEG_Param_CONFIG`** to configure JPEG_POINTER_INPUT, JPEG_POINTER_OUTPUT

vii. Call **`JPEG_Decoder_NEW`** to create decoder instance. See Reference [3] for information about instantiating the JPEG decoder. If all JPEG parameters are known, then the JPEG Decoder can be set up directly. Otherwise, if decoding an unknown MJPEG AVI, then the application needs to use information contained within the stream (see next step).

viii. Call **`JPEG_ProcessHeader`** to process the first frame of JPEG header

ix. Call **`MJPEG_AVI_RewindToFirstFrame`** to move file pointer to the start of the first frame again

x. For each frame in the MJPEG AVI stream, call **`MJPEG_AVI_ReadNextFrame`** to retrieve the image frame from the stream before calling **`JPEG_DecodeImage`** to decode it.

xi. Call **`MJPEG_AVI_CloseStreamRead`** and **`MJPEG_AVI_CloseFileRead`** to close the AVI stream and input file, respectively

xii. Destroy the instance of the JPEG decoder. See Reference [3] for information about destroying the JPEG decoder

The following code example uses these functions to decode an MJPEG AVI stream.

```
/***************************/
/* SETUP MJPEG DECODER        */
/***************************/
/**/
/* Read the JPEG information from the first frame of the previously encoded stream */
JPEG_Param_INIT(&lImageParam);
lResult = MJPEG_AVI_GetImageInfo(lStreamFileInName, &lImageParam, &lMinStreamBufLength, TestReportFile);


/**/
/* Open the AVI file */
lResult = MJPEG_AVI_OpenFileRead(&lStreamFileInHandle, (int8 *)lStreamFileInName);

/**/
/* Get the handle for AVI stream */
lResult = MJPEG_AVI_OpenStreamRead (lStreamFileInHandle, &lStreamHandle,
                                          MJPEG_AVI_StreamTypeVIDEO, 0);

lResult = MJPEG_AVI_RewindToFirstFrame(lStreamHandle, &lStreamBufferLength);
StreamBuffer_Obj = JPEG_MemAlloc_NEW(lStreamBufferLength,1,MEM_TYPE_DATA);
lStreamBuffer = (uint8*)JPEG_MemAlloc_ADDRESS(StreamBuffer_Obj);
lResult = MJPEG_AVI_ReadNextFrame(lStreamHandle, lStreamBuffer, &lMinStreamBufLength);


/***************************/
/* SETUP JPEG DECODER         */
/***************************/
/**/
/* ... Create the input stream buffer and output frame buffer here ... */
lStreamBufferInfo.Length = lStreamBufferLength;
lStreamBufferInfo.Pointer = lStreamBuffer;
JPEG_Param_CONFIG(&lImageParam, JPEG_POINTER_INPUT, (int)(&lStreamBufferInfo));
JPEG_Param_CONFIG(&lImageParam, JPEG_POINTER_OUTPUT, (int)lFrameBuffer);
```

```
/**/
/* Creating the JPEG decoder instance */
lJpegDec = JPEG_Decoder_NEW(&lImageParam);
lResult = JPEG_ProcessHeader(lJpegDec, &lImageParam);
lResult = MJPEG_AVI_RewindToFirstFrame(lStreamHandle, &lStreamBufferLength);
/***************************/
/* DECODE EACH FRAME          */
/***************************/
isInputDataAvailable = E_TRUE;
while (E_TRUE == isInputDataAvailable) {
        /* Read the frame */
        lResult = MJPEG_AVI_ReadNextFrame(lStreamHandle, lStreamBuffer, lStreamBufferLength);
        if(lResult != MJPEG_AVI_RETURN_OK)
        {
                if (lResult == MJPEG_AVI_RETURN_NOMORESAMPLES)
                {
                        fprintf(TestReportFile, "\n*** End of MJPEG stream ***\n");
                        break;
                }
                else
                {
                        fprintf(TestReportFile, "Cannot read next frame of the MJPEG stream\n");
                        exit (EXITERROR);
                }
        }
        /* Decode image */
        lResult = JPEG_DecodeImage(lJpegDec, &lImageParam);
        /* ... Write the decoded image from the frame buffer here ...*/
}; // For each frame

/*********************************/
/* FINALISE AND DESTROY DECODER              */
/*********************************/

/**/
/* Close the AVI stream*/
lResult = MJPEG_AVI_CloseStreamRead(lStreamHandle);

/**/
/* Close the output file*/
lResult = MJPEG_AVI_CloseFileRead(lStreamFileInHandle);

/**/
/* Free and destroy used resources */
JPEG_Decoder_DELETE(lJpegDec);
```

4. Programmer’s Reference

This section uses two examples to show each of the functions in the MJPEG AVI library. A reference for each function is also provided.

The function arguments are described in the Parameters sections as [IN] or [OUT]. This defines whether the function will read (IN pointers) or write to the memory (OUT pointers).

4.1. MJPEG AVI Encoder Library

This section describes MJPEG AVI API functions required for encoding operations.

4.1.1. MJPEG_AVI_OpenFileWrite

Function Name:

```
int32 MJPEG_AVI_OpenFileWrite(uint32 *pAVIFileHandle, uint8 *filename,
                              tJpegParam *ImageParam, uint32 FrameRate)
```

Parameters:

pAVIFileHandle [OUT]	Pointer to the output file handle
fileName [IN]	Pointer to the null-terminated filename
ImageParam [IN]	Pointer to JPEG image parameters.
FrameRate [IN]	Frame rate (frames/second)

Return Value:

MJPEG_AVI_RETURN_OK	if file opened OK
MJPEG_AVI_RETURN_ERROR	otherwise

Description:

Opens an MJPEG AVI file output file for encoding and create intermediate AVI header.

4.1.2. MJPEG_AVI_CloseFileWrite

Function Name:

```
int32 MJPEG_AVI_CloseFileWrite(uint32 AVIFileHandle)
```

Parameters:

AVIFileHandle [IN]	MJPEG AVI output file handle

Return Value:

MJPEG_AVI_RETURN_OK	if file opened OK
MJPEG_AVI_RETURN_ERROR	otherwise

Description:

Append index table, update AVI header, and close the MJPEG AVI output file.

4.1.3. MJPEG_AVI_OpenStreamWrite

Function Name:

```
int32 MJPEG_AVI_OpenStreamWrite(    uint32 AVIFileHandle,
                                    uint32 *pAVIStreamHandle,
                                    tMJPEG_AVI_STREAMINFO *pStreamInfo)
```

Parameters:

AVIFileHandle [IN]	MJPEG AVI file handle
pAVIStreamHandle [OUT]	Pointer to stream handle
pStreamInfo [IN]	Pointer to stream information

Return Value:

MJPEG_AVI_RETURN_OK	if file opened OK
MJPEG_AVI_RETURN_ERROR	otherwise

Description:

Creates and opens an MJPEG AVI output stream

4.1.4. MJPEG_AVI_CloseStreamWrite

Function Name:
```
int32 MJPEG_AVI_CloseStreamWrite(uint32 AVIStreamHandle)
```
Parameters:

AVIStreamHandle [IN]	MJPEG AVI stream handle

Return Value:

MJPEG_AVI_RETURN_OK	if stream closed OK
MJPEG_AVI_RETURN_ERROR	otherwise

Description:
Closes and destroys an MJPEG AVI output stream.

4.1.5. MJPEG_AVI_WriteStream

Function Name:
```
int32 MJPEG_AVI_WriteStream(  uint32 AVIStreamHandle,
                              uint8 *pDataBuffer,
                              uint32 dataLength,
                              uint8 IsKeyFrame)
```
Parameters:

AVIStreamHandle [IN]	MJPEG AVI stream handle
pDataBuffer [IN]	Pointer to data buffer.
dataLength [IN]	Length of data buffer
IsKeyFrame [IN]	Sets the key frame indicator flag at the current location if equal to 1

Return Value:

MJPEG_AVI_RETURN_OK	if data written OK
MJPEG_AVI_RETURN_ERROR	otherwise
MJPEG_AVI_RETURN_EVALUATIONLIMITREACHED	if an evaluation package has reached its limit

Description:
Write the given data to the MJPEG AVI stream. If IsKeyFrame = 1, a keyframe is flagged in the bitstream at the current position.

4.2. MJPEG AVI Decoder Library

This section describes MJPEG AVI API functions required for AVI decoding operations using the JPEG decoder library.

4.2.1. MJPEG_AVI_OpenFileRead

Function Name:
```
int32 MJPEG_AVI_OpenFileRead(uint32 *pAVIFileHandle, int8 *fileName)
```
Parameters:

pAVIFileHandle [OUT]	Pointer to receive the AVI File Handle
fileName [IN]	Name of the AVI file to be opened

Return Value:

MJPEG_AVI_RETURN_OK	if file opened OK
MJPEG_AVI_RETURN_OUTOFMEMORY	if memory could not be allocated when opening the file
MJPEG_AVI_RETURN_FILEOPENFAIL	if the input file could not be opened
MJPEG_AVI_RETURN_FILEREADFAIL	if the input file could not be read
MJPEG_AVI_RETURN_FILEINCORRECTFORMAT	if the input file was not in MJPEG AVI format

MJPEG_AVI_RETURN_FILESEEKFAIL
if the seek operation on the input file failed
MJPEG_AVI_RETURN_ERROR otherwise

Description:

Opens the MJPEG AVI file for reading.
Assumptions:

1. The index table (i.e. idx1 chunk) is assumed to be present in the file.

4.2.2. MJPEG_AVI_CloseFileRead

Function Name:

```
int32 MJPEG_AVI_CloseFileRead(uint32 AVIFileHandle)
```

Parameters:

AVIFileHandle [IN]	AVI File Handle

Return Value:

MJPEG_AVI_RETURN_OK	if file closed OK
MJPEG_AVI_RETURN_ERROR	otherwise

Description:

Closes the AVI file.

4.2.3. MJPEG_AVI_OpenStreamRead

Function Name:

```
int32 MJPEG_AVI_OpenStreamRead(     uint32 AVIFileHandle,
                                    uint32 *pAVIStreamHandle,
                                    uint32 fccType,
                                    int32 index)
```

Parameters:

AVIFileHandle [IN]	Handle of input AVI file
pAVIStreamHandle [OUT]	Pointer to receive the stream handle after the stream is opened
fccType [IN]	FCC file type.
index [IN]	Index of the stream to be opened.

Return Value:

MJPEG_AVI_RETURN_OK	if file opened OK
MJPEG_AVI_RETURN_OUTOFMEMORY	if memory could not be allocated when opening the file
MJPEG_AVI_RETURN_FILEOPENFAIL	if the input file could not be opened
MJPEG_AVI_RETURN_FILEREADFAIL	if the input file could not be read
MJPEG_AVI_RETURN_FILEINCORRECTFORMAT	if the input file was not in MJPEG AVI format
MJPEG_AVI_RETURN_FILESEEKFAIL	if the seek operation on the input file failed
MJPEG_AVI_RETURN_ERROR	otherwise

Description:

Assigns a stream handle to the AVI file.
Limitations :

1. index is supported only for value 0.
2. fccTypeParam is supported for values "MJPEG_AVI_StreamTypeVIDEO" and "MJPEG_AVI_StreamTypeAUDIO"

Assumptions :

1. The strl chunk index in the avih header is same as the corresponding stream's stream identifier

4.2.4. MJPEG_AVI_CloseStreamRead

Function Name:
`int32 MJPEG_AVI_CloseStreamRead(uint32 AVIStreamHandle)`
Parameters:
AVIStreamHandle [IN] Handle of the AVI Stream Handle to be closed.
Return Value:
MJPEG_AVI_RETURN_OK if stream closed OK
MJPEG_AVI_RETURN_ERROR otherwise
Description:
Closes the AVI stream

4.2.5. MJPEG_AVI_RewindToFirstFrame

Function Name:
`int32 MJPEG_AVI_RewindToFirstFrame(uint32 AVIStreamHandle, uint32 *pBufLength)`
Parameters:
AVIStreamHandle [IN] Handle of input AVI stream
pBufLength [OUT] Pointer to receive the size of the next frame (in bytes)

Return Value:
MJPEG_AVI_RETURN_OK if file opened OK
MJPEG_AVI_RETURN_OUTOFMEMORY
if memory could not be allocated when opening the file
MJPEG_AVI_RETURN_FILEOPENFAIL
if the input file could not be opened
MJPEG_AVI_RETURN_FILEREADFAIL
if the input file could not be read
MJPEG_AVI_RETURN_FILEINCORRECTFORMAT
if the input file was not in MJPEG AVI format
MJPEG_AVI_RETURN_FILESEEKFAIL
if the seek operation on the input file failed
MJPEG_AVI_RETURN_NOMORESAMPLES
if no more frames are available to be read
MJPEG_AVI_RETURN_EVALUATIONLIMITREACHED
if an evaluation package has reached its limit
MJPEG_AVI_RETURN_ERROR otherwise
Description:
Rewind the AVI stream up to the start of the JPEG image of the first frame. Returns the maximum number of bytes in al the subsequent JPEG images.

4.2.6. MJPEG_AVI_ReadNextFrame

Function Name:

```
int32 MJPEG_AVI_ReadNextFrame(uint32 AVIStreamHandle,
                              void* pBuffer,
                              uint32 *pBufLength)
```

Parameters:

AVIStreamHandle [IN]	Handle of input AVI stream
pBuffer [OUT]	Pointer to the frame buffer
pBufLength [OUT]	The length of the next frame

Return Value:

MJPEG_AVI_RETURN_OK	if file opened OK
MJPEG_AVI_RETURN_STREAMINVALID	if the supplied stream handle is invalid
MJPEG_AVI_RETURN_FILEREADFAIL	if the input file could not be read
MJPEG_AVI_RETURN_ERROR	otherwise

Description:

Reads the next image frame into the supplied buffer.