



Advanced Micro Peripherals Ltd

MPEG4000

Software Reference Manual

Document Version 0.18

16th December 2005

This document and the accompanying software, if any, are supplied on an as is basis. AMP does not warrant its suitability for any specific use

Introduction

This is the API reference for Advanced Micro Peripherals Ltd MPEG4000 MPEG4 hardware encoder card. The API is the same for both Windows and Linux. Functions that are specific to an OS are marked as such.

Windows Library files

The application should be linked against the MPG4K.LIB library file. This file is found in *software\lib*. The MPG4K.DLL file should be placed in the System directory on the target machine. Under Windows NT 4 and Windows 2000 this is *\winnt\system32*. Under Windows XP this is *\windows\system32*. For preview under Windows, the current version of the library requires DirectX support. A version with preview disabled and no dependency on DirectX is available on request.

Linux Library files

The application should be linked against the libmpeg4klib.a library file. This file is found in *software\lib*. For preview under Linux, the current version of the library requires SDL. This is available from <http://www.libsdl.org>. A version with preview disabled and no dependency on SDL is available on request.

Firmware files

The firmware file is *mpeg4k.bin* and can be found in the *software\lib* directory. This file should be copied into the working directory of the target application. This is normally the directory the application is run from.

Preview

The MPEG4000-4 has a preview feature where the video being encoded can be viewed on screen. Under Windows this uses DirectX surfaces. Windows requires little intervention for the preview to display. Under Linux this currently uses SDL. Linux requires that availability of the preview frame is polled for before displaying the preview.

Overview of writing applications

Writing an application to capture MPEG-4 video can be done very easily using the functions documented here.

Include files

The MPEG4K.H and CMPEG4DEFS.H header files must be included in all applications using the MPEG4000 API.

Initialisation of the MPEG4000 hardware

The MPEG4000 board is initialised using the [MPG4K_Init](#) or [MPG4K_InitCard](#) function. This must be called before any other function from the MPEG4000 library is called. Once this has been done, the MPEG4000 can be configured using the following functions.

Function list

- [MPG4K_AddAudioCallback](#)
- [MPG4K_AddVideoCallback](#)
- [MPG4K_ClearOSD](#)

MPG4K_Configure
MPG4K_DrawPreviewFrame
MPG4K_EnableChannel
MPG4K_EnablePreview
MPG4K_FlushAVIBuffer
MPG4K_GetBrightness
MPG4K_GetContrast
MPG4K_GetFrameCount
MPG4K_GetHorizontalStart
MPG4K_GetHue
MPG4K_GetSaturationU
MPG4K_GetSaturationV
MPG4K_GetVerticalStart
MPG4K_Init
MPG4K_InitCard
MPG4K_Pause
MPG4K_PreviewFrameReady
MPG4K_PrintOSD
MPG4K_SelectInput
MPG4K_SetAudioPath
MPG4K_SetAVIBufferSize
MPG4K_SetAVIFilename
MPG4K_SetBitrate
MPG4K_SetBitrateMin
MPG4K_SetBrightness
MPG4K_SetContrast
MPG4K_SetEncodingFormat
MPG4K_SetEncodingMode
MPG4K_SetEncodingType
MPG4K_SetFramePattern
MPG4K_SetFrameRate
MPG4K_SetHorizontalStart
MPG4K_SetHue
MPG4K_SetIInterval
MPG4K_SetInputPath
MPG4K_SetInputStandard
MPG4K_SetM4VFilename
MPG4K_SetMode
MPG4K_SetMP4Filename
MPG4K_SetOSDColour
MPG4K_SetOutputType
MPG4K_SetPIPPosition
MPG4K_SetPreviewColourKey
MPG4K_SetPreviewLocation
MPG4K_SetPreviewMode
MPG4K_SetPreviewInput
MPG4K_SetQuality
MPG4K_SetSaturationU
MPG4K_SetSaturationV
MPG4K_SetSingleFieldMode
MPG4K_SetVerticalStart
MPG4K_SetVideoFOURCC
MPG4K_SilenceAudio
MPG4K_Start
MPG4K_StartPreview
MPG4K_Stop
MPG4K_StopEncoder
MPG4K_StopPreview
MPG4K_WriteOSD

Function Documentation

int MPG4K_Init(void)

MPG4K_Init Initialise all MPEG4000s for use

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_NOFWFILE</i>	The firmware file was not found
<i>ID_ERR_MEMALLOC</i>	Could not allocate memory for the firmware
<i>ID_ERR_FWUPLOAD</i>	Error uploading the firmware

Remarks:

int MPG4K_InitCard(int *nCard*)

MPG4K_InitCard Initialises the specified MPEG4000 for use

Parameters:

int nCard - Specifies which MPEG4000 to initialise. Valid values are 0 to 3

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_NOFWFILE</i>	The firmware file was not found
<i>ID_ERR_MEMALLOC</i>	Could not allocate memory for the firmware
<i>ID_ERR_FWUPLOAD</i>	Error uploading the firmware

Remarks:

The MPEG4000 SDK currently supports 5 MPEG4000

```

int MPG4K_AddAudioCallback( int  nCard,
                           int  nChannel,
                           void  (*Callback)(void * lpContext, unsigned char *bpData, int
                           nDataSize, int nChannel, BOOL isKeyFrame, int nFrameCnt),
                           void  lpContext
                           *
                           )

```

MPG4K_AddAudioCallback Adds a callback to receive mu-Law encoded audio data for the specified channel

Parameters:

int nCard - Specifies which MPEG4000 to control
int nChannel - Specifies the channel to receive audio data for.
void (*Callback)(void * lpContext, unsigned char *bpData, int nDataSize, int nChannel, BOOL isKeyFrame, int nFrameCnt) - Specifies the callback function that is called with the data.
void * lpContext - Specifies the context that is passed as the first parameter to the callback function.

Returns:

int

Return values:

ID_ERR_INVALID_CHANNEL Invalid channel number specified
ID_ERR_NODEVICE No MPEG4000 was detected
Callback ID number This should be used to remove the callback

Remarks:

Minimal processing should be done in the callback function.
The callback function is called once for each packet of audio.
The bpData contains one packet of 8bit per sample, mono, 8000Hz, mu-Law encoded audio. Do not modify the data pointed to by this variable.

```

int MPG4K_AddVideoCallback( int  nCard,
                           int  nChannel,
                           void  (*Callback)(void * IpContext, unsigned char *bpData, int
                           nDataSize, int nChannel, BOOL isKeyFrame, int nFrameCnt),
                           void  * IpContext
                           )

```

MPG4K_AddVideoCallback Adds a callback to receive the compressed video data for the specified channel

Parameters:

int nCard - Specifies which MPEG4000 to control
int nChannel - Specifies the channel to receive video data for.
void (*Callback)(void * IpContext, unsigned char *bpData, int nDataSize, int nChannel,BOOL isKeyFrame, int nFrameCnt) - Specifies the callback function that is called with the data.
void * IpContext - Specifies the context that is passed as the first parameter to the callback function.

Returns:

int

Return values:

ID_ERR_INVALID_CHANNEL Invalid channel number specified
ID_ERR_NODEVICE No MPEG4000 was detected
Callback ID number This should be used to remove the callback

Remarks:

Processing in the callback function should not take very long since other callbacks may be waiting to be called. The callback function should not block.
The callback function is called once for each frame of video.
The bpData contains one frame of compressed video. This data complies with ISO 14496-2. Do not modify the data pointed to by this variable.

```
int MPG4K_ClearOSD( int nCard )
```

MPG4K_ClearOSD clears the On Screen Display

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified

Remarks:

This function only works with a MPEG4000-4.

```
int MPG4K_Configure( int nCard )
```

MPG4K_Configure configures the MPEG4000 into a paused state.

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified

Remarks:

This function can be used to configure the MPEG4000 into the same state as MPG4K_Pause without having to call MPG4K_Start first.

This can be used to decrease the startup time.

```
int MPG4K_DrawPreviewFrame( int nCard )
```

MPG4K_DrawPreviewFrame draws the preview frame.

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified

Remarks:

This function is Linux only

```
int MPG4K_EnableChannel( int nCard,  
                        int nChannel,  
                        int nEnable  
                        )
```

MPG4K_EnableChannel Enables/disables the specified channel for use

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies the channel to enable/disable

int nEnable - Specifies whether to enable or disable the channel.

- ID_ENABLE
- ID_DISABLE

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card or channel specified
ID_ERR_ENCODING	The encoder is running

Remarks:

This functions enable or disables the filewriting for the specified file.

Disabling a channel does not close the output file. To do this call MPG4K_SetXXXFilename, where XXX is the file type, with a dummy NULL filename. The exception to this is M4V files which do not require the dummy call.

```
int MPG4K_EnablePreview( int nCard,  
                        int nEnable  
                        )
```

MPG4K_EnablePreview enables the preview window

Parameters:

int nCard - Specifies which MPEG4000 to control

int nEnable - Specifies whether to enable or disable the preview window.

- ID_ENABLE
- ID_DISABLE

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card or channel specified
ID_ERR_PREVIEW	Preview not running
ID_ERR_NO_PREVIEW	No preview hardware available

Remarks:

This function should be called after MPG4K_StartPreview to show or hide the preview.

```
int MPG4K_FlushAVIBuffer( int nCard,  
                           int nChannel,  
                           )
```

MPG4K_FlushAVIBuffer Flushes the data held in the AVI buffer to disk.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies the channel to flush the AVI buffer

Returns:

int

Return values:

ID_OK Success

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid card or channel specified

Remarks:

This function can be used to force a buffer to be flushed to the disk.

```
int MPG4K_GetBrightness( int nCard,  
                        int nChannel  
                        )
```

MPG4K_GetBrightness Get the current brightness of the specified card and channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel the brightness should be retrieved from

Returns:

int - The brightness setting

Remarks:

```
int MPG4K_GetContrast( int nCard,  
                      int nChannel  
                      )
```

MPG4K_GetContrast Get the current contrast of the specified card and channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel the contrast should be retrieved from

Returns:

int - The contrast setting

Remarks:

```
int MPG4K_GetFrameCount( int nCard)
```

MPG4K_GetFrameCount Get the current count of frames encoded.

Parameters:

int nCard - Specifies which MPEG4000 to get the frame count for

Returns:

unsigned short - The frame count

Remarks:

In quad mode, the count includes frames from all channels.

The count includes frames that have been encoded but not passed to the video callbacks.

The count can be used to synchronise external events or data with the compressed video file.

The count will wrap around after 65535 frames. If the count is to be used for synchronisation, it is advisable to periodically check for this.

```
int MPG4K_GetHorizontalStart( int nCard,  
                             int nChannel  
                             )
```

MPG4K_GetHorizontalStart Get the current horizontal start for the specified channel.

Parameters:

int nCard - Specifies which MPEG4000 to to control

int nCard - Specifies which channel to get the horizontal start from

Returns:

int - The horizontal start

Remarks:

```
int MPG4K_GetHue( int nCard,  
                  int nChannel  
                  )
```

MPG4K_GetHue Get the current hue of the specified card and channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel the hue should be retrieved from

Returns:

int - The hue setting

Remarks:

```
int MPG4K_GetSaturationU( int nCard,  
                           int nChannel  
                           )
```

MPG4K_GetSaturationU Get the current saturation of the specified card and channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel the saturation should be retrieved from

Returns:

int - The saturation setting

Remarks:

```
int MPG4K_GetSaturationV( int nCard,  
                           int nChannel  
                           )
```

MPG4K_GetSaturationV Get the current saturation of the specified card and channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel the saturation should be retrieved from

Returns:

int - The saturation setting

Remarks:

```
int MPG4K_GetVerticalStart( int nCard,  
                           int nChannel  
                           )
```

MPG4K_GetVerticalStart Get the current vertical start for the specified channel.

Parameters:

int nCard - Specifies which MPEG4000 to to control

int nCard - Specifies which channel to get the vertical start from

Returns:

int - The vertical start

Remarks:

```
int MPG4K_Pause( int nCard,  
                 int nPause  
                 )
```

MPG4K_Pause pauses or unpauses the recording

Parameters:

int nCard - Specifies which MPEG4000 to control

int nPause - Specifies whether to pause the recording.

- ID_NOPAUSE
- ID_PAUSE

Returns:

int

Return values:

ID_OK Success

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid card specified

Remarks:

This function has no effect on the preview.

Due to buffering, several frames may be recorded after this function is called to pause the recording.

The first few frames after unpausing may have been recorded before the recording was paused.

```
int MPG4K_PreviewFrameReady( int nCard )
```

MPG4K_PreviewFrameReady Polls whether a frame is ready for preview.

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>0</i>	No preview data is waiting for display
<i>1</i>	Preview data is waiting for display
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid channel number specified

Remarks:

This function is Linux only.

If this function returns 1, [MPG4K_DrawPreviewFrame](#) should be called to display the frame.

```

int MPG4K_PrintOSD( int      nCard,
                    int      nX
                    int      nY
                    char *    szString
                    unsigned long ulFlags
                    )

```

MPG4K_PrintOSD writes the specified string to the On Screen Display using the current OSD colour

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nX* - Specifies the horizontal location to start the string
int *nY* - Specifies the vertical location to start the string
*char ** *szString* - Specifies the string to output
unsigned long *ulFlags* - Specifies which output to print the string to

- ID_CAPTURE - Write the string to the capture
- ID_PREVIEW - Write the string to the preview

Returns:

int

Return values:

ID_OK Success
ID_ERR_NODEVICE No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL Invalid card or channel specified

Remarks:

The locations are in units of characters. The display is 28 characters wide and 15 characters high
This function only works with a MPEG4000-4.

```
int MPG4K_SelectInput( int nCard,  
                      int nChannel,  
                      int nInput  
                      )
```

MPG4K_SelectInput

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel to change the input

int nInput - Specifies the input to the video decoder. Valid values are 0, 1, 2 and 3

Returns:

int

Return values:

ID_OK Successful

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid channel number specified

Remarks:

```
int MPG4K_SetAudioPath( int nCard,  
                        int nChannel,  
                        int nInput  
                        )
```

MPG4K_SetAudioPath routes the audio from the inputs to the specified channels callback chain.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel to change the input

int nInput - Specifies the input to the MPEG4000. Valid values are 0, 1, 2 and 3

Returns:

int

Return values:

ID_OK Successful

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid channel number specified

Remarks:

This function is similar to MPG4K_SetInputPath but for audio.

Audio can be duplicated between files by specifying the same input for multiple channels

The specified channel will not receive audio unless the channel has been enabled using MPG4K_EnableChannel and an I-frame has been received. This means multiple audio inputs can only be recorded simultaneously in ID_QUAD_MODE.

Input 0 corresponds to Audio-A

Input 1 corresponds to Audio-B

Input 2 corresponds to Audio-C

Input 3 corresponds to Audio-D

```
int MPG4K_SetAVIBufferSize( int      nCard,
                             int      nChannel,
                             unsigned long ulBufferSize
                             )
```

MPG4K_SetAVIBufferSize Sets the buffer size for the AVI file in memory.

Parameters:

<i>int</i>	nCard - Specifies which MPEG4000 to control
<i>int</i>	nChannel - Specifies which channel the buffer size should be set for
<i>unsigned long</i>	ulBufferSize - Specifies the buffer size (in bytes) to use

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified

Remarks:

This function cannot be used once MPG4K_Start has been called.

```
int MPG4K_SetAVIFilename( int    nCard,  
                           int    nChannel,  
                           char * szFilename  
                           )
```

MPG4K_SetAVIFilename Sets the filename for the AVI output

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel the filename should be set for
*char * szFilename* - Specifies the filename to use

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified

Remarks:

If the encoder is running when this function is called, the old file will be closed and all subsequent data will be saved to the new filename.
To close the file without stopping the encoder, call this function with *szFilename*=NULL

```
int MPG4K_SetBitrate( int      nCard,
                     int      nChannel,
                     unsigned long ulBitrate
                     )
```

MPG4K_SetBitrate Sets the maximum bitrate for CBR and HBR mode

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel to set the maximum bitrate for
unsigned long *ulBitrate* - Specifies the bitrate. Units of bits per second

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified
<i>ID_ERR_INVALID_MODE</i>	Not in CBR or HBR mode

Remarks:

For single channel full framerate, the bitrate should be between 500,000 and 10,000,000
 For quad mode full framerate, the bitrate should be between 100,000 and 5,000,000

```
int MPG4K_SetBitrateMin( int      nCard,  
                        int      nChannel,  
                        unsigned long ulBitrate  
                        )
```

MPG4K_SetBitrateMin Sets the minimum bitrate for HBR mode

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies the channel to set the minimum bitrate for
unsigned long *ulBitrate* - Specifies the minimum bitrate. Units of bits per second

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified
<i>ID_ERR_INVALID_MODE</i>	The channel is not using HBR mode

Remarks:

```
int MPG4K_SetBrightness( int      nCard,  
                        int      nChannel,  
                        signed char bBright  
                        )
```

MPG4K_SetBrightness Sets the brightness of the specified card and channels decoder

Parameters:

<i>int</i>	nCard - Specifies which MPEG4000 to control
<i>int</i>	nChannel - Specifies which channel to change the brightness
<i>signed char</i>	bBright - Specifies the new brightness value. This can be between -128 and 127. Positive values increase brightness.

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

```
int MPG4K_SetContrast( int      nCard,  
                      int      nChannel,  
                      unsigned char bContrast  
                      )
```

MPG4K_SetContrast Sets the contrast of the specified channels decoder

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel to change the contrast
unsigned char *bContrast* - Specifies the new contrast value. This can be between 0 and 255.

Returns:

int

Return values:

ID_OK Successful
ID_ERR_NODEVICE No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL Invalid card or channel number specified

Remarks:

```
int MPG4K_SetEncodingFormat( int nCard,  
                             int nFormat  
                             )
```

MPG4K_SetEncodingFormat Sets the format for the encoded video

Parameters:

int nCard - Specifies which MPEG4000 to control

int nFormat - Specifies the format.

- ID_MPEG4

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card specified
<i>ID_ERR_ENCODING</i>	The encoder is running
<i>ID_ERR_INVALID_FORMAT</i>	Invalid format is specified

Remarks:

```
int MPG4K_SetEncodingMode( int nCard,  
                           int nChannel,  
                           int nMode  
                           )
```

MPG4K_SetEncodingMode Sets the bitrate control mode.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies the channel to set the bitrate control mode for

int nMode - Specifies the bitrate control mode.

- ID_VBR - Variable Bit Rate
- ID_CBR - Constant Bit Rate
- ID_HBR - Hybrid Bit Rate

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card or channel specified
ID_ERR_INVALID_MODE	Invalid encoding mode specified

Remarks:

VBR varies the bitrate but keeps the quality the same

CBR keeps the bitrate constant but changes the quality to achieve this

HBR is bounded VBR where a maximum and minimum are set for the bitrate

```
int MPG4K_SetEncodingType( int nCard,  
                           int nChannel,  
                           int nType  
                           )
```

MPG4K_SetEncodingType Sets the encoding type.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies the channel to set the bitrate control mode for

int nType - Specifies the encoding type.

- ID_VIDEO - Video only
- ID_AUDIO - Audio only
- ID_SYSTEM - Audio and Video

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card or channel specified
ID_ERR_INVALID_MODE	Invalid encoding type specified

Remarks:

Audio and video are always encoded. This function specifies whether the output data is written to the output files

```
int MPG4K_SetFramePattern( int nCard,  
                           int nPattern  
                           )
```

MPG4K_SetFramePattern Sets the frame pattern for the output stream

Parameters:

int nCard - Specifies which MPEG4000 to control

int nPattern - Specifies the frame patter.

- ID_FRAME_I
- ID_FRAME_IP

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card specified
ID_ERR_INVALID_PATTERN	Invalid frame pattern

Remarks:

```
int MPG4K_SetFrameRate( int nCard,  
                        int nFrameRate  
                        )
```

MPG4K_SetFrameRate Sets the frame rate

Parameters:

int nCard - Specifies which MPEG4000 to control

int nFrameRate - Specifies the frame rate code Valid framerate codes are:

value	NTSC input	PAL input
0	30	25
1	15	12.5
2	7.5	6.25
3	3.75	3.125
4	1.865	1.562
5	0.9375	0.781

Returns:

int

Return values:

<i>ID_OK</i>	Success.
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card specified
<i>ID_ERR_INVALID_FRAMERATE</i>	Invalid framerate specified

Remarks:

Changing the framerate while the encoder is running will cause the output file to playback at the rate specified when MPG4K_Start is called.

To use this feature for time lapse recording, set the framerate to the maximum (pass 0), call MPG4K_Start and then set to a lower framerate.

Do not change the framerate after encoding has started if using ID_SYSTEM for the encoding type.

```
int MPG4K_SetHorizontalStart( int nCard,  
                             int nChannel,  
                             int nHStart  
                             )
```

MPG4K_SetHorizontalStart Sets the horizontal start of the specified channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control
int nChannel - Specifies which channel to change the horizontal start
int nHStart - Specifies the new horizontal start value.

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

```
int MPG4K_SetHue( int      nCard,  
                  int      nChannel,  
                  signed char bHue  
                  )
```

MPG4K_SetHue Sets the hue of the specified channels decoder

Parameters:

<i>int</i>	nCard - Specifies which MPEG4000 to control
<i>int</i>	nChannel - Specifies which channel to change the hue
<i>signed char</i>	bHue - Specifies the new hue value. This can be between -128 and 127. Units are 0.75degrees

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

```
int MPG4K_SetIInterval( int nCard,  
                        int nInterval  
                        )
```

MPG4K_SetIInterval Sets the interval of I frames

Parameters:

int nCard - Specifies which MPEG4000 to control

int nInterval - Specifies the I interval. Valid values are 2, 4, 8, 16, 32, 64, 128 and 256

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card specified
<i>ID_ERR_INVALID_INTERVAL</i>	Invalid I interval specified

Remarks:

```
int MPG4K_SetInputPath( int nCard,  
                        int nChannel,  
                        int nInput  
                        )
```

MPG4K_SetInputPath routes the video from the specified inputs to the specified channel

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies the destination channel

int nInput - Specifies the decoder to route to the specified channel.

Returns:

int

Return values:

ID_OK Successful

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid channel number specified

Remarks:

Input 0 corresponds to CompVidA

Input 1 corresponds to CompVidB

Input 2 corresponds to CompVidC

Input 3 corresponds to CompVidD

This function only works with a MPEG4000-4.

MPG4K_SelectInput can be used with this function to select each of the 16 possible inputs

In ID_SINGLE_MODE mode, channel 0 is always used for encoding and preview.

```
int MPG4K_SetInputStandard( int nCard,  
                           int nStandard  
                           )
```

MPG4K_SetInputStandard Sets the input standard

Parameters:

int nCard - Specifies which MPEG4000 to control

int nStandard - Specifies the input standard.

- ID_NTSC
- ID_PAL

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card specified
ID_ERR_INVALID_STANDARD	Invalid standard given
ID_ERR_ENCODING	The encoder is running.

Remarks:

```
int MPG4K_SetM4VFilename( int    nCard,  
                           int    nChannel,  
                           char * szFilename  
                           )
```

MPG4K_SetM4VFilename Sets the filename for the M4V output

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel the filename should be set for
char * *szFilename* - Specifies the filename to use

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified

Remarks:

If the encoder is running when this function is called, the old file will be closed and all subsequent data will be saved to the new filename.

```
int MPG4K_SetMode( int nCard,  
                  int nMode  
                  )
```

MPG4K_SetMode sets the mode for the encoder.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nMode - Specifies the mode.

- ID_QUAD_MODE - Record the four channels to four separate files.
- ID_QUAD_MODE_SINGLE - Record the four channels to a single file.
- ID_SINGLE_MODE - Record a single channel.
- ID_PIP_MODE - Record picture in picture. Channel 0 is full screen, channel 1 is 1/3 size.
- ID_POP_MODE - Record picture out picture. Channel 0 is full screen, channel 1 is 1/3 size.
- ID_POP2_MODE - Record picture out picture. Channel 0 is full screen, channel 1 is 1/3 size.

Returns:

int

Return values:

ID_OK	Succesful
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card specified
ID_ERR_INVALID_MODE	Invalid mode specified
ID_ERR_ENCODING	The encoder is running

Remarks:

In ID_QUAD_MODE the output file has CIF resolution.

In ID_SINGLE_MODE mode the output file has D1 resolution.

In ID_QUAD_MODE_SINGLE the output file has D1 resolution and each channel is CIF.

```
int MPG4K_SetMP4Filename( int    nCard,  
                           int    nChannel,  
                           char * szFilename  
                           )
```

MPG4K_SetMP4Filename Sets the filename for the MP4 output

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel the filename should be set for
char * *szFilename* - Specifies the filename to use

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified

Remarks:

If the encoder is running when this function is called, the old file will be closed and all subsequent data will be saved to the new filename.

To close the file without stopping the encoder, call this function with *szFilename*=NULL

```
int MPG4K_SetOSDColour( int      nCard,
                        int      nColour,
                        unsigned long ulFlags
                      )
```

MPG4K_SetOSDColour Sets the colour of the OSD text.

Parameters:

<i>int</i>	<i>nCard</i> - Specifies which MPEG4000 to control
<i>int</i>	<i>nColour</i> - Specifies which colour to use.
	• 0 - black • 1 - white • 2 - yellow • 3 - cyan • 4 - green • 5 - magenta • 6 - red • 7 - blue
<i>unsigned long</i>	<i>ulFlags</i> - Specifies which output the change the colour for. This is a bitmask of the following possibilities.
	• ID_CAPTURE - Change the colour on the capture • ID_PREVIEW - Change the colour on the preview

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

The OSD text can be only one colour. Changing the colour using this function will change the colour of all of the text.

```
int MPG4K_SetOutputType( int      nCard,  
                        int      nChannel,  
                        unsigned char bOutput  
                        )
```

MPG4K_SetOutputType Sets the output types for the specified channel.

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel to set the output type for
unsigned char *bOutput* - Specifies the output type. This is a bitmask of the following possibilities.

- ID_AVI_BIT - AVI file
- ID_MP4_BIT - MP4 file
- ID_M4V_BIT - Raw MPEG-4 video

Returns:

int

Return values:

ID_OK Successful
ID_ERR_NODEVICE No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL Invalid card or channel number specified

Remarks:

The AVI file produced can not be greater than 4GB.

```
int MPG4K_SetPIPPosition( int nCard,  
                           int nX,  
                           int nY  
                           )
```

MPG4K_SetPIPPosition sets the position of the PIP window within the main window.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nX - Specifies the horizontal location of the PIP window, units of 4 pixels.

int nY - Specifies the vertical location of the PIP window, units of 4 lines

Returns:

int

Return values:

ID_OK Successful

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid channel number specified

Remarks:

This functions only has an effect when the mode is ID_PIP_MODE.

```
int MPG4K_SetPreviewColourKey( int nCard,  
                                unsigned long ulColourKey,  
                                )
```

Sets the colour key for the preview window

Parameters:

int *nCard* - Specifies which MPEG4000 to control
unsigned long *ulColourKey* - Specifies the colour key to use

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified
<i>ID_ERR_NO_PREVIEW</i>	No preview hardware available
<i>ID_ERR_NO_COLOURKEY</i>	Colour keying not available

Remarks:

The colour key should be in the correct format for the current display mode.
Colour keying may not be available on all display hardware or in all display modes.
This function is currently Windows only

```
int MPG4K_SetPreviewInputPath( int nCard,  
                               int nChannel,  
                               int nInput  
                               )
```

MPG4K_SetPreviewInputPath routes the video from the specified inputs to the specified channel for preview output

Parameters:

int nCard - Specifies which MPEG4000 to control
int nChannel - Specifies the destination channel
int nInput - Specifies the decoder to route to the specified channel.

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid channel number specified

Remarks:

Input 0 corresponds to CompVidA
Input 1 corresponds to CompVidB
Input 2 corresponds to CompVidC
Input 3 corresponds to CompVidD
This function only works with a MPEG4000-4.
Calling MPG4K_SetInputPath after calling this function overrides the routing

```
int MPG4K_SetPreviewLocation( int nCard,  
                             RECT *lpRect,  
                             )
```

Set the location of the preview window

Parameters:

int nCard - Specifies which MPEG4000 to control
*RECT *lpRect* - Specifies the location of the preview window.

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified
<i>ID_ERR_NO_PREVIEW</i>	No preview hardware available
<i>All other values</i>	DirectX error values

Remarks:

The location specified in the *lpRect* is in units of pixels and is relative to the parent windows client area. This function can be used to position the preview window only after *MPG4K_StartPreview* has been called.

Under Windows, if *MPG4K_EnablePreview* has not been called, the preview window will not be visible.

Under Windows, this function will scale the preview to be the size specified

Under Linux, this function will crop the preview to be the size specified. The uncropped size of the preview will be 720x576 for PAL and 720x480 for NTSC

```
int MPG4K_SetPreviewMode( int nCard,  
                           int nMode  
                           )
```

MPG4K_SetPreviewMode sets the mode for the preview.

Parameters:

int nCard - Specifies which MPEG4000 to control

int nMode - Specifies the mode.

- ID_QUAD_MODE_SINGLE - Previews the four channels as a single file with each channel in a separate quadrant.
- ID_SINGLE_MODE - Previews a single channel.

Returns:

int

Return values:

ID_OK	Succesful
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card specified
ID_ERR_INVALID_MODE	Invalid mode specified
ID_ERR_ENCODING	The encoder is running

Remarks:

Calling MPG4K_SetMode after this function will override the mode.

```
int MPG4K_SetQuality( int nCard,  
                     int nChannel,  
                     int nQuality  
                     )
```

MPG4K_SetQuality Set the quality quantisation value for a VBR stream

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel to set the quality for

int nQuality - Specifies the quality quantisation value.

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified
<i>ID_ERR_INVALID_MODE</i>	Not in VBR mode

Remarks:

The value of nQuality can be between 1 and 31 inclusive, where 1 is the highest quality and 31 is the lowest.

```
int MPG4K_SetSaturationU( int      nCard,
                          int      nChannel,
                          unsigned char bSaturation
                          )
```

MPG4K_SetSaturationU Sets the saturation of the specified channels decoder

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel to change the saturation
unsigned char *bSaturation* - Specifies the new saturation value. This can be between 0 and 255.

Returns:

int

Return values:

ID_OK Successful
ID_ERR_NODEVICE No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL Invalid card or channel number specified

Remarks:

```
int MPG4K_SetSaturationV( int      nCard,  
                          int      nChannel,  
                          unsigned char bSaturation  
                          )
```

MPG4K_SetSaturationV Sets the saturation of the specified channels decoder

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nChannel* - Specifies which channel to change the saturation
unsigned char *bSaturation* - Specifies the new saturation value. This can be between 0 and 255.

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

```
int MPG4K_SetSingleFieldMode( int nCard,  
                             int nEnable  
                             )
```

MPG4K_SetSingleFieldMode puts the MPEG4000 encoding into single field mode.

Parameters:

int nCard - Specifies which MPEG4000 to control
int nEnable - Specifies whether to use single field mode.

- ID_ENABLE
- ID_DISABLE

Returns:

int

Return values:

ID_OK	Success
ID_ERR_NODEVICE	No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL	Invalid card specified

Remarks:

This function has no effect when encoding in quad mode.
By enabling single field mode, only the odd field of the incoming video is used. This can be used to prevent interlacing artefacts

```
int MPG4K_SetVerticalStart( int nCard,  
                           int nChannel,  
                           int nVStart  
                           )
```

MPG4K_SetVerticalStart Sets the vertical start of the specified channels decoder

Parameters:

int nCard - Specifies which MPEG4000 to control

int nChannel - Specifies which channel to change the vertical start

int nVStart - Specifies the new vertical start value.

Returns:

int

Return values:

ID_OK Successful

ID_ERR_NODEVICE No MPEG4000 was detected

ID_ERR_INVALID_CHANNEL Invalid card or channel number specified

Remarks:

```
int MPG4K_SetVideoFOURCC( int      nCard,
                          int      nChannel,
                          unsigned long ulFOURCC
                          )
```

MPG4K_SetVideoFOURCC Sets the FOURCC code for the specified channel

Parameters:

<i>int</i>	<i>nCard</i> - Specifies which MPEG4000 to control
<i>int</i>	<i>nChannel</i> - Specifies which channel to set the FOURCC
<i>unsigned long</i>	<i>ulFOURCC</i> - Specifies the FOURCC code to use. This is normally only used in AVI files to specify which decoder to use.

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

The value for the *ulFOURCC* parameter can be obtained by using the *mmioFOURCC* function. For example, the value for DIVX would be the return value from *mmioFOURCC('d','i','v','x')*

mmioFOURCC is defined as:

```
#define mmioFOURCC( ch0, ch1, ch2, ch3 ) \
( (unsigned long)(unsigned char)(ch0) | ( (unsigned long)(unsigned char) \
(ch1) << 8 ) | \
( (unsigned long)(unsigned char)(ch2) << 16 ) | ( (unsigned long)(unsigned \
char)(ch3) << 24 ) )
```

```
int MPG4K_SilenceAudio( int nCard,  
                        int nChannel,  
                        int nEnable  
                        )
```

MPG4K_SilenceAudio silences the audio data before sending it to the callback chain.

Parameters:

int nCard - Specifies which MPEG4000 to control
int nChannel - Specifies which channel to silence
unsigned long nEnable - Specifies whether to enable or disable silence. ID_ENABLE enables silence

Returns:

int

Return values:

<i>ID_OK</i>	Successful
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel number specified

Remarks:

```
int MPG4K_Start( int nCard )
```

MPG4K_Start Start the encoder

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified
<i>ID_ERR_ENCODING</i>	The encoder is already running
<i>ID_ERR_SAVE_FAIL</i>	Failed to start saving data

Remarks:

If the saving to disk was stopped using MPG4K_Stop, saving to disk will be resumed from the next I-frame

```
int MPG4K_StartPreview( int nCard,  
                        HWND hWnd  
                        )
```

MPG4K_StartPreview starts the preview engine

Parameters:

int nCard - Specifies which MPEG4000 to control
HWND hWnd - Specifies the window handle of the parent window

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified
<i>ID_ERR_PREVIEW</i>	An error occurred starting the preview
<i>ID_ERR_NO_PREVIEW</i>	No preview hardware available

Remarks:

```
int MPG4K_StartPreview( int nCard )
```

MPG4K_StartPreview starts the preview engine

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified
<i>ID_ERR_PREVIEW</i>	An error occurred starting the preview
<i>ID_ERR_NO_PREVIEW</i>	No preview hardware available

Remarks:

Linux only

```
int MPG4K_Stop( int nCard )
```

MPG4K_Stop Stop the encoded data being saved to disk

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified
<i>ID_ERR_ENCODING</i>	The encoder is not running

Remarks:

The saving to disk can be resumed by called MPG4K_Start. This will resuming saving to the same file.
To resume saving to disk to a different file, call MPG4K_SetXXXFilename prior to calling MPG4K_Start.

```
int MPG4K_StopEncoder( int nCard )
```

MPG4K_Stop Stop the encoder

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	An invalid card or channel number was specified
<i>ID_ERR_ENCODING</i>	The encoder is not running

Remarks:

Calling this function should be avoided as much as possible. To close a file that is currently being encoded call MPG4K_SetXXXFilename with a NULL filename instead. To start saving to disk again, call MPG4K_SetXXXFilename with a non-NULL filename.

int MPG4K_StopPreview(int *nCard*)

MPG4K_StopPreview stops the preview engine

Parameters:

int nCard - Specifies which MPEG4000 to control

Returns:

int

Return values:

<i>ID_OK</i>	Success
<i>ID_ERR_NODEVICE</i>	No MPEG4000 was detected
<i>ID_ERR_INVALID_CHANNEL</i>	Invalid card or channel specified
<i>ID_ERR_NO_PREVIEW</i>	No preview hardware available

Remarks:

```

int MPG4K_WriteOSD( int      nCard,
                    int      nX
                    int      nY
                    unsigned char Y
                    unsigned char Cb
                    unsigned char Cr
                    char *    szString
                    )

```

MPG4K_WriteOSD writes the specified string to the On Screen Display and changes the OSD colour

Parameters:

int *nCard* - Specifies which MPEG4000 to control
int *nX* - Specifies the horizontal location to start the string
int *nY* - Specifies the vertical location to start the string
unsigned char *Y* - Specifies the colour to use for the text
unsigned char *Cb* - Specifies the colour to use for the text
unsigned char *Cr* - Specifies the colour to use for the text
*char ** *szString* - Specifies the string to output

Returns:

int

Return values:

ID_OK Success
ID_ERR_NODEVICE No MPEG4000 was detected
ID_ERR_INVALID_CHANNEL Invalid card or channel specified

Remarks:

The locations are in units of characters. The display is 28 characters wide and 15 characters high
 The colour is specified in YCbCr format.
 The following code snippet shows how to convert from RGB to YCbCr.

```

int Y, Cb, Cr;
Y = ((306*r + 601*g + 117*b) >> 10) - 128;
Cr = ((-172*r - 340*g + 512*b) >> 10);
Cb = ((512*r - 429*g - 83*b) >> 10);
//now clamp to valid ranges
Y = Y < -128 ? -128 : Y;
Cr = Cr < -128 ? -128 : Cr;
Cb = Cb < -128 ? -128 : Cb;
Y = Y > 127 ? 127 : Y;
Cr = Cr > 127 ? 127 : Cr;
Cb = Cb > 127 ? 127 : Cr;

```

This function only works with a MPEG4000-4.