



Video Playback SDK

H264ULL-DECODER

Document version: A.00

SOFTWARE REFERENCE MANUAL



**Advanced Micro
Peripherals**
THE EMBEDDED VIDEO EXPERTS

Definitions

AMP and Advanced Micro Peripherals are the trading names for Advanced Micro Peripherals Inc. and Advanced Micro Peripherals Ltd.

Disclaimer

This document contains information on a new product. Specifications and information herein are subject to change without notice. AMP reserves the right to make changes to any products herein to improve functioning and design. Although the information in this document has been carefully reviewed and is believed to be reliable, AMP does not assume any liability arising out of the application or use of any product or circuit described herein. Nothing in this document is to be taken as a license to operate under or a recommendation to infringe any patents.

AMP does not recommend the use of any of its products in life support applications wherein a failure or malfunction of the product may directly threaten life or injury. The user of AMP products in life support applications assumes all risk of such use and indemnifies AMP against all damages. All information contained in this document is proprietary to AMP and may not be reproduced or disclosed to any third parties without the written consent of AMP.

The information contained in this document has been carefully checked and is believed to be reliable. However, Advanced Micro Peripherals Ltd (AMP) makes no guarantee or warranty concerning the accuracy of said information and shall not be responsible for any loss or damage of whatever nature resulting from the use of, or reliance upon, it. AMP does not guarantee that the use of any information herein will not infringe upon the patent, trademark, copyright, or other rights of third parties, and no patent or other license is implied hereby

AMP reserves the right to make changes in the products or specifications, or both, presented in this document at any time and without notice.

Trademarks

Tiny486, Tiny486, Tiny586DX, Tiny786LP, Mobile786EBX, Tiny886, TinyLX, VAC104, VAC104plus, The Embedded Video Experts are trademarks of Advanced Micro Peripherals.

All other trademarks recognized.

Revision History

Document Release	Date	Description	Approved
A.00		Initial release	



Advanced Micro Peripherals operates a company-wide quality management system, which has been certified by QMS International plc as compliant with ISO9001:2000

Table of contents

1: Introduction	5
Related Documentation	5
2: Hardware block diagram	6
H264ULL-DECODER	7
3: Overview of writing applications	8
Windows Library files	9
Linux Library files.....	9
Decoding.....	9
4: Structure Reference	10
5: Function Reference	13
Initialisation functions	14
MPG4K_InitCard.....	14
MPG4KX_InitCard.....	15
MPG4K_DelnitCard.....	16
Encoding control functions.....	17
MPG4KX_SetInputStandard.....	17
Decoding control functions.....	18
MPG4KX_StartDecode.....	18
MPG4KX_StopDecode.....	20
MPG4KX_QueueDecode.....	21
MPG4KX_DecoderRunning.....	22
MPG4KX_GetDecodeDuration	23
MPG4KX_GetDecodeLocation.....	24
MPG4KX_DecoderCommand.....	25
MPG4KX_DecoderCommandEx	27
MPG4KX_GetDecodeSource.....	29
MPG4KX_GetDecodeSourceID	30
MPG4KX_RemoveDecodeSource.....	31
MPG4KX_ClearDecodeQueue.....	32
A: Contacting AMP	33

Introduction

This is the API reference for the Advanced Micro Peripherals H264ULL-DECODER Video Playback SDK.

The API is the same for all supported operating systems. Functions that are specific to an OS are marked as such.

Related Documentation

Other documentation that may be of use whilst reading this document is described in the table below:

Document Name	Source
H264ULL-DECODER Hardware Reference Manual	H264ULL-DECODER SDK

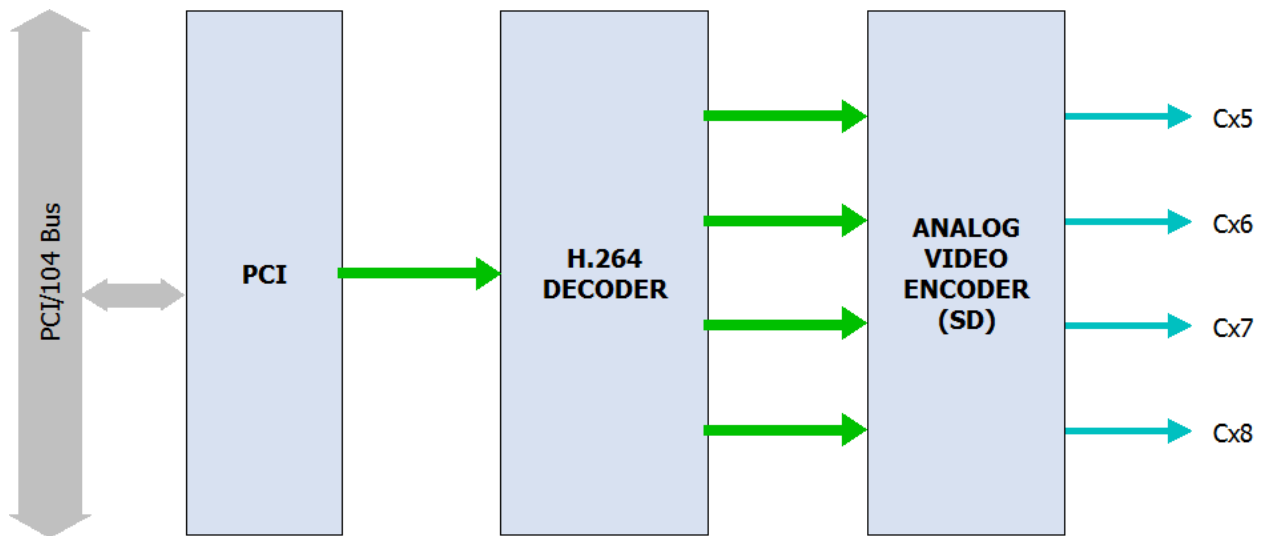
2: Hardware block diagram

H264ULL-DECODER

The analog SD video encoder on the H264ULL-DEDCODER has four digital video paths connected to the H.264 CODEC, one for each output

The encoded video data routed to the H.264 CODEC via PCI bus and is then output as analog video.

The following diagram is for the H264ULL-DECODER:



The SD outputs from the H264ULL-DECODER correspond to the following decoder channels:

Input	Channel
Cx5	7
Cx6	8
Cx7	9
Cx8	10

3:

Overview of writing applications

The H264ULL-DECODER Video PLAYBACK SDKs provides functions to control every aspect of each of the respective cards.

The ull264.h header file must be included in all applications using the H264ULL-DECODER.

The SDK library should be initialised for each card by calling the [MPG4K_InitCard](#) function. The return value should be checked to confirm that the operation succeeded. If the return is not ID_OK then the application should not continue. This must be done before any other functions from the SDK are called.

Windows Library files

The application should be linked against the h264ull.lib file. This file is found in *software/lib*

The h264ull.dll should be placed in the system directory on the target machine. Under Windows XP this is */windows/system32*

Linux Library files

The application should be linked against the libull264 file. This file is found in *software/lib*, named *libull264-glibcXXnp.so* for no onscreen preview support, where XX is the target glibc version.

The libull264 file should be placed in an appropriate library directory on the target machine. In general, either */usr/local/lib* or */usr/lib* should be used. Having copied the file, run *ldconfig* after verifying that the library path is present in the */etc/ld.so.conf* file.

Decoding

The decoding features use external plugin libraries to parse the various container formats into a form the decoder understands.

Under Windows these DLLs should be placed on the system directory on the target machine.

Under Linux, the .so files should be placed in */usr/local/lib*

4:

Structure Reference

```
typedefstruct tagErrorCallback
{
    int nCard;
    int nChannel;
    int nSource;
    int nError;
} tErrorCallback;
```

This structure is used to pass information to the error callback function. The nCard and nChannel elements specify which card and channel have generated the error.

The nSource element specifies the source of the error.

The nError element specifies the error number for the error.

```
typedefstruct tagDecodeCommand
{
    int nSize;
    unsignedlong ulCommand;
    unsignedlong ulFlags;
    union
    {
        unsignedchar bParam;
        unsignedshort usParam;
        int nParam;
        unsignedlong ulParam;
        long lParam;
#ifdef WIN32
        __int64 i64Param;
#else
        long long i64Param;
#endif
    }Param;
} tDecodeCommand;
```

This structure is used to pass commands to the decode engine.

The nSize element specifies the size of the tDecodeCommand instance and must be filled in.

The ulCommand element specifies the command.

The ulFlags element specifies flags specific to the command. It also contains flags to specify which of the Param variables is to be used.

The Param element contains an option parameter value for the command. The ulFlags element must contain the correct flag specifying which value to use.

```
#define ID_DECODE_BYTE 0x10000
```

Use the bParam entry in the Param element.

```
#define ID_DECODE_USHORT 0x20000
```

Use the usParam entry in the Param element.

```
#define ID_DECODE_ULONG 0x30000
```

Use the ulParam entry in the Param element.

```
#define ID_DECODE_INT 0x40000
```

Use the nParam entry in the Param element.

```
#define ID_DECODE_LONG 0x50000
```

Use the lParam entry in the Param element.

```
#define ID_DECODE_LONGLONG 0x60000
```

Use the i64Param entry in the Param element.

5: Function Reference

Initialisation functions

MPG4K_InitCard

```
intMPG4K_InitCard(nCard)
int nCard;    /* */
```

The **MPG4K_InitCard** function initialises the SDK and hardware.

Parameter	Description
nCard	Specifies which card to initialise

Returns

The return value is one of the following values:

Value	Meaning
ID_OK	Success
ID_ERR_NODEVICE	The card could not be opened or was not detected.
ID_ERR_NOFWFILE	The firmware file was not found.
ID_ERR_MEMALLOC	Memory allocation for the firmware failed
ID_ERR_FWUPLOAD	An error occurred uploading the firmware
ID_ERR_INVALID_CHANNEL	Invalid card number specified

Comments

The application should only continue if the return was ID_OK. All other errors are fatal.

This function enables the hardware in ID_CODEEC mode.

ID_CODEEC mode encoding is limited to a total of approximately 252,000 macroblocks across all channels.

ID_CODEEC mode is required for decode operation.

MPG4KX_InitCard

```

int MPG4KX_InitCard (nCard, nCodecMode, gopm, chansize)
int nCard;          /* */
int nCodecMode;    /* */
int gopm;          /* */
int chansize;      /* */

```

The **MPG4KX_Init** function initializes the SDK and hardware.

Parameter	Description								
nCard	Specifies the card to initialize.								
nCodecMode	Specifies the codec mode. This can be one of the following values								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>ID_CODEC</td> <td>Both encode and decode can be performed simultaneously</td> </tr> <tr> <td>ID_ENCODER</td> <td>Encode only</td> </tr> <tr> <td>ID_DECODER</td> <td>Decode only. Reserved for future use</td> </tr> </tbody> </table>	Value	Meaning	ID_CODEC	Both encode and decode can be performed simultaneously	ID_ENCODER	Encode only	ID_DECODER	Decode only. Reserved for future use
Value	Meaning								
ID_CODEC	Both encode and decode can be performed simultaneously								
ID_ENCODER	Encode only								
ID_DECODER	Decode only. Reserved for future use								
gopm	Reserved								
chansize	Reserved								

Returns

The return value is one of the following values:

Value	Meaning
ID_OK	Success
ID_ERR_NODEVICE	The card could not be opened or was not detected.
ID_ERR_NOFWFILE	The firmware file was not found.
ID_ERR_MEMALLOC	Memory allocation for the firmware failed

Comments

The application should only continue if the return was ID_OK. All other errors are fatal.

ID_CODEC mode is required for decode operation.

MPG4K_DeInitCard

```
int MPG4K_DeInitCard(nCard)
```

```
int nCard, /* */
```

The **MPG4K_DeInitCard** function de-initialises the specified card.

Parameter	Description
<i>nCard</i>	Specifies the card to de-initialise

Returns

The return value is one of the following values

Value	Meaning
ID_OK	Success
ID_ERR_NODEVICE	The card could not be opened or was not detected.

Comments

Encoding control functions

MPG4KX_SetInputStandard

```

int MPG4KX_SetInputStandard (nCard, nChannel, nStandard)
int nCard;                /* */
int nChannel;            /* */
int nStandard;          /* */

```

The **MPG4KX_SetInputStandard** function sets the standard for the input video for the specified channel.

Parameter	Description
nCard	Specifies the card to set the standard for
nChannel	Specifies the channel to set the standard for
nStandard	Specifies the video standard for the input. This can be one of the following values.
Value	Meaning
ID_PAL	PAL
ID_NTSC	NTSC
ID_AUTO	Auto detect

Returns

The return value is one of the following values.

Value	Meaning
ID_OK	Success
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_ERR_INVALID_CHANNEL	An invalid card number was specified
ID_ERR_INVALID_STANDARD	An invalid video standard was passed

Comments

This function is only required if the resolution and framerate of the H.264 stream is not full size and full framerate. Otherwise the standard will be automatically detected from the stream

Decoding control functions

MPG4KX_StartDecode

int MPG4KX_StartDecode (*nCard, nChannel, szSource*)

```
int nCard;           /* */
int nChannel;       /* */
char *szSource;     /* */
```

The **MPG4KX_StartDecode** function starts decoding the specified source.

Parameter	Description
nCard	Specifies on which card to start decoding
nChannel	Specifies the decoder channel. to use
szSource	NULL terminated string specifying the source to decode.

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_ERR_INVALID_POINTER	The specified source was invalid
ID_ERR_NOTFOUND	The specified file could not be opened or is in an incompatible format
All other non-negative numbers	ID number of file.

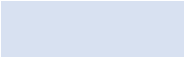
Comments

Currently only AVI files are supported as sources.

Multiple AVI files can be queued for consecutive playback with no gaps between them by calling this function multiple times or by calling [MPG4KX_QueueDecode](#) function.

The name of the file currently playing can be retrieved using the [MPG4KX_GetDecodeSource](#) function.

The ID number of the file current playing can be retrieved using the [MPG4KX_GetDecodeSourceID](#) function.



References to all queue files are stored until playback of the last file finishes. This enables skipping back to previously played files. To remove a file from the queue use the `MPG4KX_RemoveDecodeSource` function
szSource can be NULL is sources have previously been queue using [MPG4KX_QueueDecode](#) function.

MPG4KX_StopDecode

```
int MPG4KX_StopDecode (nCard, nChannel)
```

```
int nCard;           /* */
int nChannel;      /* */
```

The **MPG4KX_StopDecode** function stops the decoding.

Parameter	Description
<i>nCard</i>	Specifies on which card to stop decoding
<i>nChannel</i>	Specifies the decoder channel. to use

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_OK	Success.

Comments

This function will not return until the decoding has stopped

MPG4KX_QueueDecode

int MPG4KX_QueueDecode (*nCard*, *nChannel*, *szSource*)

```
int nCard;           /* */
int nChannel;       /* */
char *szSource;    /* */
```

The **MPG4KX_QueueDecode** function adds the specified source to the decode queue.

Parameter	Description
<i>nCard</i>	Specifies on which card to start decoding
<i>nChannel</i>	Specifies the decoder channel. to use
<i>szSource</i>	NULL terminated string specifying the source to decode.

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_ERR_INVALID_POINTER	The specified source was invalid
ID_ERR_NOTFOUND	The specified file could not be opened or is in an incompatible format
All other non-negative numbers	ID number of file.

Comments

Currently only AVI files are supported as sources.

MPG4KX_DecoderRunning

int MPG4KX_DecoderRunning (*nCard*, *nChannel*)

```
int nCard;           /* */
int nChannel;       /* */
```

The **MPG4KX_DecoderRunning** function returns whether the decoder is currently running.

Parameter	Description
<i>nCard</i>	Specifies on which card to return the decoder status
<i>nChannel</i>	Specifies the decoder channel. to use

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
0	The decoder is not running
1	The decoder is running.

Comments

MPG4KX_GetDecodeDuration

__int64 MPG4KX_GetDecodeDuration (*nCard*, *nChannel*)

```
intnCard;           /* */
intnChannel;       /* */
```

The **MPG4KX_GetDecodeDuration** function returns the duration of the current source in microseconds.

Parameter	Description
nCard	Specifies from which card to get the decode duration
nChannel	Specifies the decoder channel. to use

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
-1	The decode source is live or has no duration information
All other positive values	The duration of the source in microseconds

Comments

Under Linux the return type is long long

MPG4KX_GetDecodeLocation

__int64 MPG4KX_GetDecodeLocation (*nCard*, *nChannel*)

```
int nCard;           /* */
int nChannel;       /* */
```

The **MPG4KX_GetDecodeLocation** function returns the current decoding position of the current source in microseconds.

Parameter	Description
<i>nCard</i>	Specifies from which card to get the decode location
<i>nChannel</i>	Specifies the decoder channel to use

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
-1	The decode source is live or has no duration information
All other positive values	The current decode position of the source in microseconds

Comments

Under Linux the return type is long long

MPG4KX_DecoderCommand

int MPG4KX_DecoderCommand (*nCard*, *nChannel*, *ulCommand*)

```
int nCard;           /* */
int nChannel;       /* */
int ulCommand;      /* */
```

The **MPG4KX_DecoderCommand function** sends a command to the decode engine.

Parameter	Description	
nCard	Specifies which card to send the decode command to	
nChannel	Specifies the decoder channel. to use	
ulCommand	Specifies the command to send. The command is made up of two parts: command and parameter. The command is the least significant 8bits. The parameter is the remaining 24 bits.	
Command	Parameter	Meaning
ID_DECODE_PLAY	none	Resume normal decode
ID_DECODE_PAUSE	none	Pause the decode
ID_DECODE_STEP	none	Single frame step
ID_DECODE_FF	rate	Fast forward
ID_DECODE_RW	rate	Rewind

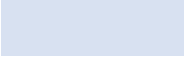
Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_OK	Success

Comments

The commands are queued so multiple calls will queue commands. Fast forward and rewind both take a rate parameter. In fast forward and rewind mode, only the I-frames within the MPEG-4 stream are played. The rate parameter specifies the number of I-frames to skip between played frames. A rate of 1 plays all I-frames, 2 plays every 2nd I-frame, etc. The normal rate is 0.



Single step mode will pause the decode and increment one frame for every single step command.

MPG4KX_DecoderCommandEx

int MPG4KX_DecoderCommandEx (*nCard, nChannel, Command*)

```
int nCard;           /* */
int nChannel;      /* */
tDecodeCommand Command; /* */
```

The **MPG4KX_DecoderCommandEx** function sends a command to the decode engine.

Parameter	Description																					
<i>nCard</i>	Specifies which card to send the decode command to																					
<i>nChannel</i>	Specifies the decoder channel. to use																					
<i>Command</i>	Specifies the command to send..																					
	<table border="1"> <thead> <tr> <th>Command</th> <th>Meaning</th> <th>Parameters</th> </tr> </thead> <tbody> <tr> <td>ID_DECODE_PLAY</td> <td>Resume normal decode</td> <td>None</td> </tr> <tr> <td>ID_DECODE_PAUSE</td> <td>Pause the decode</td> <td>None</td> </tr> <tr> <td>ID_DECODE_STEP</td> <td>Single frame step</td> <td>None</td> </tr> <tr> <td>ID_DECODE_FF</td> <td>Fast forward</td> <td>Rate</td> </tr> <tr> <td>ID_DECODE_RW</td> <td>Rewind</td> <td>Rate</td> </tr> <tr> <td>ID_DECODE_SEEK</td> <td>Seek</td> <td>Location</td> </tr> </tbody> </table>	Command	Meaning	Parameters	ID_DECODE_PLAY	Resume normal decode	None	ID_DECODE_PAUSE	Pause the decode	None	ID_DECODE_STEP	Single frame step	None	ID_DECODE_FF	Fast forward	Rate	ID_DECODE_RW	Rewind	Rate	ID_DECODE_SEEK	Seek	Location
Command	Meaning	Parameters																				
ID_DECODE_PLAY	Resume normal decode	None																				
ID_DECODE_PAUSE	Pause the decode	None																				
ID_DECODE_STEP	Single frame step	None																				
ID_DECODE_FF	Fast forward	Rate																				
ID_DECODE_RW	Rewind	Rate																				
ID_DECODE_SEEK	Seek	Location																				

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_OK	Success

Comments

The commands are queued so multiple calls will queue commands. For commands that take parameters, the `Command.ulFlags` and `Command.Param` elements must be filled in correctly.

ID_DECODE_FW and ID_DECODE_RW

The commands take a parameter to specify the rate.

A rate of 1 plays all I-frames, 2 plays every 2nd I-frame, etc. The normal rate is 0.

This can be passed using the bParam, usParam, ulParam or nParam entries in the Param element.

A negative rate reverses the direction.

ID_DECODE_STEP

Single step mode will pause the decode and increment one frame for every single step command. Single step can be combined with ID_DECODE_FW or ID_DECODE_RW to step to each I-frame.

ID_DECODE_SEEK

The seek will only happen within the current decoded file. Seeking past the start or end of a file will have no effect.

To seek based on the number of frames, the Command.ulFlags must have ID_DECODE_SEEK_FRAME set. The parameter must be an integer and contain the number of frames to seek.

To seek based on time the Command.ulFlags must have ID_DECODE_SEEK_TIME and ID_DECODE_LONGLONG set. The parameter must be an int64 and specifies the time in microseconds to seek.

The seeking can be performed relative to the current location or as an absolute location within the file.

The default is absolute. To specify relative seeking the Command.ulFlags should have ID_DECODE_SEEK_RELATIVE set.

If doing relative seeking, negative parameters will seek backwards from the current location.

All seeking is done to the nearest I-frame. Backward seeks go to the nearest I-frame before the requested location. Forward seeks go to the nearest I-frame after the requested location.

MPG4KX_GetDecodeSource

```
char *MPG4KX_GetDecodeSource (nCard, nChannel)
```

```
int nCard;           /* */
int nChannel;       /* */
```

The **MPG4KX_GetDecodeSource** function returns the name of the file currently being decoded.

Parameter	Description
nCard	Specifies from which card to get the file currently being decoded
nChannel	Specifies the decoder channel to use

Returns

The return value is one of the following values

Value	Meaning
NULL	No file is currently being decoded
Other values	NULL terminated string specifying the file currently being decoded

Comments

MPG4KX_GetDecodeSourceID

int MPG4KX_GetDecodeSourceID (*nCard*, *nChannel*)

```
int nCard;           /* */
int nChannel;      /* */
```

The **MPG4KX_GetDecodeSourceID** function returns the ID number of the file currently being decoded.

Parameter	Description
<i>nCard</i>	Specifies from which card to get the file currently being decoded
<i>nChannel</i>	Specifies the decoder channel to use

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
All other non-negative numbers	ID number

Comments

MPG4KX_RemoveDecodeSource

int MPG4KX_RemoveDecodeSource (*nCard*, *nChannel*, *nID*)

```
intnCard,           /* */
intnChannel,       /* */
intnID,            /* */
```

The **MPG4KX_RemoveDecodeSource** function removes the specified file from the decode queue.

Parameter	Description
<i>nCard</i>	Specifies from which card to remove the file from the decode queue
<i>nChannel</i>	Specifies the decoder channel to use
<i>nID</i>	Specifies the ID of the file to remove from the queue

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_ERR_NOTFOUND	The specified file could not be found
ID_OK	Success

Comments

MPG4KX_ClearDecodeQueue

```
int MPG4KX_ClearDecodeQueue (nCard, nChannel)
```

```
int nCard;           /* */
int nChannel;      /* */
```

The **MPG4KX_ClearDecodeQueue** function empties the decode queue.

Parameter	Description
<i>nCard</i>	Specifies from which card to remove the file from the decode queue
<i>nChannel</i>	Specifies the decoder channel to use

Returns

The return value is one of the following values

Value	Meaning
ID_ERR_INVALID_CHANNEL	An invalid card or channel number was specified
ID_ERR_NODEVICE	No hardware was detected or the library not initialised
ID_ERR_NOTFOUND	The specified file could not be found
ID_OK	Success

Comments

A: Contacting AMP

Sales

AMP's sales team is always available to assist you in choosing the board that best meets your requirements. Contact your local sales office or hotline.

Sales office US

Advanced Micro Peripherals Inc.
Suite 1117, 149 Madison Ave
New York
NY, 10016
USA

Tel: +1 212 951 7205
Fax: +1 212 951 7206
E-mail: sales@amp-usa.com
Web: www.amp-usa.com

Sales office UK

Advanced Micro Peripherals Ltd.
1 Harrier House, SedgewayBusinessPark
Witchford, Cambridge,
CB6 2HY
United Kingdom

Tel: +44 (0)1353 659500
Fax: 01353 659600
E-mail: sales@ampltd.com
Web: www.ampltd.com

Technical support

Comprehensive technical information is available on our websites (see above).

If you can't find the information or solution you require, AMP has a team of technical support engineers / embedded video experts available to provide a quick *and free* response to your technical queries.

Please submit your technical support query to the appropriate email address:

Technical support US

E-mail: support@amp-usa.com

Technical support UK

E-mail: support@ampltd.com