DS-40xx/41xx/42xx/43xx Series

Audio & Video Encoding/Decoding Card

SDK Documentation

(For WindowsXP/Server2003/Vista/Server2008/Windows7)

Version 6.52

2014-01-26

Notices

The information in this documentation is subject to change without notice and does not represent any commitment on behalf of HIKVISION. HIKVISION disclaims any liability whatsoever for incorrect data that may appear in this documentation. The product(s) described in this documentation are furnished subject to a license and may only be used in accordance with the terms and conditions of such license.

This documentation is issued in strict confidence and is to be used only for the purposes for which it is supplied. It may not be reproduced in whole or in part, in any form, or by any means or be used for any other purpose without prior written consent of HIKVISION and then only on the condition that this notice is included in any such reproduction. No information as to the contents or subject matter of this documentation, or any part thereof, or arising directly or indirectly therefrom, shall be given orally or in writing or shall be communicated in any manner whatsoever to any third party being an individual, firm, or company or any employee thereof without the prior written consent of HIKVISION. Use of this product is subject to acceptance of the HIKVISION agreement required to use this product. HIKVISION reserves the right to make changes to its products as circumstances may warrant, without notice.

This documentation is provided “as-is,” without warranty of any kind.

Please send any comments regarding the documentation to:
overseasbusiness@hikvision.com

Find out more about HIKVISION at www.hikvision.com

CONTENTS

1 Overview

HIKVISION DS-4xxx series card is the key component to form a powerful PC-DVR system, with full functional support such as video live view, recording/decoding and output. It adopts high performance H.264 video compression, decompression and OggVorbis audio compression, decompression by hardware. The bit rate, frame rate, frame structure (I/B/P frame mode) and image quality is configurable for each channel and it adopts built-in video loss and motion detection to meet the needs of most generic surveillance usage.

The SDK of HIKVISION DS-4xxx series card is consisted of SDK for the encoding system, SDK for network transmission and player SDK for decoding. This manual is dedicated to the functional description of encoding system SDK. As to the other types of SDK, please refer to relevant documents. Encoding system SDK (Hereinafter referred to as “the SDK”) is the local record software interface in the form of dynamic link library for 1 or multiple HIKVISION DS-4xxx card(s). There is also open-source demo software provided for the ease of developing and understanding.

When calling APIs from the SDK, software developers should especially notice that they can modify all the parameters dynamically, i.e. resolution, frame rate, bit rate, etc, except the stream type (video & audio stream / video stream / audio stream). Namely, it can adjust frame rate (SetIBPMode) and quantization coefficient (SetDefaultQuant) in the mid of a recording file, and the player is also auto-adaptive to this parameter changing situation.

Also please notice that Motion detection is independent from compression. It can be done without compression. The frame rate can be adjusted when motion happened, i.e. to a higher level for better viewing effect and more information; whereas, recording at lower frame rate to save HD space when there is no motion.

The product line and encoding/decoding performance of HIKVISION DS-4xxx card are listed on the next page.

Encoding/Decoding Performance of DS-4XXX Card

Product Series	Model	Maximum Encoding/Decoding Performance
DS-40XX	DS-4004HCI	4-ch 2CIF/CIF/QCIF real-time encoding, or 2-ch 4CIF/DCIF real-time encoding, or *4-ch 4CIF/DCIF non-real-time encoding **
	DS-4008HCI	8-ch 2CIF/CIF/QCIF real-time encoding, or 4-ch 4CIF/DCIF real-time encoding, or *8-ch 4CIF/DCIF non-real-time encoding **
	DS-4016HCI	16-ch 2CIF/CIF/QCIF real-time encoding, or 8-ch 4CIF/DCIF real-time encoding, or *16-ch 4CIF/DCIF non-real-time encoding **
	DS-4008HSI	*8-ch CIF/QCIF real-time encoding, or ** *8-ch 4CIF/DCIF/2CIF non-real-time encoding**
	DS-4016HSI	*16-ch CIF/QCIF real-time encoding, or** *16-ch 4CIF/DCIF/2CIF non-real-time encoding**
	DS-4002MDI	4-ch 2CIF/CIF/QCIF decoding, or 2-ch 4CIF/DCIF decoding
	DS-4004MDI	8-ch 2CIF/CIF/QCIF decoding, or 4-ch 4CIF/DCIF decoding
DS-41XX	DS-4101HDI	1-ch 2 Megapixel decoding, or 2-ch 720p decoding, or 4-ch 4CIF decoding
DS-42XX	DS-4208HFVI	8-ch 4CIF/2CIF/CIF/QCIF real-time encoding 8-ch sub stream CIF/QCIF encoding.
	DS-4216HFVI	16-ch 4CIF/ 2CIF/CIF/QCIF real-time encoding 16-ch sub stream CIF/QCIF encoding.
	DS-4216HCI	16-ch 2CIF/CIF/QCIF real-time encoding, or 4CIF non real-time recording 16-ch sub stream CIF/QCIF encoding
DS-43XX	DS-4308HFVI-E	8-ch 4CIF real time main stream encoding 8-ch CIF real time sub stream encoding
	DS-4308HCVI-E	16-ch CIF real time main stream encoding 16-ch QCIF real time sub stream encoding
	DS-4316HCVI-E	16-ch CIF real time main stream encoding 16-ch QCIF real time sub stream encoding
	DS-4316HFVI-E	16-ch 4CIF real time main stream encoding 16-ch CIF real time sub stream encoding
	DS-4304HFHI-E	4ch 1080P/720P real time main stream encoding 4ch CIF real time sub stream encoding
	DS-4304HDI-E	2-ch 5.0 Megapixel (2560 x 1920) decoding, or 4-ch 1080P decoding, or 8-ch 720P decoding, or 16-ch 4CIF decoding

	DS-4308HWI-E	8-ch WD1 real time main stream encoding 8-ch CIF real time sub stream encoding
	DS-4316HWI-E	16-ch WD1 real time main stream encoding 16-ch CIF real time sub stream encoding
	DS-4308MDI-E	Real time decoding of : **Standard H.264:** 11-ch WD1/800x600 13-ch 4CIF 16-ch 2CIF/DCIF/CIF/QCIF **MPEG-4:** 16-ch 4CIF 16-ch 2CIF/DCIF/CIF/QCIF **HIK H.264:** 5-ch 4CIF 16-ch 2CIF/DCIF/CIF/QCIF Or up to 16-ch non-real-time decoding of WD1/800x600/ 4CIF/ 2CIF/DCIF/CIF/QCIF

* The recording performance in *grey&Italic* font is NOT VALID for dual stream mode.
* For DS-40XX series card non-real-time encoding, DS-40XXHCI card supports 12fps - 15fps per channel, while DS-40XXHSI card supports 3fps - 5fps.

2 Version Update

V6.52.12.118(build20140118)

Updates

1. DS-43xx series encode card support H.264,MPEG4 and MJPEG video codec type selection.
2. New function:SMD(Smart Motion Detection),include corss direction and intrustion.
3. New function:VQD(Video Quality Diagnostics),include Blur detection, Luma detection and Chroma detection.

V6.5.12.507(build20130507)

Updates

1.New board_type definition for DS-4308MDI-E decoding card:

1) 16 decoding channel and 8 output channel
2) Real time video/audio decoding
3) Real time decoding capability:

Standard H.264:
11-ch WD1/800x600
13-ch 4CIF
16-ch 2CIF/DCIF/CIF/QCIF
MPEG-4:
16-ch 4CIF
16-ch 2CIF/DCIF/CIF/QCIF
HIK H.264:
5-ch 4CIF

V6.4.12.1101(build20121101)

Updates

1. New board_type definition for DS-43xxHWI-E encoding card:
   1) Up to 8ch WD1 encoding for DS-4308HWI-E and up to 16ch WD1 encoding for DS-4316HWI-E
   2) WD1 resolution for encoding, image capturing and original video stream
   3) 1ch CVBS output at D1 resolution

Version 6.3.12.731 (31st July, 2012)

Updates

1. New board_type definition for DS-4304HDI-E high definition decoding card:
   1) 16 decoding channel and 4 output channel
   2) 2-ch 5.0 Megapixel (2560 x 1920) decoding, or
      4-ch 1080P decoding, or
      8-ch 720P decoding, or

16-ch 4CIF decoding

3) HDMI output, up to 4 x 4 screen-split with configurable output resolution of:
   1920 × 1080 / 60Hz (1080P @ 60)
   1920 × 1080 / 50Hz (1080P @ 50)
   1280 × 720 / 50Hz (720P @ 50)
   1280 × 720 / 60Hz (720P @ 60)
   1280 × 960 / 60Hz
   1024 × 768 / 60Hz
4) Call API SetDisplayVideoMode() to set HDMI output resolution and frame rate.
5) Support audio output via HDMI, and this audio output is from the same audio source as BNC audio output.
6) Support 8x8 OSD enlargement for the decoding channels.

2. New board_type definition for DS-53xx series embedded DVR.

Version 6.2.12.0423 (23th April, 2012)

Special Notice

For the DS-43xxHFHI-E HD encoding card, all the coordinate-parameters related with regional settings (i.e. Logo and Mask regions) shall be portional normalized to a 4CIF region considering the downward-compatibility.

The normalizing formula is given below for reference.

Normalized_Horizontal_Coordinate = Absolute Coordinate (unit: pixel) x 704 / Absolute Width of the Resolution;

Normalized_Vertical_Coordinate = Absolute Coordinate (unit: pixel) x 480 / Absolute Height of the Resolution

Example: Convert absolute coordinate of region (x, y) under 1920x1080 resolution to normalized coordinates:

Normalized_x = 704x/1920

Normalized_y = 480y/1080

The Normalized_x and Normalized_y can be used as input parameters for API regional settings (e.g. logo and mask settings)

Updates

1. New board_type definition for DS-43xxHFHI-E High-Definition compression card
2. New video input resolution definition: 720P/25Hz, 720P/30Hz, 720P/50Hz, 720P/60Hz, 1080I/50Hz, 1080I/60Hz, 1080P/25Hz, 1080P/30Hz.
3. New HDMI output resolution definition: XGA/60Hz, SXGA/60Hz, 720P/50Hz, 720P/60Hz, 1080P/50Hz, 1080P/60Hz.
4. New resolution option for encoding, snapshot and YUV raw data: HD_F, HD_H, HD_Q.

Resolution Video In	HD_F	HD_H	HD_Q

1080P/1080I	1920*1080	1920*540	960*540
720P	1280*720	1280*360	640*360

5. API SetDisplayStandard is invalid for DS-43xxHFHI-E card. API SetDisplayVideoMode() shall be called for display configuration (output interface, frame rate and resolution).
6. For DS-43xxHFHI-E card, SetDefaultHDVideoStandard() shall be called instead of SetDefaultVideoStandard().
7. Support 8x8 OSD enlargement for the encoding/output channel of DS43xxHFHI-E.

Version 6.1.12.1129 (29th November, 2011)

Updates

1. New board_type definition for DS-4316HFVI-E compression card

Version 6.1.12.1031 (31th October, 2011)

Updates

1. New API: SetStreamPackType() for packet type configuration
2. New API: SetDeNoise() for video noise reduction
3. New API: SetSceneMode() for setting video input scene
4. New board_type definition for DS-43xx series compression cards

Version 6.0.12.411 (11th April, 2011)

Updates

1. New APIs: Start PCI live audio without connecting audio cable.
   Related APIs: SetDirectAudioPreview, HW_SetDirectAudioPreview
2. Support to get image from encode channel
   1) DS-40xxHFI/HCI/HCI+ and DS-41xx series, DS-4208HFVI and DS-4216HFVI cards, support to get the image of 4CIF, 2CIF, DCIF, CIF, or QCIF.
   2) DS-40xxHSI and DS-4216HCI only support to get the image of CIF or QCIF.
   Related APIs: GetEncoderOriginalImage, GetEncoderJpegImage
3. Support to set Unicode OSD on encode, decode and display channel
   1) Support setting flash mode
   2) Support setting colour and translucent mode of OSD and background
   3) Support 8 string lines, and can set magnification of every line independently
   4) Support adjusting brightness automatically
   5) DS-42xx series card only supports setting OSD of encode channel, and doesn't support setting the colour of OSD characters and background.
   Related APIs: SetEncoderOsdDisplayMode, SetDecoderOsdDisplayMode, SetDisplayOsdDisplayMode
4. Support to set Logo on display channel
   1) DS-40xx/41xx series, DS-4208HFVI and DS-4216HFVI cards support Logo of 4CIF(704*576)
   2) DS-4216HCI series cards support Logo of CIF(352*288)

Related APIs: SetDisplayLogo, SetDisplayLogoDisplayMode, StopDisplayLogo

5. Support DS-4101HDI decoding card

The card supports up to 4-ch decoding and 1-ch output;

1) Supports decoding stream up to 2560×1920

2) Supports decoding stream of 1-ch 1080P/1080I/UXGA; or

2-ch 720P/XVGA decoding;

3-ch SVGA decoding;

4-ch 4CIF/VGA decoding;

Related APIs: GetDisplayVideoMode, SetDisplayVideoMode, GetDisplayFormatCapability

3) Supports video matrix function:

Set encode or decode channel video stream of other card to output from the display channel of the DS-4101HDI.

Set decode channel video stream of the DS-4101HDI to output from the display channel of other card.

Note: Cross-card output will affect the card performance.

6. Support DS-4002MDI+ and DS-4004MDI+ decoding card

1) DS-4002MDI+ supports 8 decode channel & 2 output channel, and can decode 2-ch 4CIF, or 4-ch DCIF/2CIF, or 8-ch CIF/QCIF real-time.

2) DS-4004MDI+ supports 16 decode channel & 4 output channel, and can decode 4-ch 4CIF, or 8-ch DCIF/2CIF, or 16-ch CIF/QCIF at the same time.

7. Support decoding of multiple video compression standard and packet format

1) Video packet format: PS/ Private / RTP

2) Video encoding standard: Private H.264/ Standard H.264/ MPEG4

3) Audio encoding standard: OggVorbis/ MPEG1 Layer2/ MPEG2 Layer2

4) DS-40xxMDI/ DS-40xxMDI+ supports video resolution of:

i. 4CIF/2CIF/CIF/QCIF/DCIF

ii. 640×480(VGA), 320×240(QVGA), 160×120, 160×128

iii. 640×480, 320×240, 160×120 and 160×128 support 1, 4, 9, 13 and 16 windows division video output. The windows division numbers supported by the other resolutions are unchanged.

8. DS-40xxMDI/DS-40xxMDI+/DS-4101HDI support decoding non-standard stream of resolution below 4CIF:

160×120, 160×128, 400×300, 320×240, 400×304 (DSP cut to 400×300), 640×480, 640×368 (DSP cut to 640×360), 640×360 or 320×192.

9. Updates of DS-42xx series compression card :

Support getting original YUV data

Related APIs: RegisterImageStreamCallback, SetImageStream

i. DS-4216HCI supports CIF max.

ii. DS-4208HFVI and DS-4216HFVI support 4CIF max.

10. Support getting channel capability set

11. Support getting channel capability set via channel number and getting the corresponding channel capability via the name of function.

Related APIs: GetChannelCapability, CheckFunctionChannelCapability

12. Support partial zoom function on live video and draw on the display screen.
    Related APIs: ZoomOutEncoderVideoPreview, RegisterDrawFun, StopRegisterDrawFun, SetEncoderZoomOutAntiTearing
13. Support partial zoom function on decoded live video and draw on the display screen.
    Related APIs: _ZoomOutDecoderVideoPreview, HW_RegisterDrawFun, HW_StopRegisterDrawFun, SetDecoderZoomOutAntiTearing
14. Support to set video sharpness of encode channel
    Related APIs: GetEncoderVideoSharpness, SetEncoderVideoSharpness
15. Extended API used to fill display region: FillDisplayRegionEx

Version 5.1 (July 2009)

1. Compatible with DS-42xx card and DS-40xx card.
2. Support multiple screens for display by vertical or horizontal stretch.
3. DS-42xx Card supports Live audio via PCI slot, which means live audio cable isn't needed any more.
4. DS-42xxHFVI supports TV-Out function by invoking SetEncoderVideoExtOutput() and SetEncoderAudioOutput()
5. DS-4216HCI supports 4CIF/2CIF not real-time recording. While using 4CIF not real-time recording, all sub channels are disabled. .

Note:

1. **V5.1 SDK will not enable Overlay for live video automatically even the resolution is larger than 704*576, so please invoke SetPreviewOverlayMode() if Overlay is needed.**

Version 4.4 (May 2009)

1. Support new released card, DS-4216HCI,DS-4208HFVI,DS-4216HFVI

Note:

1. **Current version does not support TV-out function of DS-42xxHFVI card;**
2. **Current version is not compatible with DS-40xxHCI/HSI/MDI card;**
3. **nVidia card doesn't support Overlay;**
4. **The plaher SDK should be updated;**
5. **DS-40xxMDI card doesn't support decoding DS-42xxHFVI/HCI stream.**

Version 4.3 (22th April, 2008)

1. Support new released DS-40xxHSI card (DS-4008HSI/DS-4016HSI) whose DSP can encode 8 channels CIF/QCIF in real-time or 2CIF/DCIF/4CIF in non real-time. DS-40xxHSI supports grab YUV image, JPG image, raw stream, local matrix output, but doesn't support sub stream.
2. Support new released DS-4016HCI card which is in the same series with DS-4004HCI and DS-4008HCI.
3. Update the driver to support Windows Vista OS.
4. Improve the performance of encoding.
5. Improve the video signal detection judgement.

6. Support encoding 256 channels in one PC.
7. Support encoding Audio stream only.
8. Add the handling of color crosstalk on DS-40xxHCI, DS-40xxHCI+ card

DS-40xxMDI card:

1. Lower decoding delay of DS-40xxMDI card.
2. DS-40xxMDI supports capturing yuv420 data by invoking: RegisterDecoderVideoCaptureCallback() and HW_SetDecoderVideoCapture().
3. DS-40xxMDI supports adding user data on OSD by calling HW_RegisterDrawFun().
4. Fixed some bugs.

Version 4.2 (26th July, 2006)

1. Support new released DS-40xxHFI card;
2. Support the CRC(Cylic redundancy check)checksum when encoding by calling function SetChannelStreamCRC() for main channel stream and SetSubChannelStreamCRC() for sub channel stream;
3. Be compatible with more video display adapters , such as ATI X1300/X1600, Intel 925G integratd graphic controller;
4. New API functions: SetChannelStreamCRC(HANDLE hChannel,BOOL bEnable) and SetSubChannelStreamCRC(HANDLE hChannel,BOOL bEnable) were added;
5. Fixed some bugs;

DS-40xxMDI card:

1. Support the import and output of file index when playing
2. Support to capture the original uncompressed video stream before it's output to monitor;
3. Support the adjustment of analog output video;
4. Improve the performance of hardware decode so that the adjustment of decode speed, pause, locate operation can also be supported in stream mode;

Version 4.1 **(17th Oct. 2005)**

1. Support higher frequency DSP, which is adopted in DS-40xxHCI+ card;
2. Improve encode video quality ,especially at the resolution 4CIF;
3. Fixed some bugs;

DS-40xxMDI card:

4. Support the function to create file index when playing back files, locate the file based on time and frame number, get the start and end time of file;
5. Support to capture a video frame and convert it to a BMP file when decoding;
6. Fixed some bugs;

Version 4.0 (25th July, 2005)

1. The new version sdk supports the new card: DS-4016HCSI and DS-4004MDI. The DS-4016HCSI supports 16 CIF/QCIF realtime encode in one piece of card and supports WatchDog, alarm in and out. It doesn't support 4CIF/2CIF/DCIF

encode.

The DS-4004MDI supports 8ch 2CIF/CIF/QCIF or 4ch 4CIF decode. It supports 4 channel video matrix output;

2. The frame rate of live video can be adjusted, which ranges from 1 to 30;
3. For the motion detection, new function: SetupMotionDetectionEx() is added .The application just call s three functions to complete the motion detection. SetupMotionDetectionEx() ,StartMotionDetection() and StopMotionDetection() .
4. New function SetOsdDisplayModeEx() is added. This function supports 8 rows OSD ;
5. New function GetJpegImage() is added to capture JPEG picture.

DS-40xxMDI card:

6. Support DS-4004MDI card, the DS-4004MDI supports 8 channels' CIF/2CIF or 4 channels' 4CIF decoding, support 4 channels' video and audio output;
7. Enhance the matrix function of DS-40xxMDI card, perfect the video output ;
8. Fixed some bugs;

Version 3.3 **(8th April, 2005)**

1. Optimize the video encode quality;
2. Perfect the live video;
3. Fixed some bug;

Version 3.2 (5th March, 2005)

1. Optimize the live video, video encode quality.
2. For the motion detection, new algorithm is used. With the new algorithm, the precision of motion detection is greatly improved. Based on the function AdjustMotionDetectPrecision(),the application need to compare the parameter iGrade and 0x80000000 with the OR operator(|),then the new algorithm will be adopted .
3. For some VGA adapters, DS-40xxHCI card support live video with overlay mode, which improves the image quality and reduce the CPU usage. In the case, HWND in StartVideoPreview() must be the handle of main window and rect must be relative coordinate of sub window to main window.

   The application need call the function SetPreviewOverlayMode(BOOL bTrue) to check if the video display adapter supports the overlay mode or not .

   At present, the following VGA adapters can support overlay mode:

   ATI 7000/7200/8500/9000/9200/9600 series, X700 or higher

   Intel 845G/865G/915G series

   NVIDIA FX5200/5600 series (need use the version 6.6.9.3 driver or newer) or higher

   NVIDIA SIS651 series
4. The new DS-40xxHCI card supports PCI-X slot.
5. Each video of DS-40xxHCI card can be output to two different monitors by DS-4002MDI card;

DS-40xxMDI card:

6. Optimize the network decoding delay.
7. Consummate the PCI transmission.
8. Add 1/3 horizontal shrink, supporting 9 windows division video output.

Version 3.1 (2nd December, 2004)

1. Optimize the live video ,encode and decoder;
2. The encode resolution of sub channel can be changed dynamically.
3. New function SetDeInterlace() is added to set the intensity of DeInterlace.
4. Perfect the Logo setting .The application needn't call the function SetLogoDisplayMode() and SetLogo() sequentially.

Version 3.0 (3rd October, 2004)

1. Support DS-4002MDI (matrix decode card);

A). Hardware Decode Features:

- Each DS-4002MDI can be used to decode 4 channels image;
- Support the images made by the following products: DS-40xxHI and DS-40xxHCI series compression cards, DS-8000ME, DS-8002AH/AHL, DS-8000HCI and DS-8000SI series embedded DVRDVS;
- Audio output of any 2 channels selected from the 4 decode channels;
- Video output of any 2 channels, each video output can be made up of 16 windows max;
- Support 1 channel live audio;
- The version3.0 above SDK can support DS-40xxH, DS-40xxHCI and DS-40xxMDI at the same time;
- Most of the decode API are compatible with the SDK of DS-4004D (the old decode card);
- At present, 16 pieces of DS-4002MDI decode card can be installed in one PC, that means it can support 64 decode channels and 32 video and audio output max;
- DSP resources usage of decoding one channel (Fix bit rate, stable image):
- CIF: 12% (512Kbps), 16% (2Mbps);
- 2CIF: 30% (1Mbps);
- DCIF: 28% (768Kbps);
- 4CIF: 50% (1.5Mbps), 60% (3Mbps);
- The decode ability is always being optimized, newer version SDK will be released.

B). Matrix Features:

- Video input from either the DS-40xxHCI or DS-40xxMDI card (including the local file and network real time stream);
- Video output of DS-40xxMDI card supports screen division. The screen can be divided into 16 windows max; also can switch the image of each window.
- Matrix control: In one PC, DS-40xxMDI card can output any image of DS-40xxHCI encode channel and DS-40xxMDI decode channel, its image can be display in any window of the DS-40xxMDI video output channel.

- Each DS-4002MDI supports 2 video output, and each output can be 4CIF resolution;
- Each encode channel of DS-40xxHCI can support one graphic card live video and one matrix output at the same time;
- Each decode channel of DS-4002MDI can support one graphic card live video and two matrix output at the same time;
- Each video output can support picture in picture, and the location of each window can be adjust randomly;
- Each video output can support 4CIF+QCIF at most; it means that output a 4CIF image and a QCIF image with the mode of “picture in picture”.

2. Optimize the dual stream encode; any resolution (4CIF/2CIF/DCIF/CIF/QCIF) can be set on the sub channel. Based on the dual stream encode, the SDK delivers 2 different video streams from the same channel with different quality, different bit rate, different frame rate and different encode resolution .The application software can storage the high resolution, high quality ,high frame rate video to local disk and transmit the low bit rate ,low quality ,low encode resolution video over the internet
3. Support changing the font size of OSD character by calling SetOsdDisplayMode();
4. New functions are added to retrieve the detail information of card, DSP, encode channel, decode channel, matrix output channel.
5. Fixed some BUG.

Version 2.1 (10th September, 2004)

1. New driver is provided which supports Windows 2000, XP and Windows Server 2003.
2. Optimized the system attemper and communication, improve the data transfer efficiency, reduce the lost of frame in live video and it will be more smooth.
3. Optimize the image processing algorithm, the image quality in encoding and live video is improved.
4. Optimized the codec, greatly improve the encoding efficiency and the recording quality at the same time.
5. 2 channel 4CIF resolution real-time encoding or 4 channel 2CIF (PAL: 704*288 NTSC: 704*240) real-time encoding can be realized in the current 4004HCI card.
6. A new encoding resolution is added: Double CIF (ENC_DCIF_FORMAT) PAL: 528*384, NTSC: 528*320.
7. Compared DCIF with 2CIF, under the same bit rate, the image quality and definition is greatly improved.
8. More flexible configuration, when in the process of encoding, except stream mode (video/audio stream, video stream only), all parameters including resolution, stream code and frame structure can be changed.
9. When in the process of encoding, the change of video signal standard (PAL and NTSC) can be detected, and it will automatically switch the corresponding encoding and live video picture size.

10. Upgrade code stream format, can support the function that you need not the switch file while changing the resolution freely.
11. A new function is added: the frame rate can be set when in capture the original picture stream.
12. Fixed some BUG in the old version

Note:

1. Due to the huge calculating work in DCIF encoding, if there are 4 channel DCIF encoding simultaneously, it may lose frame. In the following versions, this kind of phenomenon will be improved with the optimization of the software.

2. Due to the version will not limit the resolution of each channel, the functions set by user may exceed the upper limit of the card itself. It may cause lost frame in recording file or failure of operation. For example:

- When there are 4 channel DCIF or 2CIF encoding with 4 channel sub-channel double encoding at same time, system will give the process as losing frames according to the complexity of pictures.
- When there are 4 channels in 4CIF encoding: Due to deficiency of the resources needed leads to the failure, it will return to error (ERR_NOT_SUPPORT).
- When there are 4 channels in CIF encoding, dynamically change the resolution into 4 CIF. Due to deficiency of the resources needed leads to the failure, it will return to error (ERR_KERNEL).

Version 1.8 (7th August, 2004)

1. Optimized the live video algorithm in small screen so that the live video picture is clearer when displayed in the small screen. .
2. Perfect the judgment to video signal input. It will give the correspondingly check and judgment to the video signal after the initiation of the card or work for a long period of time.
3. Perfect the initiation to reduce the failure of initiation.
4. Improve the display of OSD of H card so that the OSD position of H card can be adjusted in any place between (0, 0, 703, 575), the same configuration with DS-40xxHCI card.

Version 1.7 (20th July, 2004)

1. Support capturing original image stream: RegisterImageStreamCallback() and SetImageStream( )
2. Add a function to set video input position: SetInputVideoPosition( )
3. Add a function to stop callback of the picture: StopRegisterDrawFun( )
4. Add a shield function in H card.
5. When the live video window for single screen reaches to 704*576, the window live video will use Overlay mode.
6. Fixed some BUG

Version 1.6 (17th June, 2004)

1. Optimized the check system when video signal lost is detected.

2. Optimized the position of OSD and Logo can be the same as H card in CIF format.
3. Fixed BUG that SetOsd () cannot check the time.
4. Fully compatible with the new card newly promoted DS-4008HCI and DS-4001HF.

Version 1.5 (24th May, 2004)

1. Optimized the Logo bit picture definition in preview.
2. To create an off-screen buffer the same size as the live video window when in multi-window. User can draw the lines and to display the pictures in the original size rectangle frame in live video window. (Please refer to drawFun funtion in DEMO).
3. Adjust the defaulted video picture parameters.
4. Add a new function to adjust OSD time, can be used as net checking time.
5. Fixed some BUG

Version 1.0 (26th April, 2004)

1. Compared with DS-40xxM, compressed bit rate reduced by more than 30% on the premise of keeping the same image quality. In the typical circumstances such as in the office, frame rate is only 20-120kbps.
2. Provide quite accurate bit rate control mode, can output the appointed bit rate under any circumstances and CBR control mode is added.
3. Adopt the new video collecting processing chip, which can greatly decrease such phenomenon as image distortion, background strolling because of noise from video camera.
4. Use G.722 audio compressed algorithm and vocality is smoother.
5. Support PCI2.2 interface, higher transmission rate, which can stably support multiple channel encoding and recording (more than 32 channels, and the max 64 channels)
6. Will support high resolution (640*480) video compression function.
7. Add a new way to read the data stream directly, more efficient to read
8. New add screen MADK function, support max 32 regions
9. The setting of the relative coordinate (OSD, Logo MASK motion detection etc.) has been unified as 704*576, no matter what kind of encoding format
10. The live video manner is changed. When in more windows, build offscreen in the display card and BLT to the main window.
    Only in the single window full screen, it can adopt automatically Overlay manner. Through our test, nvidia Tnt/Tnt2, Geforce Mx200/400/420/440 Fx5200/5600 serial, Ati Radeon 7000/7200/7500/8500/9000/9200/9500/9600 serial, MatroxG450/550 serial are all support the new live video manner. It is suggested to use Geforce Mx420/440 or Ati serial display card to improve the efficiency of the display. Please pay attention: the drive of the display card should support the zoom function of the hardware, especially the drive of nvidia serial card, it'd better to use the new version (higher than 53.00 version). SDK interface is completely

the same as those of DS-40xxM/DS-40xxH serial cards and more functions are added. Developed internet applications can be migrated very fast.

3 Error Code Definition

Error	Code	Explanation
Encoding		
ERR_WAIT_TIMEOUT	0xc0000001	SDK operation timeout
ERR_INVALID_HANDLE	0xc0000002	Assigned invalid handle for SDK API
ERR_INVALID_ARGUMENT	0xc0000003	Invalid argument
ERR_DDRAW_CREATE_FAILED	0xc0000004	Error Returned by DDRAW *
ERR_DDRAW_CAPS_FAULT	0xc0000005	Error Returned by DDRAW *
ERR_SET_COOPERATIVELEVEL_FAILED	0xc0000006	Error Returned by DDRAW *
ERR_PRIMARY_SURFACE_CREATE_FAILED	0xc0000007	Error Returned by DDRAW *
ERR_GET_OVERLAY_ADDRESS_FAILED	0xc0000008	Error Returned by DDRAW *
ERR_OVERLAY_SURFACE_CREATE_FAILED	0xc0000009	Error Returned by DDRAW *
ERR_OVERLAY_UPDATE_FAILED	0xc000000a	Error Returned by DDRAW *
ERR_TMMAN_FAILURE	0xc000000b	Internal error of SDK
ERR_CHANNELMAGIC_MISMATCH	0xc000000c	Data corruption on the video channel
ERR_QUEUE_OVERFLOW	0xc000000e	Stream data buffer overflow
ERR__STREAM_THREAD_FAILURE	0xc000000f	Can't start stream thread
ERR_THREAD_STOP_ERROR	0xc0000010	Error when stopping the thread
ERR_NOT_SUPPORT	0xc0000011	This function can't be supported now
ERR_OUTOF_MEMORY	0xc0000012	Insufficient system memory
ERR_DSP_BUSY	0xc0000013	DSP is busy
ERR_DATA_ERROR(v2.4)	0xc0000014	Series data error, need to restart the system
ERR_KERNEL	0xc0000016	System kernel error
ERR_OFFSCREEN_CREATE_FAILED	0xc0000017	Create OFFSCREEN buffer error
ERR_MULTICLOCK_FAILURE	0xc0000018	Multimedia clock error
ERR_INVALID_DEVICE	0xc0000019	Invalid device
ERR_INVALID_DRIVER	0xc000001a	Invalid driver
ERR_OFFSCREEN_BLT_FAILED	0xc000001b	Does not support draw callback
ERR_ORDER	0xc000001c	API calling reference error
ERR_DDRAW_NONE	0xc000001d	DDRAW not installed
ERR_DDRAW7_UNSUPPORT	0xc000001e	Does not support DDRAW 7.0
ERR_GLOBAL_OVE_FAILED	0xc000001f	Does not support Overlay
ERR_DDRAW_GENERAL	0xc0000020	DirectDraw general error
Decoding		
HWERR_ALLOCATE_MEMORY	0xc1000001	Memory allocate failed
HWERR_INVALID_HANDLE	0xc1000002	Invalid handle

HWERR_DDRAW_CREATE_FAILED	0xc1000003	Create DirectDraw failed
HWERR_DDRAW_CAPS_FAULT	0xc1000004	DirectDraw get the display device failed
HWERR_SET_COOPERATIVELEVEL_FAILED	0xc1000005	DirectDraw set cooperation level failed
HWERR_PRIMARY_SURFACE_CREATE_FAILED	0xc1000006	DirectDraw create primary surface failed
HWERR_OVERLAY_CREATE_FAILED	0xc1000007	DirectDraw create overlay failed
HWERR_GET_OVERLAY_ADDRESS_FAILED	0xc1000008	DirectDraw get overlay address failed
HWERR_OVERLAY_UPDATE_FAILED	0xc1000009	DirectDraw display overlay failed
HWERR_SURFACE_NULL	0xc100000a	DirectDraw surface is null
HWERR_FILEHEADER_UNKNOWN	0xc100000b	File header error
HWERR_CREATE_FILE_FAILED	0xc100000c	Create or open file failed
HWERR_FILE_SIZE_0	0xc100000d	Length of file is 0
HWERR_FILE_SIZE_INVALID	0xc100000e	Invalid file size
HWERR_CREATE_OBJ_FAILED	0xc100000f	Create thread or object failed
HWERR_CHANNELMAGIC_MISMATCH	0xc1000010	Channel data is destroyed
HWERR_PARA_OVER	0xc1000011	Parameters are illegal
HWERR_ORDER	0xc1000012	Apply sequence error
HWERR_COMMAND	0xc1000013	Command transfer failed
HWERR_UNSUPPORTED	0xc1000014	Not supported
HWERR_DSPOPEN	0xc1000015	Open DSP failed
HWERR_DSPLOAD	0xc1000016	DSP code download failed
HWERR_ALLOCATE_DSPMEMORY	0xc1000017	DSP memory allocate failed
HWERR_DSPCHECHER	0xc1000018	DSP check failed
HWERR_IMGFILE_UNKNOWN	0xc1000019	Unknown image file
HWERR_INVALID_FILE	0xc100001a	Invalid file
HWERR_OFFSCREEN_CREATE_FAILED	0xc100001b	Create Offscreen surface failure
HWERR_OFFSCREEN_BLT_FAILED	0xc100001c	Do not support draw callback
HWERR_DDRAW_NONE	0xc100001d	System failure, DDRAW not installed
HWERR_DDRAW7_UNSUPPORT	0xc100001e	Does not support DDRAW 7.0
HWERR_GLOBAL_OVE_FAILED	0xc100001f	Does not support Overlay
HWERR_DDRAW_GENERAL	0xc1000020	DirectDraw general error

** For DDRAW error, please refer to MSDN for detail description.*

4 Data Structure

4.1 Frame Type Definition

```
typedef enum {
PktError = 0,                    // Invalid frame data
PktIFrames = 0x0001,             // I frame packet
PktPFrames = 0x0002,             // P frame packet
PktBBPFrames = 0x0004,           // BBP frame packet
PktAudioFrames = 0x0008,         // Audio frame packe for both main and sub channel
PktMotionDetection = 0x00010,    // Motion Detect packet
PktSysHeader = 0x00080,          // Encode system file header
PktBPFrames = 0x00100,           //BP frame packet
PktSFrames = 0x00200,            // Packet type when I frame captured, invalid
PktSubIFrames = 0x00400,         // I frame packet of sub channel
PktSubPFrames = 0x00800,         // P frame packet of sub channel
PktSubBBPFrames = 0x01000,       // BBP frame packet of sub channel
PktSubSysHeader = 0x02000        // Encode system file header of sub channel
}FrameType_t
```

4.2 Video Standard

```
StandardNone                     //No Video Signal
StandardNTSC                     //NTSC
StandardPAL                      //PAL
Standard720P_24HZ                //720P_24HZ
Standard720P_25HZ                //720P_25HZ
Standard720P_30HZ                //720P_30HZ
Standard720P_50HZ                //720P_50HZ
Standard720P_60HZ                //720P_60HZ
Standard1080I_50HZ               //1080I_50HZ
Standard1080I_60HZ               //1080I_60HZ
Standard1080P_24HZ               //1080P_24HZ
Standard1080P_25HZ               //1080P_25HZ
Standard1080P_30HZ               //1080P_30HZ
Standard1080P_50HZ               //1080P_50HZ
Standard1080P_60HZ               //1080P_60HZ
```

4.3 Video Resolution

```
typedef enum {
ENC_CIF_FORMAT = 0,
ENC_QCIF_FORMAT = 1,
```

```
ENC_2CIF_FORMAT = 2,
ENC_4CIF_FORMAT = 3,
ENC_QQCIF_FORMAT = 4,
ENC_CIFQCIF_FORMAT =5,
ENC_CIFQQCIF_FORMAT =6,
ENC_DCIF_FORMAT =7,
ENC_VGA_FORMAT=8,
ENC_HD_Q_FORMAT=9,
ENC_HD_H_FORMAT=10,
ENC_HD_F_FORMAT=11,
ENC_WD1_FORMAT=12
}PictureFormat_t;
```

4.4 Special Capability Definition

```
typedef struct tagChannelCapability{
    UCHAR bAudioPreview;            //Live audio
    UCHAR bAlarmIO;                 //Alarm I/O
    UCHAR bWatchDog;                //Watch Dog
}CHANNEL_CAPABILITY, *PCHANNEL_CAPABILITY;
```

4.5 Frame Data Statistics

```
typedef struct tagFramsStatistics{
    ULONG VideoFrames;              //Video Frame
    ULONG AudioFrames;              //Audio Frame
    ULONG FramesLost;               //Frame Lost
    ULONG QueueOverflow;            //Buffer Overflow
    ULONG  CurBps                   //Current bitrate: kb/s
}FRAMES_STATISTICS, *PFRAMES_STATISTICS;
```

4.6 Version Information

```
typedef struct tagVersion{
    ULONG DspVersion, DspBuildNum;      //Version and Build number of DSP
    ULONG DriverVersion, DriverBuildNum; //Version and Build number of driver
    ULONG SDKVersion, SDKBuildNum;      //Version and Build number of SDK
}VERSION_INFO, *PVERSION_INFO;
```

4.7 Mask Area

```
typedef struct {
    UINT left;          //Upper left X-axis coordinate of mask area
    UINT top;           //Upper left Y-axis coordinate of mask area
    UINT width;         //Width of mask area
    UINT height;        //Height of mask area
    COLORREF color;     //Color of mask area
}MASK_AREA_PARAM;
```

4.8 Normalized Structure of Partial Zoom Region

```
/* Normalized region, the float value is the percentage of original video width and
   height size, accurate to 0.001 */
typedef struct _ZOOM_RECT_NORMALIZE_
{
    float left;          //Upper left X-axis coordinate of zoom area
    float top;           //Upper left Y-axis coordinate of zoom area
    float width;         //Width of zoom area
    float height;        //Height of zoom area
}ZOOM_RECT_NORMALIZE;
```

5 SDK Calling Reference

Calling Order of Encode Card

A.

Set default video standard	**SetDefaultVideoStandard()**
Set default HD video standard	**SetDefaultHDVideoStandard()**

B.

Initialize DSP	**InitDSPs()**

C.

Get total encode channels	**GetTotalChannels()**
Open channel	**ChannelOpen()**
Register draw function	**RegisterDrawFun()**
Register encoded stream data callback function	**RegisterStreamDirectReadCallback()**
Register message notify	**RegisterMessageNotifyHandle()**
Register original video data callback function	**RegisterImageStreamCallback()**
Set overlay color	**SetOverlayColorKey()**

D.

Set live video mode	**SetPreviewOverlayMode()**
Start live video	**StartVideoPreview()**

E.

/* Set OSD Mode 1*/	
Set OSD display mode (support 2 rows)	**SetOsdDisplayMode()**
Set OSD display mode (supports 8 rows)	**SetOsdDisplayModeEx()**
Display OSD	**SetOsd()**
/* Set OSD Mode 2*/	
Set OSD display mode (Support foreground/background color, translucent, etc)	**SetEncoderOsdDisplayMode()**
/* Set Logo */	
Load 24-bit BMP file to YUV format	**LoadYUVFromBmpFile()**
Set Logo display mode	**SetLogoDisplayMode()**
Display Logo	**SetLogo()**

/* Set mask */	
Set video mask	**SetupMask()**

F.

Set packing type of main stream	**SetStreamPackType()**
Set encoding resolution of main channel	**SetEncoderPictureFormat()**
Set encoding stream type of main channel	**SetStreamType()**
Set encoding video quality	**SetDefaultQuant()**
Set encoding stream structure and frame rate	**SetIBPMode()**
Set maximum bit rate	**SetupBitrateControl(**
Set encoding bit rate control mode	**SetBitrateControlMode()**
Set brightness, contrast, saturation and hue	**SetVideoPara()**

G. Motion Detection Mode 1

Set motion detection sensitivity	**AdjustMotionDetectPrecision()**
Set motion detection areas	**SetupMotionDetection()**
Start motion detection	**StartMotionDetection()**
Analyze motion detect frame	**MotionAnalyzer()**

G. Motion Detection Mode 2

Set motion detection	**SetupMotionDetectionEx()**
Start motion detection	**StartMotionDetection()**

H. Capture Picture

Get one YUV422 image	**GetOriginalImage()**
Save the YUV422 image to BMP file	**SaveYUVToBmpFile()**
Get one JPEG image	**GetJpegImage()**

I. Get Sound Level and Monitor the Live Audio

Get sound volume	**GetSoundLevel()**
Live audio	**SetAudioPreview()**

J. Get the Info Related to Video, SDK and Card

Signal loss detection	**GetVideoSignal()**
Get SDK version	**GetSDKVersion()**
Get video parameter	**GetVideoPara()**
Get the type and SN of card	**GetBoardInfo()**
Get frame statistics	**GetFramesStatistics()**
Get card detail info.	**GetBoardDetail()**
Get DSP detail info.	**GetDspDetail()**

K. Start Capturing Video& Audio (Encoded Data)

Start capturing main channel data	**StartVideoCapture()**

L. Start Capturing Original Data

Start capturing original stream data	**SetImageStream()**

M. Set Parameters and Start Recording Stream of Sub Channel

Set pack type of sub stream	**SetStreamPackType()**
Set encoding stream type of sub channel	**SetSubStreamType()**
Set encoding resolution of sub channel	**SetSubEncoderPictureFormat()**
Switch to sub channel	**SetupSubChannel(, 1)**
/* Setting of other parameters is the same as main stream parameter settings. Users can set different video quality, frame rate, etc for main channel and sub channel. */	
Switch back to main channel	**SetupSubChannel(, 0)**
Start capturing video& audio of sub channel	**StartSubVideoCapture()**

N. Set Video/Audio Output of Encoding Channel

/*Set Video Output*/	
Get supported resolution	**GetDisplayFormatCapability()**
Set video output mode	**SetDisplayVideoMode()**
Get video output mode	**GetDisplayVideoMode()**
Set display regions	**SetDisplayRegion()**
Enable external output of encoding channels	**SetEncoderVideoExtOutput(, , 1, , ,)**
Disable external output of encoding channels	**SetEncoderVideoExtOutput(, , 0, , ,)**
Clear display regions	**ClearDisplayRegion()**
/*Set Audio Output*/	
Enable internal output of encoding channels	**SetEncoderAudioOutput(, 1,)**
Disable internal output of encoding channels	**SetEncoderAudioOutput(, 0,)**
Enable external output of encoding channels	**SetEncoderAudioExtOutput(, , 1, ,)**
Disable external output of encoding channels	**SeEncoderAudioExtOutput(, , 0, ,)**

O. Exit

Stop register draw function	**StopRegisterDrawFun()**
Stop getting original video data	**SetImageStream()**
Stop motion detection	**StopMotionDetection()**
Stop capturing video& audio of main channel	**StopVideoCapture()**
Stop capturing video& audio of sub channel	**StopSubVideoCapture()**
Stop live video	**StopVideoPreview()**
Close channel	**ChannelClose()**
De-initialize DSP	**DeInitDSPs()**

Note: All APIs for configuration can be called during the encoding process and adjusted dynamically except SetStreamType(), SetSubStreamType() and SetStreamPackType().

Calling Order of Decode Card

A.

Initialize decode card	**HW_InitDecDevice()**
Initialize DirectDraw	**HW_InitDirectDraw()**

B.

Open decode channel	**HW_ChannelOpen()**

C. Decode file data

Set the file index	**HW_SetFileRef()**
Open file	**HW_OpenFile()**
Set display parameter	**HW_SetDisplayPara()**
Start decoding video	**HW_Play()**
Start decoding audio	**HW_PlaySound()**
Set live audio (open)	**HW_SetAudioPreview()**
Set live audio (close)	**HW_SetAudioPreview()**
Stop decoding audio	**HW_StopSound()**
Stop decoding video	**HW_Stop()**
Close file	**HW_CloseFile()**

C. Decode stream data

Open stream	**HW_OpenStream()**
Set display parameter	**HW_SetDisplayPara()**
Start decoding video	**HW_Play()**
Set the stream open mode	**HW_SetStreamOpenMode()**
Start decoding audio	**HW_PlaySound()**
Set live audio (open)	**HW_SetAudioPreview()**
Input stream data	**HW_InputData()**
Set live audio (close)	**HW_SetAudioPreview()**
Stop decoding audio	**HW_StopSound()**
Stop decoding video	**HW_Stop()**
Stop stream	**HW_CloseStream()**

D. Set Video/Audio Output of Decoding Channel

Get supported resolution	**GetDisplayFormatCapability**
Set video output mode	**SetDisplayVideoMode**
Get video output mode	**GetDisplayVideoMode**
Set display region	**SetDisplayRegion**
Enable internal video output	**SetDecoderVideoOutput(, , 1, , ,)**
Disable internal video output	**SetDecoderVideoOutput(, , 0, , ,)**
Enable external video output	**SetDecoderVideoExtOutput(, , 1, , ,)**
Disable external video output	**SetDecoderVideoExtOutput(, , 0, , ,)**

Clear display region	**ClearDisplayRegion**
/*Set Audio Output*/	
Enable internal audio output	**SetDecoderAudioOutput(, 1,)**
Disable internal audio output	**SetDecoderAudioOutput(, 0,)**
Enable external audio output	**SetDecoderAudioExtOutput(, , 1, ,)**
Disable external audio output	**SetDecoderAudioExtOutput(, , 0, ,)**

E. Exit

Close channel handle	**HW_ChannelClose()**
Release DirectDraw	**HW_ReleaseDirectDraw**
Release decode card	**HW_ReleaseDecDevice**

6 API Description

6.1 Initialize and DeInitialize DSP

6.1.1 Initialize DSP: InitDSPs()

API:

int InitDSPs()

Parameters:

--

Return:

If the function succeeds, the return value is the total number of encode channel.

If the function fails, the return value is 0.

Remarks:

InitDSPs initializes all the HIKVISION DS-4xxx series cards installed in the PC. This API shall be called at the start of the application to make the card(s) work.

6.1.2 DeInitialize DSP: DeInitDSPs()

API:

int DeInitDSPs()

Parameters:

--

Return:

If the function succeeds, the return value is 0.

Remarks:

DeInitDSPs closes the function of all the HIKVISION DS-4xxx series cards installed in the PC, and shall be called before exiting the application software.

6.2 Get Information of Card

6.2.1 Get the Number of Card Installed: GetBoardCount()

API:

unsigned int GetBoardCount()

Parameters:

--

Return:

The total number of Hikvision encoding and decoding card(s) installed in the system.

6.2.2 Get the Number of DSP: GetDspCount()

API:

unsigned int GetDspCount()

Parameters:

--

Return:

The sum of DSP(s) installed in the system.

6.2.3 Get the Number of Encode Channel: GetEncodeChannelCount()

API:

unsigned int GetEncodeChannelCount()

Parameters:

--

Return:

The function gets the available number of encode channels.

6.2.4 Get the Number of Decode Channel: GetDecodeChannelCount()

API:

unsigned int GetDecodeChannelCount()

Parameters:

--

Return:

The function gets the available number of decode channels.

6.2.5 Get the Number of Display Channel: GetDisplayChannelCount()

API:

unsigned int GetDisplayChannelCount()

Parameters:

--

Return:

Get the available number of display channels (video output channel).

Remarks:

Call this API to get the available number of display channel (video output channel) of DS-40xxMDI/DS-40xxMDI+/DS-4101HDI card or other card with analog output.

6.2.6 Get Detail Information of Card: GetBoardDetail()

API:

int GetBoardDetail (UINT boardNum,DS_BOARD_DETAIL *pBoardDetail)

Parameters:

UINT boardNum; card index (0-based index)

DS_BOARD_DETAIL *pBoardDetail; pointer to DS_BOARD_DETAIL

```
typedef struct{
    BOARD_TYPE_DS type;
                    /* card type */
    BYTE sn[16];    /* serial number */
    UINT dspCount;
                    /* sum of DSP(s) in this card */
    UINT firstDspIndex;
                    /* index of the first DSP of this card in all DSP */
    UINT encodeChannelCount;
                    /* sum of encode channels on this card */
    UINT firstEncodeChannelIndex;
                    /* the index of the first encode channel of the current
                       card in all encode channels */
    UINT decodeChannelCount;
                    /* sum of decode channels on the current card */
    UINT firstDecodeChannelIndex;
                    /* the index of the first decode channel of the current
                       card in the all decode channels */
    UINT displayChannelCount;
                    /* sum of display channels included in the current card */
    UINT firstDisplayChannelIndex;
                    /* the index of the first video output (display) channel of
                       the current card in the all video output (display)
                       channels */
    UINT reserved1;
    UINT reserved2;
    UINT initInfo;  /* initial info and format: bit31 stands for the initialize
                       status (1 for initializing failure and 0 for initializing
                       success), bit16-30 stands for PCI slot number (its
                       value is 0), bit 8-15 stands for the BUS number and bit
                       0-7 is the device number (device number is 0 for cards)
```

```
                                */
        UINT version;       /* Hardware version, format: major.minor.build,
                               major: bit 16-19, minor: bit8-15, build: 0-7 */
}DS_BOARD_DETAIL;
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get card detail information.

6.2.7 Get Detail Information of DSP: GetDspDetail()

API:

int GetDspDetail (UINT dspNum,DSP_DETAIL *pDspDetail)

Parameters:

UINT	dspNum;	DSP index (0-based index)
DS_BOARD_DETAIL *	pDspDetail;	pointer to DSP_DETAIL

```
typedef struct
{
    UINT encodeChannelCount;
                    /* sum of encode channels included in the current DSP */
    UINT firstEncodeChannelIndex;
                    /* the index of the first encode channel of the current DSP in
                       all encode channels */
    UINT decodeChannelCount;
                    /* sum of decode channels included in the current DSP */
    UINT firstDecodeChannelIndex;
                    /* the index of the first decode channel of the current DSP in
                       all decode channels */
    UINT displayChannelCount;
                    /* sum of display channels (video output channel) included
                       in the current DSP */
    UINT firstDisplayChannelIndex;
                    /* the index of the first video output(display) channel of the
                       current DSP in all video output (display)channels */
    UINT reserved1;
    UINT reserved2;
    UINT reserved3;
    UINT initInfo;  /* initial info and format: bit31 stands for the initialize status (1
                       for initializing failure and 0 for initializing success), bit16-30
                       stands for PCI slot number(its value is 0), bit 8-15 stands
                       for the BUS number and bit 0-7 is the device number
```

(device number is 0 for cards) */

}DSP_DETAIL;

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get detail information of DSP.

6.2.8 Get the Card Model and Serial Number: GetBoardInfo()

API:

int GetBoardInfo(HANDLE hChannelHandle, ULONG *BoardType, UCHAR *SerialNo);

Parameters:

HANDLE	hChannelHandle;	channel handle
ULONG	*BoardType;	card type
UCHAR	*SerialNo;	the serial number of card; The content is ASCII number of card sequence, SerialNo[0] corresponding to the highest, SerialNo[11] corresponding to the lowest. For instance: “121300094984” corresponding to array 1,2,1,3,0,0,0,9,4,9,8,4.

```
typedef enum {
    DS4004HC=2,              //DS-4004HCI
    DS4008HC=3,              //DS-4008HCI
    DS4016HC=4,              //DS-4016HCI
    DS4004HF=6,              //DS-4004HFI
    DS4002MD=7,              //DS-4002MDI
    DS4004MD=8,              //DS-4004MDI
    DS4016HCS=9,             //DS-4016HCSI
    DS4004HC_PLUS=13,        //DS-4004HCI+
    DS4008HC_PLUS=14,        //DS-4008HCI+
    DS4008HF=16,             //DS-4008HFI
    DS4008HS=18,             //DS-4008HSI
    DS4016HS=19,             //DS-4016HSI

    DS4208HFV=23,            //DS-4208HFVI
    DS4216HC=24,             //DS-4216HCI
    DS4216HFV=25,            //DS-4216HFVI

    DS5216HF=30,             //DS-5216HFI

    DS4101HD=31,             //DS-4101HDI
```

```
    DS4002MD_PLUS=34,          //DS-4002MDI+
    DS4004MD_PLUS=35,          //DS-4004MDI+

    DS4308HCV          =37,    //DS-4308HCVI-E
    DS4308HFV          =38,    //DS-4308HFVI-E
    DS4316HCV          =39,    //DS-4316HCVI-E
    DS4316HFV          =40,    //DS-4316HFVI-E
    DS4304HD           =41,    //DS-4304HDI-E
    DS4304HFH          =42,    //DS-4304HFHI-E
    DS4304HFV          =43,    //DS-4304HFVI-E
    DS4302HFH          =44,    //DS-4302HFHI-E
    DS4308HW           =46,    //4308HWI
    DS4316HW           =47,    //4316HWI
    DS4308MD           =48,    // DS-4308MDI-E
    INVALID_BOARD_TYPE=0xffffffff,
}BOARD_TYPE_DS
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get the type and serial number of card. The application can also get the detail information of card by calling GetBoardDetail();

6.2.9 Get SDK Version: GetSDKVersion()

API:

int GetSDKVersion(PVERSION_INFO VersionInfo)

Parameters:

PVERSION_INFO VersionInfo; pointer to VERSION_INFO

```
typedef struct tagVersion{
    ULONG DspVersion, DspBuildNum;
                                //DSP version and BUILD number
    ULONG DriverVersion, DriverBuildNum;
                                //Driver version and BUILD number
    ULONG SDKVersion, SDKBuildNum;
                                //SDK version and BUILD number
}VERSION_INFO, *PVERSION_INFO;
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

6.3 API Description of Encoding Card

Open and Close Channel

6.3.1 Open Channel: ChannelOpen()

API:
HANDLE ChannelOpen(int ChannelNum);
Parameters:
int ChannelNum; channel number, index (0,1,2,3,4,…)
Return:
If the function succeeds, the return value is valid handle;
If the function fails, the return value is 0xFFFFFFFF or the error code defined on section 3.
Remarks:
Open channel and get the channel handle that will be used in other API.

6.3.2 Close Channel: ChannelClose()

API:
int ChannelClose(HANDLE hChannelHandle)
Parameters:
HANDLE hChannelHandle; channel handle
Return:
If the function succeeds, the return value is 0.
If the function fails, the return value is the error code defined on section 3.
Remarks:
Close the Channel.

Live Video

Video Live View in Overlay Mode

6.3.3 Set Live Video Mode: SetPreviewOverlayMode()

API:
int SetPreviewOverlayMode(BOOL bTrue)
Parameters:
BOOL bTrue; True - enable; FALSE - disable

Return:

If the return value is 0, it means the video display adapter supports overlay mode and SDK will use overlay mode.

If return value is non0, it means the video display adapter doesn't support the overlay mode and it will use offscreen mode.

Remarks:

For some VGA adapters, it can support live video in overlay mode, the overlay preview mode can improve the image quality and may reduce the CPU usage. In the case, HWND in StartVideoPreview() must be the handle of main window and rect must be relative coordinate of sub window to main window.

Overlay mode is closed by default. After initialization, when overlay mode is used, RegisterDrawFun() doesn't take effect.

6.3.4 Set Live Video Mode (Extended): SetPreviewOverlayModeEx()

API:

int SetPreviewOverlayModeEx(BOOL bTrue)

Parameters:

BOOL	bTrue;	true: if Graphics card supports overlay, it will enable global Overlay; false: disable Overlay.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Enable or disable global Overlay. It is extended, and adds detecting NV12.

6.3.5 Set Overlay Color: SetOverlayColorKey()

API:

int SetOverlayColorKey(COLORREF DestColorKey)

Parameters:

COLORREF	DestColorKey;	OVERLAY color

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

If you want to use overlay mode, you can call this function to set the Overlay color. By default the Overlay color is RGB(10,10,10).

The background color of preview window also should be set to the same color.

Call SetOverlayColorKey() before calling StartVideoPreview().

6.3.6 Restore Overlay Surface: RestoreOverlay()

API:

int RestoreOverlay()

Parameters:

--

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

If the overlay surface is grabbed by other application, call this function to draw back. E.g. when CTRL+ALT+DEL is pressed and the windows system will grab the overlay surface away from the application, when the grab is released this function must be called in order to restore the overlay surface.

Start and Stop Live Video

6.3.7 Start Live Video: StartVideoPreview()

API:

int StartVideoPreview(HANDLE hChannelHandle,HWND WndHandle, RECT *rect, BOOLEAN bOverlay, int VideoFormat, int FrameRate);

Parameters:

HANDLE	hChannelHandle;	Channel handle
HWND	WndHandle;	Parent window handle of display window
RECT	*rect;	Display window rectangle coordinate relative to parent window
BOOLEAN	bOverlay;	preview mode (invalid)
int	VideoFormat;	Invalid
int	FrameRate;	video frame rate (PAL: 1-25, NTSC: 1-30)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

If the function SetPreviewOverlayMode() is called and the return value is 0, then the overlay preview mode is used. Then the third parameter of function StartVideoPreview() should be the client rectangle of live video window that is relative to the main window of application .The application can draw lines and strings on the overlay surface directly.

If the function SetPreviewOverlayMode() isn't called or the return value is not 0, the SDK will preview the video with offscreen mode. Then the application can call the function RegisterDrawFun() to draw lines and rectangles on the surface of live video.

6.3.8 Stop Live Video: StopVideoPreview()

API:

int StopVideoPreview(HANDLE hChannelHandle)

Parameters:

HANDLE hChannelInfo; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

6.3.9 Set Partial Zoom of Live Video: ZoomOutEncoderVideoPreview()

API:

int ZoomOutEncoderVideoPreview(HANDLE hChannelHandle, UINT nRectIdx, BOOL bEnable, HWND hWnd, ZOOM_RECT_NORMALIZE* pZoomRect, RECT* pRect)

Parameters:

HANDLE	hChannelHandle;	channel handle
UINT	nRectIdx;	index of zoom area. 0 means display on the live view window of current encode channel after zoomed in.
BOOL	bEnable;	enable or disable the partial zoom operation.
HWND	hWnd;	window handle to display zoom area. When nRectIdx is 0, the parameter is invalid.
ZOOM_RECT_NORMALIZE*	pZoomRect;	coordinate of partial zoom area relative to encode channel video preview area, and its value is normalized. Please see to ZOOM_RECT_NORMALIZE
RECT*	pRect;	target area to display zoomed video, usually it is the client area of hWnd. When nRectIdx is 0, the parameter is invalid.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Each encode channel supports max. 16 partial zoom areas. The nRectIdx value range is [0,16]. When nRectIdx=0, it will display on current encoding preview window, hWnd and pRect are invalid now; when nRectIdx ranges from 1 to 16, it indicates this function will effect on corresponding partial zoom area.

6.3.10 Register Draw Callback Function: RegisterDrawFun()

API:

int RegisterDrawFun(DWORD nport, DRAWFUN(DrawFun), LONG nUser)

Parameters:

DWORD	nport;	//channel index. The value of higher 16 bits means the index of partial zoom area (starts from 1), and if the value is 0, it means original encoded video preview. The value of lower 16 bits means the encode channel number. E.g. nport = 0x00010000, it means NO.1 partial zoom area of NO.0 encode channel; nport = 0x00000000, it means video live view of NO.0 encode channel.
DRAWFUN(DrawFun);		//callback function
LONG	nUser;	//reserved

Callback Function:

```
#define DRAWFUN(x)
Void (CALLBACK* x) (LONG nPort, HDC hDc, LONG nUser);
```

LONG	nPort;	//channel number
HDC	hDc;	//devide context of offscreen surface is similar to the //DC of display window
LONG	nUser	//user data

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This parameter 'nport' in the API has been modified in version 6.0.

When the live video is set as offcreen mode, the application calls this function to get the device context of offscreen surface so that the application can draw lines, strings, rectangles on the surface of video .

If the function SetPreviewOverlayMode(True) is called and the return value is 0, then the RegisterDrawFun() needn't to be called .Since the live video is in Overlay mode, the application can draw lines, strings and rectangles on the Overlay sureface directly.

Note:

Before calling this API to register the draw callback of one partial zoom area, please call ZoomOutEncoderVideoPreview() to enable partial zoom function of the area.

6.3.11 Stop Register Draw Function: StopRegisterDrawFun()

API:

int StopRegisterDrawFun(DWORD nport)

Parameters:

DWORD nport; channel index

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Stop register draw function, if many channels call the function RegisterDrawFun() to draw lines and rectangles on the offcreen surface, it will result in the increase of CPU usage.The application can call this function to stop register draw function.

6.3.12 Set Vertical Sync of Partial Zoom: SetEncoderZoomOutAntiTearing()

API:

int SetEncoderZoomOutAntiTearing(HANDLE hChannelHandle, UINT nRectIdx, BOOL bAntiTearing, DWORD reserved)

Parameters:

HANDLE	hChannelHandle;	channel handle
UINT	nRectIdx;	index of zoom area, must not be less than 1
BOOL	bAntiTearing;	enable vertical sync or not
DWORD	reserved;	reserved

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Before calling this API to enable vertical sync of one partial zoom area, please call API ZoomOutEncoderVideoPreview to enable partial zoom function of the area. If API ZoomOutEncoderVideoPreview is called before to re-enable partial zoom function, the vertical sync will be invalid, and it requires calling the API again to enable the vertical sync.

Set and Get Video Parameters

6.3.13 Set Video Parameters: SetVideoPara()

API:

int SetVideoPara(HANDLE hChannelHandle, int Brightness, int Contrast, int Saturation, int Hue)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	Brightness;	value of Brightness(0—255)
int	Contrast;	value of Contrast(0—127)
int	Saturation;	value of Saturation(0—127)
int	Hue;	value of Hue(0—255)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set video parameters, this API is invalid for DS-43xxHFHI-E.

6.3.14 Get Video Parameters: GetVideoPara()

API:

int GetVideoPara(HANDLE hChannelHandle, VideoStandard_t *VideoStandard, int *Brightness, int *Contrast, int *Saturation, int *Hue);

Parameters:

HANDLE	hChannelHandle;	channel handle
VideoStandard_t	*VideoStandard;	video standard (PAL/NTSC/None)
int	*Brightness;	pointer to value of Brightness(0—255)
int	*Contrast;	pointer to value of Contrast(0—127)
int	*Saturation;	pointer to value of Saturation(0—127)
int	*Hue;	pointer to value of Hue(0—255)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get Video Parameters, parameters Brightness, Contrast, Saturation and Hue are invalid for DS-43xxHFHI-E.

6.3.15 Set Video Sharpness: SetEncoderVideoSharpness()

API:

int SetEncoderVideoSharpness(HANDLE hChannelHandle, int sharpness)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	sharpness;	sharpness parameter, value range: [0,3]. The larger the value, the greater sharpness.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

DS-4016HCSI and DS-43xxHFHI-E doesn't support this function.

6.3.16 Get Video Sharpness: GetEncoderVideoSharpness()

API:

int GetEncoderVideoSharpness(HANDLE hChannelHandle, int* pSharpness)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	*sharpness;	sharpness parameter

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

DS-4016HCSI and DS-43xxHFHI-E doesn't support this function.

Set Video Signal Parameters

6.3.17 Set Default Video Standard: SetDefaultVideoStandard()

API:

int SetDefaultVideoStandard(VideoStandard_t VideoStandard)

Parameters:

VideoStandard_t VideoStandard; video standard, the higher 16 digits stand for HD video standard (1080P_25Hz by default), and the lower 16 digits stand for SD video standard (PAL by default).

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the default video input standard.

If there's no video input for some of the channels, then these channels will be set as this default video standard.

This function shall be called before API InitDSPs().

6.3.18 Set Default HD Video Standard: SetDefaultHDVideoStandard()

API:

int __stdcall SetDefaultHDVideoStandard(VideoStandard_t VideoStandard)

Parameters:

VideoStandard_t	VideoStandard;	The default HD video standard (1080P_25Hz by default)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the default HDvideo input standard.

If there's no HD video input for some of the channels, then these channels will be set as the default HD video standard.

This function shall be called before API InitDSPs().

6.3.19 Set Video Signal Detect Precision: SetVideoDetectPrecision()

API:

int SetVideoDetectPrecision (HANDLE hChannel,unsigned int value)

Parameters:

HANDLE	hChannelHandle;	Channel handle
int value;		Sensitivity(0-100 , 20 is default)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the sensitivity of video detection .If the intensity of video signal is very weak, or the switch between signal on and signal off is too frequent, you can change video signal checkup sensibility in order to avoid the false alarm of "No Video Signal", The bigger the value, the lower the precision, and the frequency for "No Video Signal" to appear is lower. Set the value to 0xffffffff, the "No Video Signal" won't appear any more.

6.3.20 Check Video Signal Input: GetVideoSignal()

API:

int GetVideoSignal(HANDLE hChannelHandle);

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

To check the input state of video signal. This API can be used for video lost alarm.

6.3.21 Set Input Video Position: SetInputVideoPosition()

API:

int SetInputVideoPosition(HANDLE hChannelHandle,UINT x,UINT y)

Parameters:

HANDLE	hChannelHandle;	Channel handle
UINT	x ;	X coordinate, default value is 8
UINT	y ;	Y coordinate, default value is 2

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the position of video input, some cameras may preview with some black lines on the left. (x,y) is original picture coordinate of top left camera input in system processing pictures, and x must be multiple of 2 .The parameter range of (x,y) axis have relationship with the mode of the cameras.

If the appointed value does not match with the camera input, it may cause the stillness of the picture or roll in horizontal or vertical direction. Please be careful to call this function.

This API is invalid for DS-43xxHFHI-E, DS-42xx and DS-52XX series products.

DS-43xxHWI-E card supports adjustment of Y coordinate only.

6.3.22 Set DeInterlace: SetDeInterlace()

API:

int SetDeInterlace(HANDLE hChannelHandle,UINT mode,UINT level)

Parameters:

HANDLE	hChannelHandle;	channel handle
UINT	mode;	0: disable deinterlace

1: set 'level' to adjust deinterlace intensity
*Mode 1 is valid for DS-40xx series card only
If 2 is assigned (default value): SDK will use the default algorithm, there is no level adjustment for mode 2, and the parameter 'Level' will be invalid.
*Mode 2 is valid for DS-40xx, DS-42xx & DS-43xx series card

UINT level; Only valid when mode is 1, and the level ranges from 0 to 10 .The higher the level is, the stronger the intensity of DeInterlace is. 5 is the default value.
*For DS-40xx series card only

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set de-interface for the video input. Users can also adjust de-interlace level for DS-40xx series card.

6.3.23 Set Video Scene Mode: SetSceneMode()

API:

int __stdcall SetSceneMode(HANDLE hChannelHandle,UINT mode)

Parameters:

HANDLE hChannelHandle; channel handle

UINT mode; Video Scene Mode, please refer to the definition below.

```
#define VIDEO_MODE_STANDARD      0   //Standard
#define VIDEO_MODE_INDOOR        1   //Indoor
#define VIDEO_MODE_DIM_LIGHT     2   //Night
#define VIDEO_MODE_OUTDOOR       3;  //Outdoor
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This API is for DS-43xxHFVI-E, DS-43xxHCVI-E and DS-43xxHWI-E only.

Dual Stream Encoding

6.3.24 Main Stream / Sub Stream Switching: SetupSubChannel()

API:

int SetupSubChannel(HANDLE hChannelHandle, int iSubChannel)

Parameters:

HANDLE	hChannelHandle;	channel handle
Int	iSubChannel;	0 means main channel, 1 means sub channel

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This API switches channel operation between main stream and sub stream mode.

The SDK supports to deliver 2 different video streams from the same channel with different quality, bit rate, frame rate and encoding resolution. The application software can store the high resolution, high quality, high frame rate video to local disk and transmit the low bitrate, low quality, low encoding resolution video over the Internet .

When some channel is used to dual stream, the channel is divided to two sub channels. Usually 1 channel is named as main channel and the other is sub channel. Main channel is mainly used to encode high resolution,high bit rate ,high frame rate video and the sub channel is to encode low resolution,low bit rate ,low frame rate video. The frame structure, encoding bit rate, encoding resolution of both main channel and sub channel can be set independently.

Call the function SetupSubChannel() to switch between main channel and sub channel .

The code sample below shows how to switch between main channel and sub channel.

```
SetSubStreamType(ChannelHandle[i],STREAM_TYPE_AVSYNC);
SetSubEncoderPictureFormat(ChannelHandle[i],ENC_CIF_FORMAT);
SetupSubChannel(ChannelHandle[i],1);
    // switch to sub channel ,the later setting is for sub channel
SetBitrateControlMode(ChannelHandle[i], brVBR );
    //set the bit rate control mode of sub channel
SetupBitrateControl(ChannelHandle[i],256000);
    //set the max bit rate of sub channel
SetDefaultQuant(ChannelHandle[i], 18, 18, 20);
     //set the video quality of sub channel
SetIBPMode(ChannelHandle[i],200,2,2,8);
     //set the frame structure of sub channel
SetupSubChannel(ChannelHandle[i],0);
    //the setting for sub channel is finished, switch to main channel
StartSubVideoCapture(ChannelHandle[i]);
    // start video capture of sub channel
```

When the dual stream mode is not in use, all the settings are for main stream by

default and the sub channel settings are invalid.

The DSP resource is assigned to main stream in priority, as is, when the DSP is overload, frame loss will occur on sub stream first, and the SDK will protect the main stream frame rate in priority.

Another usage of the sub channel is to encode all the channels of DS-40XXHCI card (4ch per DSP) in 4CIF resolution. The frame rate of each channel will be at 12 fps - 15 fps. If the application calls function SetEncoderPictureFormat() and StartVideoCapture() to set and record in 4CIF video, only 2 channels per DSP will be valid (which could be randomly picked from each DSP).These 2 selected channels of each DSP will be real-time (PAL: 25 fps, NTSC: 30 fps).

6.3.25 Get the Stream Type of Sub Channel: GetSubChannelStreamType()

API:

int GetSubChannelStreamType(void *DataBuf, int FrameType)

Parameters:

void	*DataBuf;	address of data buffer, it is invalid.
int	FrameType;	frame type

Return:

0 other data
1 file header of main channel encoding data stream
2 file header of sub channel encoding data stream
3 video frame of main channel encoding data stream
4 video frame of sub channel encoding data stream
5 audio frame (both for main channel and sub channel)

Remarks:

Call the stream type of sub stream under dual stream mode.

Set and Get Stream Type

Note: Stream type setting is NOT dynamic adjustable.

6.3.26 Set Encoding Stream Type on Main Channel: SetStreamType()

API:

int SetStreamType(HANDLE hChannelHandle, USHORT Type);

Parameters:

HANDLE hChannelHandle; Channel handle

USHORT Type; Stream Type

The stream type macro is defined as follows:

```
#define STREAM_TYPE_VIDEO 1 //Video Stream
#define STREAM_TYPE_AUDIO 2 //Audio Stream
#define STREAM_TYPE_AVSYNC 3 //Video & Audio Stream
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set main stream type. Only DS-40xx series card supports pure audio stream.

6.3.27 Get Encoding Stream Type on Main Channel: GetStreamType()

API:

int GetStreamType(HANDLE hChannelHandle, USHORT *StreamType);

Parameters:

HANDLE hChannelHandle; channel handle

USHORT *StreamType; pointer to Stream type

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get main stream type.

6.3.28 Set Encoding Stream Type on Sub Channel: SetSubStreamType()

API:

int SetSubStreamType (HANDLE hChannelHandle, USHORT Type)

Parameters:

HANDLE hChannelHandle; channel handle

USHORT Type; stream Type

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the stream type of sub channel. Only DS-40xx series card supports pure audio stream.

6.3.29 Get Encoding Stream Type on Sub Channel: GetSubStreamType()

API:

int GetSubStreamType(HANDLE hChannelHandle, USHORT *StreamType)

Parameters:

HANDLE	hChannelHandle;	channel handle
USHORT	*StreamType;	pointer to Stream Type

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get the stream type of sub channel

Set Encoding Parameters (Dynamic Adjustable)

6.3.30 Set Video Quality: SetDefaultQuant()

API:

int SetDefaultQuant(HANDLE hChannelHandle, int IQuantVal, int PQuantVal, int BQuantVal);

Parameters:

HANDLE	hChannelHandle;	channel handle
int	IQuantVal;	I frame quantization parameters. Value: 12~30
int	PQuantVal;	P frame quantization parameters (Invalid)
int	BQuantVal;	B frame quantization parameters (Invalid)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set video encoding quantization parameters. It is used for adjusting encoding video quality, a simple rule is that lower quantization will produce higher quality image, and its range is from 12 to 30. For example: 15, 15, 20 and 18, 18, 23. The default of system is 18, 18, 23; Edition V2.0 and the above version support high quality image, and recommended value for it is 12, 12, and 17. The video quality can be adjusted

dynamically.

Quantitative coefficient is the parameter which greatly affects on encoding image quality and bit rate under MPEG4 and H.264 standard. The lower the quantitative coefficient is, the better the quality is and the higher the bit rate is. Whereas, the worse the quality is and the lower the bit rate is.

6.3.31 Set Frame Structure and Frame Rate: SetIBPMode()

API:

int SetIBPMode(HANDLE hChannelHandle, int KeyFrameIntervals, int BFrames, int PFrames, int FrameRate);

Parameters:

HANDLE hChannelHandle; channel handle

int KeyFrameIntervals; key frame intervals(12~400, 100 is default)

int BFrames; number of B frame

For DS-40xx series card, BFrames can be set as 0 or 2 (default);

For DS-42xx and DS-43xx series card, Bframe shall be set as 0 (default);

int PFrames; number of P frame (invalid)

int FrameRate; frame rate(PAL: 1~25, NTSC: 1~30, 25 is default)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set frame structure, key frame interval, number of B frame and frame rate. The value of key frame interval can be set from 12 to 400 , number of B frame can be set 0, 2 , number of P frame is fixed, the range of frame rate is from 1 to 25 (PAL),1 to 30(NTSC), and these value can be set during capturing ;

Key Frame is the image frame which is compressed within frames. It is usually looked on as original reference frame which is compressed among frames with large data capacity and better image definition. Key frame interval is the number of the frame which is encoded continuously among frames. Key frame interval needs to be properly configured to get good image quality.

Set Encoding Resolution

6.3.32 Set Main Stream Encoding Resolution: SetEncoderPictureFormat()

API:

int SetEncoderPictureFormat(HANDLE hChannelHandle, PictureForamt_t PictureFormat)

Parameters:

HANDLE	hChannelHandle;	Channel handle
PictureForamt_t	PictureFormat;	Encoding Resolution: QCIF\CIF\DCIF\2CIF\4CIF\WD1\HD_F\HD_H\HD_Q

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set encoding resolution on main channel (the resolution can be changed when encoding, it doesn't need to stop before reset the resolution)

Some of the resolution (i.e. DCIF, HD_F\HD_H\HD_Q) is not supported by all the compression cards. Please refer to the user manual for details.

DS-43xxHFVI/HCVI-E supports 4CIF/2CIF/CIF/QCIF encoding

DS-43xxHWI-E supports WD1/4CIF/2CIF/CIF/QCIF encoding

DS-4304HFHI-E supports HD_F, HD_H and HD_Q encoding additionally, these 3 resolution depends on the video input. Please refer to the following table for details.

Resolution / Video Input	HD_F	HD_H	HD_Q
Description	Original video input resolution	1/2 of original video input resolution	1/4 of original video input resolution
1080P/1080I	1920x1080	1920x540	960x540
720P	1280x720	1280x360	640x360

6.3.33 Set Sub Stream Encoding Resolution: SetSubEncoderPictureFormat()

API:

int SetSubEncoderPictureFormat(HANDLE hChannelHandle, PictureFormat_t PictureFormat)

Parameters:

HANDLE	hChannelHandle;	Channel handle
PictureFormat_t	PictureFormat;	Encoding Resolution QCIF\CIF\DCIF\2CIF\4CIF\WD1\HD_F\HD_H\HD_Q

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set encoding resolution on sub channel (the resolution can be changed when encoding, it doesn't need to stop before reset the resolution)

Call the function StartSubVideoCapture() and StopSubVideoCapture(), the application can start and stop video capture on sub channel at any time.

Sub channel of DS-42xx card only supports QCIF/CIF resolution

Some of the resolution (i.e. DCIF, HD_F\HD_H\HD_Q) is not supported by all the compression cards. Please refer to the user manual for details.

DS-43xxHFVI/HCVI-E supports 4CIF/2CIF/CIF/QCIF encoding

DS-43xxHWI-E supports WD1/4CIF/2CIF/CIF/QCIF encoding

DS-4304HFHI-E supports HD_F, HD_H and HD_Q encoding additionally, these 3 resolution depends on the video input. Please refer to the following table for details.

Resolution / Video Input	HD_F	HD_H	HD_Q
Description	Original video input resolution	1/2 of original video input resolution	1/4 of original video input resolution
1080P/1080I	1920x1080	1920x540	960x540
720P	1280x720	1280x360	640x360

Set Bit Rate and Stream Control Mode

6.3.34 Set Max Bit Rate: SetupBitrateControl()

API:

int SetupBitrateControl(HANDLE hChannelHandle, ULONG Maxbps)

Parameters:

HANDLE hChannelHandle; channel handle

ULONG Maxbps; max bit rate per second (10000b/s at least)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

For DS-42xxHCI/HFVI card, if the stream is video only, the bit rate should be set above 32*1024bps, and if the stream is Video&Audio the bitrate should be set above 96*1024bps.

Set the maximum bit rate generated by encoder card. If the parameter of MaxBps is set to 0 then bit-rate control is closed. When we set the parameter of MaxBps as a certain bit rate , the data stream generated by DSP can keep the bit rate so that it doesn't exceed by automatically adjust the encoding parameter, But if the data stream is smaller than the maximum baud rate then DSP don't bother it; the adjustable error is <10%.

6.3.35 Set Bit Rate Control Mode: SetBitrateControlMode()

API:

int SetBitrateControlMode(HANDLE hChannelHandle, BitrateControlType_t brc)

Parameters:

HANDLE	hChannelHandle;	channel handle
BitrateControlType_t	brc;	bitrate control mode, (brCBR: constant bit rate, brVBR: variable bit rate)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This function should be cooperated with SetupBitrateControl function, When brVBR is selected and SetBitrateControl is called with specified bitrate, the encoding system will output data bits which will not exceed the limit set by SetBitrateControl, if the picture quality has already reached then the output bitrate will be a lower value compared to the bitrate set .If the brCBR is set then the bitrate will be the value set by SetBitrateControl and the picture quality is adjust automatically to maintain constant bitrate.

6.3.36 Capture I Frame: CaptureIFrame()

API:

int CaptureIFrame(HANDLE hChannelHandle)

Parameters:

HANDLE	hChannelHandle;	channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Force the current frame encoded as I frame.The application can get the I frame from the encode stream and transmit it to client side.

6.3.37 Get Frame Statistics: GetFramesStatistics()

API:

int GetFramesStatistics(HANDLE hChannelHandle,
PFRAMES_STATISTICS framesStatistics);

Parameters:

HANDLE	hChannelHandle;	channel handle
PFRAMES_STATISTICS	framesStatistics;	frame statistics info

```
typedef struct tagFramsStatistics{
    ULONG VideoFrames;     /* the number of encoded video frames
                              after starting encoding */
```

```
    ULONG AudioFrames;      /* the number of encoded audio frames
                               after starting encoding */
    ULONG FramesLost;       /* lost frames */
    ULONG QueueOverflow; /* lost streams(bytes) */
    ULONG CurBps;           /* current frame rate(bps) */
}FRAMES_STATISTICS, *PFRAMES_STATISTICS
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get Frame Statistics

Set Stream Packet Type

6.3.38 Set Stream Packet Type: SetStreamPackType()

API:

int __stdcall SetStreamPackType(HANDLE hChannelHandle, USHORT Type)

Parameters:

HANDLE	hChannelHandle;	channel handle
USHORT	Type;	Stream packet type defined as below

Macro Definition:

#define STREAM_PACK_TYPE_HIKVISION 2	HIK packet type
#define STREAM_PACK_TYPE_PS 3	PS packet type

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This API is for DS-43xx series card only.

Get Data

Capture Picture (Get Image Data of 1 Frame)

Get Original Image and Save as BMP File

6.3.39 Get YUV422 Image of Fixed Resolution: GetOriginalImage()

API:

int GetOriginalImage(HANDLE hChannelHandle, UCHAR *ImageBuf, ULONG *Size);

Parameters:

HANDLE	hChannelHandle;	channel handle
UCHAR	*ImageBuf;	pointer to original image
ULONG	*Size;	[in] the size of buffer [out]the size of original image in bytes.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get an frame of original image. The original image uses standard 4CIF image format (no matter what is the encoding resolution), and the application can use the function SaveYUVToBmpFile() to save the image as a 24-bits color BMP file;

Note:

DS-4216HCI card delivers CIF raw image;

DS-40xxHSI card delivers CIF raw image;

DS-40xxHCI/HCI+/HFI/HCSI card delivers 4CIF raw image.

DS-42xxHFVI card delivers 4CIF raw image;

DS-43xxHCVI/HFVI-E card delivers 4CIF raw image.

DS-43xxHFHI-E card delivers HD_F raw image.

DS-43xxHWI-E card delivers WD1 raw image

6.3.40 Get YUV422 Original Image: GetEncoderOriginalImage()

API:

int GetEncoderOriginalImage(HANDLE hChannelHandle, UCHAR *ImageBuf, ULONG *Size,PictureFormat_t picFormat)

Parameters:

HANDLE	hChannelHandle;	channel handle
UCHAR	* ImageBuf;	buffer to save image data
ULONG	* Size;	size of the buffer

PictureFormat_t picFormat image resolution

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This API is added in version 6.0 SDK.

DS-40xxHFI/HCI/HCI+, DS-41xx series and DS-42xxHFVI cards support to get the image of 4CIF/ 2CIF/ DCIF/ CIF/ QCIF snapshot.

DS-40xxHSI and DS-4216HCI card supports CIF/ QCIF snapshot.

DS-43xxHCVI/HFVI-E series card supports 4CIF/ 2CIF/ CIF/ QCIF snapshot.

DS-43xxHFHI-E card supports HD_F/ HD_H/ HD_Q/ WD1/ 4CIF/ 2CIF/ CIF/ QCIF snapshot.

DS-43xxHWI-E card supports WD1/4CIF/2CIF/CIF/QCIF snapshot.

6.3.41 Save as BMP: SaveYUVToBmpFile()

API:

int SaveYUVToBmpFile(char *FileName, unsigned char *yuv, int Width, int Height);

Parameters:

char	*FileName;	file name
unsigned char	*yuv;	pointer to yuv buffer data(YUV422 format)
int	Width;	width of yuv picture
int	Height;	height of yuv picture

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Get JPEG Image

6.3.42 Capture JPEG of Fixed Resolution: GetJpegImage()

API:

int GetJpegImage(HANDLE hChannelHandle,UCHAR *ImageBuf,
ULONG *Size,UINT nQuality);

Parameters:

HANDLE	hChannelHandle;	channel handle
UCHAR	*ImageBuf;	pointer to data buffer
ULONG	*Size;	[in] the size of buffer [out] the size of JPEG picture in bytes.
UINT	nQuality;	JPEG picture quality, which ranges from 1

to 100, the greater the value is, the better quality the JPEG picture returned.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Capture JPEG picture of fixed resolution.

Note:

DS-4216HCI card outputs JPEG image in CIF resolution;

DS-40xxHSI card outputs JPEG image in CIF resolution;

DS-42xxHFVI card outputs JPEG image in 4CIF resolution;

DS-40xxHCI/HCSI card outputs JPEG image in 4CIF resolution.

DS-42xxHFVI card outputs JPEG image in 4CIF resolution.

DS-43xxHCVI/HFVI-E card outputs JPEG image in 4CIF resolution.

DS-43xxHFHI-E card outputs JPEG image in HD_F resolution.

DS-43xxHWI-E card outputs JPEG image in WD1 resolution.

6.3.43 Get JPEG of Specified Resolution: GetEncoderJpegImage()

API:

int GetEncoderJpegImage(HANDLE hChannelHandle, UCHAR *ImageBuf, ULONG *Size, UINT nQuality, PictureFormat_t picFormat)

Parameters:

HANDLE	hChannelHandle;	channel handle
UCHAR	* ImageBuf;	buffer to save image data
ULONG	* Size;	size of the buffer
UINT	nQuality;	image quality
PictureFormat_t	picFormat	image definition

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

It is a newly added API in version 6.0 SDK.

DS-40xxHFI/HCI/HCI+, DS-41xx series and DS-42xxHFVI cards support to get the image of 4CIF/ 2CIF/ DCIF/ CIF/ QCIF snapshot.

DS-40xxHSI and DS-4216HCI card supports CIF/ QCIF snapshot.

DS-43xxHCVI/HFVI-E series card supports 4CIF/ 2CIF/ CIF/ QCIF snapshot.

DS-43xxHFHI-E card supports HD_F/ HD_H/ HD_Q/ WD1/ 4CIF/ 2CIF/ CIF/ QCIF snapshot.

DS-43xxHWI-E card supports WD1/4CIF/2CIF/CIF/QCIF snapshot.

Start/Stop Getting YUV420 Raw Video

6.3.44 Register Callback for Raw Video: RegisterImageStreamCallback()

API:

int RegisterImageStreamCallback(IMAGE_STREAM_CALLBACK, void *context)

Parameters:

IMAGE_STREAM_CALLBACK; callback function

Callback function Explanation:

Typedef void (*IMAGE_STREAM_CALLBACK) (UINT channelNumber, void *context)

UINT channelNumber; channel index

void *context; context provided when calling this function

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Register callback function of get original image stream .The application can get realtime YUV420 data (without any compression) .

6.3.45 Start/Stop Getting Raw Video: SetImageStream()

API:

int SetImageStream(HANDLE hChannelHandle,BOOL bStart,UINT fps,UINT width,UINT height,unsigned char *imageBuffer)

Parameters:

HANDLE	hChannelHandle;	channel handle
BOOL	bStart;	TRUE/ Start Capture, FALSE/Stop Capture
UINT	fps;	frame rate
UINT	width;	image width
UINT	height;	image height
Char	*imageBuffer;	address of image buffer

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Call this function to start or stop getting the original image stream, the speed depends on the frequency of host's CPU.

Note:

DS-40xxHSI and DS-4216HCI support to get stream up to CIF. While DS-40xxHCI/HCI+/HFI, DS-4208HFVI and DS-4216HFVI support 4CIF max.

DS-43xxHFHI-E supports 1920x1080, 960x540, WD1, 4CIF, or 1/4, 1/2 of 4CIF stream if the input signal is 1080P/1080I; or 1280x720, 640x360, WD1, 4CIF, or 1/4, 1/2 of 4CIF stream if the input signal is 720P.

DS-43xxHWI-E supports WD1, 4CIF, or 1/4, 1/2 of 4CIF stream.

Not all the compression cards support DCIF resolution, and please refer to the card user manual for more details.

Capture Encoded Stream Data

Method 1 Direct Read

6.3.46 Register Direct Read Callback: RegisterStreamDirectReadCallback()

API:

int RegisterStreamDirectReadCallback(
STREAM_DIRECT_READ_CALLBACK StreamDirectReadCallback,
void* Context)

Parameters:

STREAM_ DIRECT _READ_CALLBACK StreamDirectReadCallback;	When the stream data is ready, this function will be called automatically.
Void * Context;	context when this function is called

Callback function Explanation:

STREAM_DIRECT_READ_CALLBACK (ULONG channelNumber, void* DataBuf, DWORD Length, int FrameType, void* context)

ULONG	channelNumber;	channel number
Void	* DataBuf;	address of data buffer
DWORD	Length;	length of data buffer
int	FrameType;	current encoding fram type
void	* context;	context

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Method 2 Message Notify Read

6.3.47 Setup Message Notify: SetupNotifyThreshold()

API:

int SetupNotifyThreshold(HANDLE hChannelHandle, int iFramesThreshold)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	iFramesThreshold;	threshold value (1-10)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

The application defines value of reading message, then the data of buffer could be taken out at one time. The range of iFramesThreshold is from 1 to 10. It should be called before starting video capture.

This function is valid for DS-40xxHI card only.

6.3.48 Register Message Notification: RegisterMessageNotifyHandle()

API:

int RegisterMessageNotifyHandle(HWND hWnd, UINT MessageId);

Parameters:

HWND hWnd;	channel handle
UINT MessageId;	user define message

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

When the data is ready, SDK will send user message of MessageId to the window of hWnd, then user program shall call function ReadStreamData to read data.

It is suggested using the first method to capture data from the cards.

Method 3 Direct Read (Mode 2)

6.3.49 Register Stream Read CallBack: RegisterStreamReadCallback()

API:

int RegisterStreamReadCallback(STREAM_READ_CALLBACK StreamReadCallback, void *Context)

Parameters:

STREAM_READ_CALLBACK StreamReadCallback;	when the stream data is ready, this function will be called automatically
Void * Context;	context when this function is called

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

An old callback function .the function RegisterStreamDirectReadCallback() is suggested.

6.3.50 Read Encoding Stream Data: ReadStreamData()

API:

int ReadStreamData(HANDLE hChannelHandle, void *DataBuf, DWORD * Length, int *FrameType);

Parameters:

HANDLE	hChannelHandle;	channel handle
void	*DataBuf;	pointer to data buffer
DWORD	*Length;	[in]the size of buffer
		[out]size of current encoding frame
Int	*FrameType;	frame type

Return:

If the function succeeds, the return value is 0.

If the return value is 1, it means the current encoding frame is I frame ;

If the function fails, the return value is the error code defined on section 3.

Remarks:

After calling StartVideoCapture(), the SDK thread will send out consistency stream data.

Also, it will send out stream data after calling StartMotionDetection(), if setup motion detection by calling SetupMotionDetection(). While it will only send out the motion detection message if setup motion detection by calling SetupMotionDetectionEx().

User handle will be informed as long as data is ready. The parameter Length must be assigned to the size of buffer provided by user program, the SDK will judge whether the buffer is enough to hold a complete frame, if the buffer is big enough then the length will point to the frame length read actually, otherwise the return value is error ;

If the function RegisterStreamDirectReadCallback() has been called, there is no need to call the function RegisterMessageNotifyHandle() and ReadStreamData(). Since the function RegisterStreamDirectReadCallback() was called, the stream data will be returned directly in the callback function .

Start and Stop Video Capture

6.3.51 Start Main Stream Video Capture: StartVideoCapture()

API:

int StartVideoCapture(HANDLE hChannelHandle);

Parameters:

HANDLE hChannelHandle Channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Start Video Capture, then the video and audio stream data will be returned in the callback function StreamDirectReadCallback() which is registered by the function RegisterStreamDirectReadCallback.

The application can also register a message by calling the function RegisterMessageNotifyHandle, SDK will send the message to the handler registered by the function RegisterMessageNotifyHandle, then the application calls the function ReadStreamData to read data stream;

6.3.52 Stop Main Stream Video Capture: StopVideoCapture()

API:

int StopVideoCapture(HANDLE hChannelHandle);

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Stop video capture.

6.3.53 Start Sub Stream Video Capture: StartSubVideoCapture()

API:

int StartSubVideoCapture (HANDLE hChannelHandle)

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Start video capture of sub channel

6.3.54 Stop Sub Stream Video Capture: StopSubVideoCapture()

API:

int StopSubVideoCapture (HANDLE hChannelHandle)

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Stop video capture of sub channel

Motion Detection

Method 1

6.3.55 Setup Motion Detect Precesion: AdjustMotionDetectPrecision()

API:

int AdjustMotionDetectPrecision(HANDLE hChannelHandle, int iGrade,
int iFastMotionDetectFps, int iSlowMotionDetectFps)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	iGrade;	sensitivity grade of motion analysis (0-6)
int	iFastMotionDetectFps;	frame intervals of fast motion(0-12), 0 means the sdk doesn't detect fast motion, generally it is set as 2
int	iSlowMotionDetectFps;	frame intervals of slow motion(>13),0 means the sdk doesn't detect slow motion

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

It is used for adjusting motion synthesize sensitivity, and can adjust the sensitivity of motion detection dynamically during encoding, It also decides the sensitivity of whole DSP motion synthesize. It is different from the parameter iThreshold of

MotionAnalyze function, the latter is mainly used in analyzing some area's motion by host computer. The grade 0 is most sensitive and grade 6 is most insensitive . The recommended value is 2.

Note:

Please compare the parameter iGrade and 0x80000000 with the OR operator(|), then the new algorithm will be adopted. With the new algorithm, the precision of motion detection is greatly improved.

DS-42xx card doesn't support self-adaptive motion detection.

6.3.56 Setup Motion Detect Region and Numbers: SetupMotionDetection()

API:

int SetupMotionDetection(HANDLE hChannelHandle, RECT *rectList, int numberOfAreas)

Parameters:

HANDLE	hChannelHandle;	channel handle
RECT	*rectList;	array of rectangles
int	numberOfAreas;	number of rectangles(the max is100)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Setup motion detection areas.When the sdk returns motion detect frame PktMotionDetection, call the function MotionAnalyzer() to analyze. MotionAnalyzer would analyze the areas set by function SetupMotionDetection() . If the threshold of some areas (iThreshold in MotionAnalyzer function) are reached, the finally result will be marked in returned array (iResult in MotionAnalyze function);

6.3.57 Analyze Motion Frame: MotionAnalyzer()

API:

int MotionAnalyzer(HANDLE hChannelHandle, char *MotionData, int iThreshold, int *iResult);

Parameters:

HANDLE	hChannelHandle;	channel handle
char	*MotionData;	pointer to data buffer
int	iThreshold;	the area's threshold, which is used to judge the movement(0-100)
int	*iResult;	it is the analyzed result of motion detection according to the threshold.It is an array which size is the parameter numberOfAreas in the function SetupMotionDetection(). If any element of the array is greater than 0, it means

that there is movement in the corresponding area.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Analyze the motion .The motion detection is done by DSP. The frame PktMotionDetection, which is sent to the application by DSP, is a frame that has been analyzed.

Based on the host PC,the application can call the function MotionAnalyzer() to analyze which area motions or not . The detailed motion data is provide by the frame PktMotionDetection, which is storaged in the data buffer.

Besides the function MotionAnalyzer(), the application can also analyze the frame PktMotionDetection and calucate if the video motions in the area based on the motion data.

Method 2

6.3.58 Setup Motion Detection (Extended): SetupMotionDetectionEx()

API:

int SetupMotionDetectionEx (HANDLE hChannelHandle, int iGrade, int iFastMotionDetectFps, int iSlowMotionDetectFps, UINT delay, RECT *RectList, int iAreas, MOTION_DETECTION_CALLBACK MotionDetectionCallback, int reserved);

Parameters:

HANDLE hChannelHandle;	channel handle
int iGrade;	sensitivity grade of motion analysis (0-6)
int iFastMotionDetectFps;	frame intervals of fast motion(0-12),0 means the SDK doesn't detect fast motion,generally it is set as 2
int iSlowMotionDetectFps;	frame intervals of slow motion(>13),0 means the sdk doesn't detect slow motion
UINT delay;	expired time sinece last motion was detected
RECT *RectList;	array of rectangles
int iAreas;	number of rectangles (the max is100)
MOTION_DETECTION_CALLBACK MotionDetectionCallback;	callback function
int reserved;	reserved

Callback function :

typedef void (*MOTION_DETECTION_CALLBACK)(ULONG channelNumber,

BOOL bMotionDetected, void *context);

Parameters:

ULONG channelNumber; channel index

BOOL bMotionDetected; motion flag. If motion's detected, bMotionDetected will be marked as TRUE. Otherwise, it will be marked as FALSE.

void *context; context device.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Setup motion detection.

Note:

Please compare the parameter iGrade and 0x80000000 with the OR operator(|),then the new algorithm will be adopted .With the new algorithm, the precision of motion detection is greatly improved.

DS-42xx card doesn't support self-adaptive motion detection.

Start and Stop Motion Detection

6.3.59 Start Motion Detection: StartMotionDetection()

API:

int StartMotionDetection(HANDLE hChannelHandle);

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Start motion detection, motion detection information can be transmit by data stream. When the PktMotionDetection frame is returned, the application call the function MotionAnalyzer() to analyze. The result is returned by the parameter iResult in function MotionAnalyzer.

Motion detection is independent of video decoding, so we can run motion detection without start video capture;

6.3.60 Stop Motion Detection: StopMotionDetection()

API:

int StopMotionDetection(HANDLE hChannelHandle);

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Stop motion detection.

Overlay Video Infomation

Set OSD, Logo and Mask

OSD

6.3.61 *Set OSD Display Mode: SetOsdDisplayMode()

API:

int SetOsdDisplayMode(HANDLE hChannelHandle, int Brightness,
BOOL Translucent, int param, USHORT *Format1, USHORT *Format2)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	Brightness;	brightness of OSD. Value 0~255, 255 means brightest. DS-42xx card only support B/W OSD (16: black, 235: white). If the value is less than 128, it will be black, if the value is more than 128, it will be white.
BOOL	Translucent;	whether translucent when overlay OSD string is on the active video.
Int	param;	Bit 0: 1 means the OSD will adjust the brightness of OSD automatically according to brightness of background, 0 means the OSD doesn't adjust. Bit 16-23: vertical zoom coefficient; Bit 24-31: horizontal zoom coefficient
USHORT	*Format1;	describle the position of overlay and pattern of order.
USHORT	*Format2;	

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set OSD display mode.

USHORT *Format1, *Format2, overlay position and order of description

characters are defined as following:

USHORT X, USHORT Y, CHAR0, CHAR1, CHAR2, …, CHARN, NULL.

X and Y indicate the starting coordinate of OSD in standard 4CIF image. X shall be multiple of 8 and Y can be set in the extent of image height : (0-575) PAL , (0-479) NTSC (Notice: the Y-coordinate of DS-42xx card should be multiple of 16, if not, SDK will adjust automatically). CHARN is a parameter of USHORT, which can be either ASCII or GB code. Date and time format is defined as following:

_OSD_YEAR4	show year time by length of 4 , for example: 2005
_OSD_YEAR2	show year time by length of 2, for example : 05
_OSD_MONTH3	show month time in English, for example: Jan
_OSD_MONTH2	show month time by two Arabic numerals , for example : 07
_OSD_DAY	show day time by two Arabic numerals , for example : 31
_OSD_WEEK3	show week time in English , for example : Tue
_OSD_CWEEK1	show week time in Chinese GB code , for example : Monday
_OSD_HOUR24	show 24 hours clock , for example : 18
_OSD_HOUR12	show 12 hours clock, for example: AM09 or PM09
_OSD_MINUTE	show minute time by length of 2
_OSD_SECOND	show second time by length of 2

Note:

NULL (0) should be set for the end of format strings; otherwise there will be error in contents display.

The display of string and time can be set in FORMAT1 or FORMAT2, and they can be mixed together, but the width of them can not exceed the width of one line CIF image.

The format string about showing the string of ‘office’ on the position (16,19) as follows:

USHORT Format[] = {16, 19, ‘o’,’f’,’f’,’i’,’c’,’e’,’\0’};

The time string showing on the position (8, 3) as follows:

USHORT Format[]={8, 3, _*OSD*_YEAR4, ‘/’,_OSD_MONTH2,’/’,_OSD_DAY, ‘/’,_OSD_HOUR24,’: ’, _*OSD*_MINUTE, ‘: ’, _OSD_SECOND, ‘\0’};

If we only want to show one line of them, we can define the format string as follows:

USHORT FormatNoDisplay [] = {0, 0, ‘\0’};

6.3.62 Set OSD Display Mode (Extended): SetOsdDisplayModeEx()

API:

int SetOsdDisplayModeEx(HANDLE hChannelHandle, int color, BOOL Translucent, int param,int nLineCount,USHORT **Format);

Parameters:

HANDLE hChannelHandle; channel handle

int Brightness; brightness of OSD , Value 0~255, 255 means brightest. **DS-42xx card only support black and white, 16 is**

black and 235 is white. If the value is less than 128, it will be black, if the value is more than 128, it will be white.

BOOL Translucent; whether translucent when overlay OSD string is on the active video.

int param; Bit 0: 1 means the OSD will adjust the brigtness of OSD automatically according to brightness of background, 0 means the OSD doesn't adjust.
Bit 16-23: vertical zoom coefficient
Bit 24-31: horizontal zoom coefficient

int nLineCount; spcify the line count of OSD charact (2 - 8)

USHORT **Format; pointer to pointer to USHORT(2 dimensional array or array of pointer)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This function is extended from the API SetOsdDisplayMode(). The function SetOsdDisplayMode() shows 2 rows OSD only. And the SetOsdDisplayModeEx() supports to show 8 rows OSD. The Y-coordinate of DS-42xx card should be multiple of 16, if not, SDK will adjust automatically

For the 2 dimensional array, the first and the second element of each row is the the initiative position (X, Y) of this string in the normal 4CIF image.The last element of each row must be NULL.

Example:

```
USHORT Format1[8][44] = {{2, 2, '0','1','2','3','4','5','6','7','8','9','\0'},
                         {2, 25, 'a','b','c','d','e','f','g','\0'},
                         {2, 50, 'h','i','j','k','l','m','n','o','p','q','r','s','t','\0'},
                                {2, 75, 'o','p','q','r','s','t', '\0'},
                        {2,100,_OSD_YEAR4,'-',_OSD_MONTH2,'-',_OSD_DAY,'-'
                        ,_OSD_HOUR24,':                    ',_OSD_MINUTE,':
                        ',_OSD_SECOND,'-',_OSD_MONTH3,'-',_OSD_WEEK3,_
                        OSD_CWEEK1,_OSD_HOUR12,'\0'},
                         {2,125,'h','a','n','g','z','h','o','u',' ','h','i','k','v','i','s','i','o','n', '\0'},
                         {2, 150, 'u','v','w','x','y','z','\0'},
                         {2,175,'D','S','-','4','0','x','x','H','C', 'I','\0'}};
    USHORT *Format2[8];
    for(int i=0;i<8;i++)
    {
        Format2[i]=Format1[i];
    }

    for(i = 0; i < GetTotalDSPs(); i++)
    {
```

```
        //show osd with 8 rows
        SetOsdDisplayModeEx(ChannelHandle[i], 255, TRUE, 1, 8, Format2);
        SetOsd(ChannelHandle[i], TRUE);
    }
```

6.3.63 Set OSD Display of Encode Channel: SetEncoderOsdDisplayMode()

API:

int SetEncoderOsdDisplayMode(HANDLE hChannelHandle, BOOL bEnableOsd, UINT nLuminance, UINT nFlash, COLORREF CharacterColor, COLORREF BackGroundColor, UINT nTranslucentMode, BOOL bAutoAdjustLum, int *param, int nLineCount, UINT **Format)

Parameters:

HANDLE	hChannelHandle;	channel handle
BOOL	bEnableOsd;	enable or disable OSD
UINT	nLuminance;	OSD brightness, value range: [0,255]
UINT	nFlash;	OSD flash parameter. Bit0-7: stopping seconds, Bit8-15: display seconds. E.g. flash= (2<<8)|1 means Logo display for 2 seconds, stopping for 1s.
COLORREF	CharacterColor;	colour of OSD, supports RGB format. If its value is '-1', it is of normal mode, and OSD is white.
COLORREF	BackGroundColor;	background colour of OSD, supports RGB format. If its value is '-1', it is of normal mode, and the background is transparent.
UINT	nTranslucentMode;	set to be translucent or not. 0- both foreground and background are not transparent, 1- foreground translucent and background not transparent, 2- foreground not transparent and background translucent, 3- both foreground and background translucent
BOOL	bAutoAdjustLum;	adjust brightness automatically or not. When set to adjust automatically, the defined brightness value is invalid
int	param;	int type array, to set horizontal and vertical magnification of every OSD line. Bit0-7: vertical magnification, Bit8-15: horizontal magnification
int	nLineCount;	lines of OSD display, max 8 lines
UINT	** Format;	describle the position of overlay and pattern of order.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

In OSD characters, standard resolution of ASCII character is 8×16, and standard resolution of Chinese character is 16×16. Because it requires shrinking the original image before encoding to get required resolution, so should enlarge the OSD characters and then overlay them to the original image of 4CIF, to ensure that we can clearly see OSD characters in shrinked and encoded image.

To avoid the above problem, the user can specify the size of OSD.

For example, if program wants to record video with resolution of CIF, DCIF, 2CIF, or 4CIF, we can set the horizontal and vertical magnification to 2 and 2, and the OSD position will have always been fixed. The definition of OSD in different encoding resolution is different, so if recording with QCIF, OSD characters can become blurred (That is because the image will be reduced to 1 / 4 to QCIF, while OSD characters are only enlarged to 2 times).

Specific configuration details are listed as follows:

Horizontal magnification	Vertical magnification	Fit with Video Resolution	Remark
1	1	4CIF, WD1	OSD will be blurred under other resolution
1	2	2CIF	OSD cannot be differentiated if resolution is smaller than 2CIF
2	2	CIF, DCIF	OSD cannot be differentiated if resolution is QCIF
4	4	QCIF	OSD will be very large on on other resolution
Arbitrary coefficient: 0		Auto-adaptive (set as default value)	
Other invalid value		Handled as ‘horizontal: 2, vertical: 2’	

OSD Setting for DS-43xxHFHI-E Card

Horizontal magnification	Vertical magnification	Fit with Video Resolution	Remark
1	1	HD	--
1	2	HD_F	--
2	2	4CIF, HD_Q	--
2	4	2CIF	--
4	4	CIF	--

8	8	QCIF	--
Arbitrary coefficient: 0		Auto-adaptive (default value)	
Other invalid value		Handled as 'horizontal 2, vertical 2'	

UINT **Format, overlay position and order of description characters are defined as following:

USHORT X, USHORT Y, CHAR0, CHAR1, CHAR2…, CHARN, NULL.

X and Y means the start coordinate of OSD in standard 4CIF image, and X must be the multiple of 8 , and Y can be set in the extent of image height : (0-575) PAL , (0-479) NTSC (the Y-coordinate of DS-42xx card should be multiple of 16, if not, SDK will adjust automatically). CHARN is a parameter of USHORT, it can be ASCII or GB code. Some fixed constant value descript date and time format defined as follows:

_OSD_YEAR4_EX show year time by length of 4, for example: 2005
_OSD_YEAR2_EX show year time by length of 2, for example: 05
_OSD_MONTH3_EX show month time in English, for example: Jan
_OSD_MONTH2_EX show month time by two Arabic numerals, for example: 07
_OSD_DAY_EX show day time by two Arabic numerals, for example: 31
_OSD_WEEK3_EX show week time in English, for example: Tue
_OSD_CWEEK1_EX show week time in Chinese GB code
_OSD_HOUR24_EX show 24 hours clock, for example: 18
_OSD_HOUR12_EX show 12 hours clock, for example: AM09 or PM09
_OSD_MINUTE_EX show minute time by length of 2
_OSD_SECOND show second time by length of 2, 00~59
_OSD_MILISECOND_EX show millisecond time by length of 3, 000~999
_OSD_APM_EX show a.m. or p.m. by length of 2 bit, AM or PM

Note that we must set NULL (0) in the end of format strings, otherwise there will be some error contents displayed.

OSD color settings, OSD background (CharacterColor and BackGroundColor) and display of milliseconds are designed for DS-40xx & DS-43xx series card only.

Note: OSD features of decoding card is valid for DS-40xx, DS-41xx and DS-43xx card only.

6.3.64 Enable/Disable OSD: SetOsd()

API:

int SetOsd(HANDLE hChannelHandle, BOOL Enable);

Parameters:

HANDLE hChannelHandle; channel handle
BOOL Enable; be enabled or not

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set OSD (Overlay String Display) mode. It can make the current system time (e.g. year, month, day, hour, minute and second) or custom string overlay with the real-time active video window, translucent processing is also permitted

Logo

6.3.65 Convert 24bit BMP to YUV422: LoadYUVFromBmpFile()

API:

int LoadYUVFromBmpFile(char *FileName, unsigned char *yuv, int BufLen, int *Width, int *Height);

Parameters:

char	*FileName;	file Nname
unsigned char	*yuv;	address of yuv buffer
int	BufLen;	the size of yuv buffer
int	*Width;	[out]the width of yuv picture
int	*Height;	[out]the height of yuv picture

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Convert the 24 bit BMP file to YUV422 data .The height and width of BMP file must be multiple of 16 pixels, the max size is 128*128 pixels. V4.3 SDK supports 256*128 pixels. From v6.0 SDK, DS-40xx series, DS-42xxHFVI and DS-43xx card support 4CIF Logo, and DS-4216HCI card support CIF.

6.3.66 Set Logo Display Mode: SetLogoDisplayMode()

API:

int SetLogoDisplayMode(HANDLE hChannelHandle, COLORREF ColorKey, BOOL Translucent, int Twinkle)

Parameters:

HANDLE	hChannelHandle;	channel handle
COLORREF	ColorKey;	color-key of Logo image, which will be completely translucent during display;
BOOL	Translucent;	whether translucent when overlay Logo image on the active video
int	Twinkle;	time setting of twinkle. It is formed as hex 0xXXYY, and XX is display time and YY is

the time of stopping display. Both times is measured as second. When XX and YY are all 0 it can display normally

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Call this function to set the display mode of Logo.

6.3.67 Setup Logo: SetLogo()

API:

int SetLogo(HANDLE hChannelHandle, int x, int y, int w, int h,
unsigned char *yuv);

Parameters:

HANDLE hChannelHandle; channel handle

int x; specifies the x-coordinate of the upper-left corner of the Logo (0-703)

int y; specifies the y-coordinate of the upper-left corner of the Logo (0-575)

int w; width(0-256), it should be the same as the width of Logo picture and be multiple of 16

int h; height (0-128), it should be the same as the width of Logo picture and be multiple of 16

unsigned char *yuv; pointer to YUV422 buffer data

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the Logo and position of Logo; The application can call LoadYUVFromBmpFile() to get yuv422 data from 24 bit BMP file, the transparency is done by DSP.

DS-40xxHSI: x should be multiple of 16, w should be multiple of 32, y and h should be multiple of 8.

DS-52xx and DS-42xx: w and h should be multiple of 16.

SetLogo is valid for encoding card only.

6.3.68 Stop Logo Display: StopLogo()

API:

int StopLogo(HANDLE hChannelHandle);

Parameters:

HANDLE hChannelHandle; Channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Stop the display of Logo.

Video Mask

6.3.69 Set Mask: SetupMask()

API:

int SetupMask(HANDLE hChannelHandle, RECT *rectList, int iAreas)

Parameters:

HANDLE	hChannelHandle;	channel handle
RECT	* rectList;	array of rectangle
int	iAreas;	number of rectangles

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Start video mask, the max number of rectangles is 32 .

The range of rectangle is PAL: (0, 0, 703, 575), NTSC: (0, 0, 703, 479), the x-coordinate of the upper-left corner and width of the rectangle must be multiple of 16, the y-coordinate of the upper-left corner and height of the rectangle must be multiple of 8.

DS-42xx card: width and height should be multiple of 16.

6.3.70 Set Mask: SetupEncoderMask ()

API:

int SetupEncoderMask (HANDLE hChannelHandle, int iAreaCount, MASK_AREA_PARAM* pAreaParam)

Parameters:

HANDLE	hChannelHandle;	channel handle
int	iAreaCount;	mask area number, 0 means to disable mask function
MASK_AREA_PARAM*	pAreaParam;	mask area parameter

```
typedef struct
{
    UINT left;          /*Upper left X-axis coordinate*/
    UINT top;           /*Upper left Y-axis coordinate */
    UINT width;         /*Width of mask area*/
    UINT height;        /*Height of mask area */
```

COLORREF color; */*Colour of mask area*/*

}MASK_AREA_PARAM;

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

It supports up to 32 mask regions, and support color settings of each mask region.

Note: Color setting of mask area (pAreaParam->color) is supported by DS-40xx & DS-43xx series card only.

6.3.71 Stop Mask: StopMask()

API:

int StopMask(HANDLE hChannelHandle)

Parameters:

HANDLE hChannelHandle; channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Stop video mask .

Overlay on the Live View Image

Remarks:

Users can add text or draw on the image surface directly under overlay viewing mode, while drawing callback APIs shall be called to achieve similar functions under the default offscreen viewing mode.

All the extra information on the live view screen will not be encoded into the video record.

Set Live Audio and Get Live Audio Level

6.3.72 Set Live Audio: SetAudioPreview()

API:

int SetAudioPreview(HANDLE hChannelHandle, BOOL bEnable);

Parameters:

HANDLE hChannelHandle; channel handle

BOOL bEnable; be enabled or not

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set audio monitor, only one channel can be monitored at the same time.

DS-40xx card requires connecting audio cable to audio input port of sound card, while DS-42xx & DS-43xx cards don't need the audio cable.

6.3.73 Get Live Audio Level: GetSoundLevel()

API:

int GetSoundLevel(HANDLE hChannelHandle)

Parameters:

HANDLE hChannelHandle; channel handle

Return:

Return the sound level of current channel .

Remarks:

When there is no audio input,maybe the return value is no0 because of the noise.

6.3.74 Start PCI Live Audio of Encode Channel: SetDirectAudioPreview()

API:

int SetDirectAudioPreview(HANDLE hChannelHandle,BOOL bEnable)

Parameters:

HANDLE hChannelHandle; channel handle

BOOL bEnable; enable or not

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Users can get DS-40xx card live audio by this API without audio cable connection. For DS-42xx & DS-43xx cards, this API achieves same function as SetAudioPreview()

CRC Checksum

6.3.75 Set CRC of Main Stream: SetChannelStreamCRC()

API:

int SetChannelStreamCRC(HANDLE hChannel,BOOL bEnable)

Parameters:

HANDLE	hChannelHandle;	channel handle
BOOL	bEnable;	indicates whether the CRC checksum should be enabled or disabled, Non0 enables the CRC checksum, 0 disables the CRC checksum

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Enable or Disable the CRC checksum on main channel stream;

To make the CRC checksum take effect on main channel stream, call this function before StartVideoCapture().

Note: this API is designed for DS-40xx series card only.

6.3.76 Set CRC of Sub Stream: SetSubChannelStreamCRC()

API:

int SetSubChannelStreamCRC (HANDLE hChannel,BOOL bEnable)

Parameters:

HANDLE	hChannelHandle;	channel handle
BOOL	bEnable;	indicates whether the CRC checksum should be enabled or disabled, Non0 enables the CRC checksum, 0 disables the CRC checksum

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Enable or disable the CRC checksum on sub channel stream;

For CRC on sub stream, please call this API before StartSubVideoCapture().

Note: this API is designed for DS-40xx series card only.

Noise Reduction

6.3.77 Set Noise Reduction: SetDeNoise()

API:

int __stdcall SetDeNoise(HANDLE hChannelHandle,UINT level)

Parameters:

HANDLE	hChannelHandle;	channel handle
UINT	level;	Noise reduction level, range from 0-3

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This API is valid for DS-43xxHFVI-E, DS-5316HFI and DS-43xxHWI-E only.

6.3.78 Set Video Codec: SetVideoCodec()

API:

int __stdcall SetVideoCodec(HANDLE hChannelHandle,USHORT type)

Parameters:

HANDLE hChannelHandle; channel handle

USHORT type; video codec type defined as below

Macro Definition:

#define VIDEO_TYPE_H264 1 ;

#define VIDEO_TYPE_MPEG4 2 ;

#define VIDEO_TYPE_MJPEG 3 ;

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

This API is valid for DS-43xx encode card only.

6.4 Decoding Card APIs

Init and Release Decoding Card

6.4.1 Initialize decoding card: HW_InitDecDevice()

API:

int HW_InitDecDevice(long *pDeviceTotal)

Parameters:

long * pDeviceTotal; [out] total decode channel number

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Initialize the decode card. If **InitDSPs()** has been called, you needn't call this.

6.4.2 Release decoding card: HW_ReleaseDecDevice()

API:

int HW_ReleaseDecDevice()

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Close the DS-40xxMDI/DS-40xxMDI+/DS-4101HDI card. It must be used when the software quits.

6.4.3 Initialize DirectDraw: HW_InitDirectDraw()

API:

int HW_InitDirectDraw(HWND hParent, COLORREF colorKey)

Parameters:

HWND	hParent;	the window handle. The display area must be in the window, and the display area coordinate must be the coordinate of the window.
COLORREF	colorKey;	the transparent color you will set. It likes clairvoyant film. The display image can only penetrate this kind of color. Other color will mask the display image. The display window must use this color. We suggest you to use the color not often used by other software. The value is DWORD 0x00rrggbb, the latter 3 bytes represent the value of r (red), g (green), b (blue) individually.

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Initialize DirectDraw.

6.4.4 Release DirectDraw: HW_ReleaseDirectDraw()

API:

int HW_ReleaseDirectDraw()

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Release DirectDraw.

Open/Close Decode Channel

6.4.5 Open Decode Channel: HW_ChannelOpen()

API:

int HW_ChannelOpen(long nChannelNum, HANDLE* phChannel)

Parameters:

long nChannelNum; the channel number to open (start from 0).

HANDLE* phChannel; [out] the channel handle.

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Open channel to get the correspond handle. All the related operation must use the handle.

6.4.6 Close Decode Channel: HW_ChannelClose()

API:

int HW_ChannelClose(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle.

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Close the channel to release the related resources.

Get Information of Decoding Card

6.4.7 Get SDK Version: HW_GetVersion()

API:

int HW_GetVersion(PHW_VERSION pVersion)

Parameters:

pVersion: the version information, the structure is as following:

typedef struct{

ULONG DspVersion, DspBuildNum;

ULONG DriverVersion, DriverBuildNum;

ULONG SDKVersion, SDKBuildNum;

}HW_VERSION, PHW_VERSION;

DspVersion and DspBuildNum: the Version and Build number of DSP.

DriverVersion and DriverBuildNum: the Version and Build number of driver.

SDKVersion and SDKBuildNum: the Version and Build number of the SDK.

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Get the SDK version information.

Live Video/Audio and Output of Decoding Card

Live Audio Settings

6.4.8 Start Live Audio: HW_SetAudioPreview()

API:

int HW_SetAudioPreview(HANDLE hChannel, BOOL bEnable)

Parameters:

HANDLE	hChannel;	channel handle.
BOOL	bEnable;	TRUE to start live audio and FALSE to close live audio

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Start or stop the live audio output. The audio connection method is the same as that of compression card. You can only get the live audio of 1 channel; and the audio of other channels will be closed automatically. To start playing live audio, API HW_ChannelOpen shall be called first, and then API HW_PlaySound shall be called to play the live audio.

Note:

From version 4.3, audio output is closed by default and SetDecoderAudioOutput() should be called first before calling HW_SetAudioPreview()

6.4.9 PCI Live Audio of Decode Channel: HW_SetDirectAudioPreview()

API:

int HW_SetDirectAudioPreview(HANDLE hChannel,BOOL bEnable)

Parameters:

HANDLE hChannel; channel handle
BOOL bEnable; enable/disable PCI live audio

Return:

If the function succeeds, the return value is 0.
If the function fails, the return value is the error code defined on section 3.

Remarks:

It doesn't require connection of audio cable.

VGA Live View Settings

6.4.10 Set Video Display Parameters: HW_SetDisplayPara()

API:

int HW_SetDisplayPara(HANDLE hChannel,DISPLAY_PARA *pPara)

Parameters:

HANDLE hChannel; The channel handle
DISPLAY_PARA *pPara; The video display parameter
typedef struct{
long bToScreen; Output to VGA, 1-enable, 0-disable
long bToVideoOut; Output to monitor, 1-enable,0-disable (reserved)
long nLeft; Display region related to hParent client
long nTop;
long nWidth;
long nHeight;
long nReserved; reserved
}DISPLAY_PARA,*PDISPLAY_PARA

Return value:

Return 0 if success, or return error code.

Remarks:

Set the video display parameters.

6.4.11 Refresh the Display Surface: HW_RefreshSurface()

API:

int HW_RefreshSurface()

Return value:

Return 0 if success, or return error code.

Remarks:

Refresh the display area. When location of the window (hParent in the 3rd API) is changed, you need refresh the overlay surface to fit for the new area.

6.4.12 Restore Overlay Surface: HW_RestoreSurface()

API:

int HW_RestoreSurface()

Return value:

Return 0 if success, or return error code.

Remarks:

The overlay surface may be engrossed by other software during using process. You need use this API to get the overlay surface again. The most convenient method in MFC is calling this API in the “OnPaint” function.

6.4.13 Clear the Overlay Surface: HW_ClearSurface()

API:

int HW_ClearSurface()

Return value:

Return 0 if success, or return error code.

Remarks:

Clear the data on the overlay surface. When the window is switching, the data of the last frame image will be remained on the overlay surface. If you do not want to display the data, you can use this API to clear them.

6.4.14 Zoom Overlay Surface: HW_ZoomOverlay()

API:

int HW_ZoomOverlay(RECT* pSrcClientRect, RECT* pDecScreenRect)

Parameters:

RECT *pSrcClientRect;

the source area, client coordinate in hParent window.

RECT *PDecScreenRect;

the destination area, screen coordinate.

Return value:

Return 0 if success, or return error code.

Remarks:

Zoom in or zoom out the certain area on the overlay surface. This is also one of methods to display in full screen. At present, there are two methods:

One is using the enlarge function of the graphic card, to enlarge the area pSrcClientRect (client area coordinate) into full screen pDecSrceenRect (screen coordinate). Please note that the pSrcClientRect must not be less that the original image size (for example, CIF image of PAL is 352*288), or the enlarged quality is bad. Please refer to the related demo source code.

The other is using the enlarge function of our card. If you set DISPLAY_PARA as appropriate value, the enlarge function of card will be adjust automatically. Please **Note:** if you use this method, when you initialize DirectDraw, hParent must be set as the whole screen, because the display area (DISPLAY_PARA) can not be greater than the hParent area.

6.4.15 Enable Deflash Function: HW_SetDecoderPostProcess()

API:

int HW_SetDecoderPostProcess(HANDLE hChannel,UINT param)

Parameters:

HANDLE	hChannel;	channel handle.
UINT	param;	bit 0: 1- enable the function to eliminate the flicker; 0- disable the function to eliminate the flicker; Other bits are reserved and should be 0.

Return:

Return 0 if success, or return error code.

Remarks:

When there is noise in still area, the image will flicker (or refresh) frequently. If enable the function to eliminate the flicker, the effect will be eliminated or reduced.

6.4.16 Set Partial Zoom: ZoomOutDecoderVideoPreview()

API:

int ZoomOutDecoderVideoPreview(HANDLE hChannel, UINT nRectIdx, BOOL bEnable, HWND hWnd, ZOOM_RECT_NORMALIZE* pZoomRect, RECT* pRect)

Parameters:

HANDLE	hChannel;	channel handle
UINT	nRectIdx;	index of zoom area 0 means display on the video live window of current encode channel after zoomed in.
BOOL	bEnable;	enable or disable the zoom function
HWND	hWnd;	window handle to display zoom area. When nRectIdx is 0, the parameter is invalid.
ZOOM_RECT_NORMALIZE*	pZoomRect;	partial zoom area relative to decode channel video live area, and its value is normalized. Please see to ZOOM_RECT_NORMALIZE
RECT*	pRect;	target area to display zoomed video, usually it is the client area of hWnd. When nRectIdx is 0, the parameter is invalid.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

One decode channel supports max 16 partial zoom areas. nRectIdx value range is [0,16]. When nRectIdx=0, it will display on current decoding live video window, hWnd and pRect are invalid now; when nRectIdx values from 1 to 16, the value means this function will effect on corresponding partial zoom area.

6.4.17 Register Draw Callback Function: HW_RegisterDrawFun()

API:

int HW_RegisterDrawFun(DWORD nport, DRAWFUN(DrawFun),
LONG nUser)

Parameters:

DWORD nport	the channel handle, the value of higher 16 bits means the index of partial zoom area (starts from 1), and if the value is 0, it means original decoded video preview. The value of lower 16 bits means the decode channel number. E.g. nport = 0x00010000, it means NO.1 partial zoom area of NO.0 decode channel; nport = 0x00000000, it means live video of NO.0 decode channel.
DRAWFUN(DrawFun)	callback function for drawing
LONG nUser	user data, reserved

Callback function

#define DRAWFUN(x) void (CALLBACK* x) (long nPort, HDC hDc, LONG nUser)

LONG nPort;	channel number
HDC hDc ;	DC
LONG nUser;	user data

Return:

Return 0 if successfully, or return error code.

Remarks:

Before calling this API to register the draw callback of one partial zoom area, please call ZoomOutDecoderVideoPreview to enable partial zoom function of the area firstly.

6.4.18 Stop Decoding Draw Callback Function: HW_StopRegisterDrawFun()

API:

int HW_StopRegisterDrawFun(DWORD nport)

Parameters:

DWORD nport;	the channel handle

Return:

Return 0 if successfully, or return error code.

Remarks:

Stop draw callback.

6.4.19 Set Vertical Sync of Partial Zoom: SetDecoderZoomOutAntiTearing()

API:

int SetDecoderZoomOutAntiTearing(HANDLE hChannel, UINT nRectIdx, BOOL bAntiTearing, DWORD reserved)

Parameters:

HANDLE	hChannel;	channel handle
UINT	nRectIdx;	index of zoom area, must not be less than 1
BOOL	bAntiTearing;	enable vertical sync or not
DWORD	reserved;	reserved

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Before calling this API to enable vertical sync of one partial zoom area, please call ZoomOutDecoderVideoPreview to enable partial zoom function of the area firstly. If call ZoomOutDecoderVideoPreview to re-enable its partial zoom function after calling the API, the vertical sync will invalid, and it requires calling the API again to enable the vertical sync.

Set Display Region of Video Output Settings

6.4.20 Set Video Standard for Video Output: SetDisplayStandard()

API:

int SetDisplayStandard(UINT nDisplayChannel,
VideoStandard_t VideoStandard)

Parameters:

UINT	nDisplayChannel;	index of the video output channel
VideoStandard_t	VideoStandard;	video standard

Return value:

Return 0 if success, or return error code.

Remarks:

Set the video standard of the video output channel.

6.4.21 Set Display Region for Video Output: SetDisplayRegion()

API:

int SetDisplayRegion(UINT nDisplayChannel, UINT nRegionCount,
REGION_PARAM *pParam,UINT nReserved)

Parameters:

UINT	nDisplayChannel;	display channel index
UINT	nRegionCount;	the total number of regions
REGION_PARAM	*pParam;	pointer to REGION_PARAM
UINT	nReserved;	reserved

```
typedef struct
{
    UINT left;
        /* specifies the x-coordinate of the upper-left corner of region */
    UINT top;
        /* specifies the y-coordinate of the upper-left corner of region */
    UINT width;
        /* specifies the width of the region */
    UINT height;
        /* specifies the height of the region */
    COLORREF color;
        /* specifies the background color of the region */
    UINT param;
        /* reserved */
}REGION_PARAM;
#define MAX_DISPLAY_REGION    16
        /* each display channel can be divided to max 16 small regions.*/
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

For each output of DS-40xxMDI/DS-40xxMDI+, the sum of display area can't be greater than 4CIF+2*QCIF, or SetDisplayRegion() will return ERR_NOT_SUPPORT, which means that the DSP doesn't have enough resourse .

ERR_INVALID_DEVICE: nDisplayChannel is overflowed.

ERR_INVALID_ARGUMENT: nRegionCount is overflowed, The parameter left and width are not multiples of 16, the parameter top and height are not multiples of 8. Other error will make the function return ERR_KERNEL.

Remarks:

Divide the display channel to multiregions. The parameter left and width must be multiples of 16, the parameter top and height must be multiples of 8.

6.4.22 Adjust the Display Position for Video Output:

SetDisplayRegionPosition()

API:

unsigned int SetDisplayRegionPosition(UINT nDisplayChannel,UINT nRegion, UINT nLeft, UINT nTop)

Parameters:

UINT	nDisplayChannel;	display Channel Index
UINT	nRegion;	region to be adjusted
UINT	nLeft;	the position after being adjusted
UINT	nTop;	the position after being adjusted

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nDisplayChannel is overflowed,the nRegion is invalid.

Other error will make the function return ERR_KERNEL.

Remarks:

Adjust the position of display region.

Note: this API is designed for DS-40xx & DS-43xx series card.

6.4.23 Fill Display Region with Image: FillDisplayRegion()

API:

unsigned int FillDisplayRegion(UINT nDisplayChannel,UINT nRegion, unsigned char *pImage)

Parameters:

UINT	nDisplayChannel;	display Channel Index
UINT	nRegion;	region to be filled
unsigned char	*pImage;	pointer to YV12(YUV420) data buffer

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nDisplayChannel is overflowed, the nRegion is invalid.

Other error will make the function return ERR_KERNEL.

Remarks:

Fill display region with picture files. The height and width of picture pointed by pImage must be same as the height and width of the current region which is set by function SetDisplayRegion(),or the picture can't be showed normally .

If the current region is displaying video, please stop it first; otherwise the picture can't be displayed.

6.4.24 Fill Display Region with Image: FillDisplayRegionEx()

API:

int FillDisplayRegionEx(UINT nDisplayChannel, UINT nRegion, unsigned char *pImageBuf, UINT nWidth, UINT nHeight)

Parameters:

UINT	nDisplayChannel;	display channel
UINT	nRegion;	region to be filled
unsigned char	*pImageBuf;	pointer to YV12(YUV420) image data
UINT	nWidth;	width of the image data to be filled
UINT	nHeight;	height of the image data to be filled

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

Remarks:

Fill display region with picture files.

For the API FillDisplayRegion, it requires the size of picture pointed by pImage same as the size of the current region which is set by function SetDisplayRegion(). The width value can't be larger than 704 pixels, and the height can't be larger than 576(PAL) or 480(NTSC).

While FillDisplayRegionEx supports zoom. If the picture is too large, it will be adjusted to the size of display region automatically. If the width of display region is larger than 704, the width of the picture will be zoomed to 704, and If the heigth of display region is larger than 576(PAL) or 480(NTSC), the height of the picture will be zoomed to 576(PAL) or 480(NTSC).

6.4.25 Clean up The Display Region: ClearDisplayRegion()

API:

int ClearDisplayRegion(UINT nDisplayChannel,UINT nRegionFlag)

Parameters:

UINT	nDisplayChannel;	display channel index
UINT	nRegionFlag;	Bit0 - Bit15 stand for region 1 to 16. If the bit is 1, the corresponding region will be cleaned up.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nDisplayChannel is overflowed, the nRegion is invalid.

Other error will make the function return ERR_KERNEL.

Remarks:

Clean up the image of display region and display the color set by SetDisplayRegion().

If the current region is displaying video, please stop displaying first otherwise the

picture can't be displayed.

Video Output Settings

6.4.26 Set Brightness for Video Output: SetDisplayVideoBrightness()

API:

SetDisplayVideoBrightness(UINT nDisplayChannel, int Brightness)

Parameters:

UINT nDisplayChannel;	display channel
Int Brightness;	brightness level, ranges from 0 to 255

Return:

Return 0 if success, or return error code.

Remarks:

Adjust the brightness of video output from matrix card or decoding card.

Decode and Control

Decoding Stream Data

6.4.27 Set Open Stream Mode HW_SetStreamOpenMode()

API:

int HW_SetStreamOpenMode(HANDLE hChannel, ULONG nMode)

Parameters:

HANDLE hChannel;	the channel handle
ULONG nMode;	the value is from 0 to 5. 0 represents no adjustment, fits for file playback (not real time stream). Set nMode as 1, 2, 3, 4, 5 to adjust the fluency and delay of the real time stream. The bigger the value is, more fluent and more delay the real time network stream is.

Return value:

Return 0 if success, or return error code.

Remarks:

Set the stream playing parameter.

6.4.28 Get Open Stream Mode HW_GetStreamOpenMode()

API:

int HW_GetStreamOpenMode(HANDLE hChannel, ULONG *pMode)

Parameters:

HANDLE	hChannel;	the channel handle.
ULONG	* pMode;	[out] the value is among 0, 1, 2, 3, 4, 5.

Return value:

Return 0 if success, or return error code.

Remarks:

Get the playing parameter of current stream.

6.4.29 Open Stream for Decoding: HW_OpenStream()

API:

int HW_OpenStream(HANDLE hChannel, PBYTE pFileHeadBuf, DWORD nSize)

Parameters:

HANDLE	hChannel;	the channel handle.
PBYTE	pFileHeadBuf;	the file header data buffer.
DWORD	nSize;	the file header size.

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

Open the stream interface (like open file).

6.4.30 Close Stream: HW_CloseStream()

API:

int HW_CloseStream(HANDLE hChannel)

Parameters:

HANDLE	hChannel;	the channel handle.

Return value:

Return 0 if success, or return error code.

Remarks:

Close the data stream.

6.4.31 Input Data for Decoding: HW_InputData()

API:

int HW_InputData(HANDLE hChannel, PBYTE pBuf, DWORD nSize)

Parameters:

HANDLE hChannel; the channel handle.
PBYTE pBuf; the buffer address.
DWORD nSize; the buffer size, which should be 4 bytes alignment

Return value:

Return nSize if successfully, or return 0.

Remarks:

You can input data after open the data stream interface.

6.4.32 Restart Decoder in Stream Mode: HW_ResetStream()

API:

int HW_ResetStream(HANDLE hChannel)

parameters:

HANDLE hChannel; the channel handle.

Return value:

Return 0 if successfully, or return error code.

Remarks:

You can restart the decoder in stream mode.

Extended Stream Decoding Mode (Video/Audio Separated)

6.4.33 Open Stream in Separation Mode: HW_OpenStreamEx()

API:

int HW_OpenStreamEx (HANDLE hChannel, PBYTE pFileHeadBuf,
DWORD nSize)

Parameters:

HANDLE hChannel; the channel handle.
PBYTE pFileHeadBuf; file header data.
DWORD nSize; the length of file header.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Open the data stream in video and audio detaching mode. You can use No. 44 and No. 45 API to input video and audio data respectively.

6.4.34 Close Stream in Seperation Mode: HW_CloseStreamEx()

API:

int HW_CloseStreamEx(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Close the data stream that is opened in detaching mode.

6.4.35 Input Video Data: HW_InputVideoData()

API:

int HW_InputVideoData(HANDLE hChannel, PBYTE pBuf, DWORD nSize)

Parameters:

HANDLE hChannel; the channel handle.
PBYTE pBuf; the buffer address.
DWORD nSize; the buffer size, which should be 4 bytes alignment

Return value:

Return nSize if success, or return 0.

Remarks:

Input video data, only after the data stream is opened (HW_OpenStreamEx).

6.4.36 Input Audio Data: HW_InputAudioData()

API:

int HW_InputAudioData(HANDLE hChannel, PBYTE pBuf, DWORD nSize)

Parameters:

HANDLE hChannel; the channel handle.
PBYTE pBuf; the buffer address.
DWORD nSize; the buffer size.

Return value:

Return nSize if success, or return 0.

Remarks:

Input audio data, only after the data stream is opened (HW_OpenStreamEx).

Decoding Record File

6.4.37 Open File for Decoding: HW_OpenFile()

API:

int HW_OpenFile(HANDLE hChannel, LPTSTR sFileName)

Parameters:

HANDLE hChannel; the channel handle.

LPTSTR sFileName; the file name.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Open file.

6.4.38 Close File: HW_CloseFile()

API:

int HW_CloseFile(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Close the opened file.

6.4.39 Set Message Informing End of File: HW_SetFileEndMsg()

API:

int HW_SetFileEndMsg(HANDLE hChannel, HWND hWnd, UINT nMsg)

Parameters:

HANDLE hChannel; the channel handle.

HWND hWnd; the window handle to receive message.

UINT nMsg; self-defined windows message.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Register the self-defined windows message. When the playing file is end, the end message will be posted to the assigned window.

Video&Audio Playback

6.4.40 Start Video Playback: HW_Play()

API:

int HW_Play(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle

Return value:

Return 0 if successfully, or return error code.

Remarks:

Start video playback.

6.4.41 Stop Video Playback: HW_Stop()

API:

int HW_Stop(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle

Return value:

Return 0 if successfully, or return error code.

Remarks:

Stop playback.

6.4.42 Open Audio Channel: HW_PlaySound()

API:

int HW_PlaySound(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle

Return value:

Return 0 if successfully, or return error code.

Remarks:

Open the channel sound, default value: Close.

6.4.43 Close Audio: HW_StopSound()

API:

int HW_StopSound(HANDLE hChannel)

Parameters:

HANDLE hChannel; the channel handle

Return value:

Return 0 if successfully, or return error code.

Remarks:

Close the channel sound.

6.4.44 Adjust Volume: HW_SetVolume()

API:

int HW_SetVolume(HANDLE hChannel, ULONG nVolume)

Parameters:

HANDLE hChannel; the channel handle

ULONG nVolume; the sound volume, from 0 to 0xffff.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Adjust the sound volume.

6.4.45 Pause: HW_Pause()

API:

int HW_Pause(HANDLE hChannel, ULONG bPause)

Parameters:

HANDLE hChannel; the channel handle

ULONG hPause; 1 means pause, 0 means continue

Return value:

Return 0 if successfully, or return error code.

Remarks:

Pause or continue playing.

Set /Get Playback Speed

6.4.46 Set Playback Speed: HW_SetSpeed()

API:

int HW_SetSpeed(HANDLE hChannel, long nSpeed)

Parameters:

HANDLE hChannel; the channel handle.

long nSpeed; the playing speed. -4: stop playing; -3: 1/8; -2: 1/4; -1: 1/2; 0: normal play; 1: two times; 2: four times; 3:

eight times; 4: the max speed it can.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Set the playing speed. For the stepforward function, please set nSpeed as -4'and then call API HW_Pause(hChannel, 0).

6.4.47 Get Playback Speed: HW_GetSpeed()

API:

int HW_GetSpeed(HANDLE hChannel, long nSpeed)

Parameters:

HANDLE	hChannel;	the channel handle
long	*nSpeed;	[out] the playing speed, from –4 to +4.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the playing speed.

Set and Get Playing Position

6.4.48 Set file position for decoding: HW_SetPlayPos()

API:

int HW_SetPlayPos(HANDLE hChannel, ULONG nPos)

Parameters:

HANDLE	hChannel;	the channel handle.
ULONG	nPos;	from 0 to 100 percent position of the file.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Set playing position.

6.4.49 Get Playing Position: HW_GetPlayPos()

API:

int HW_GetPlayPos(HANDLE hChannel, ULONG* pPos)

Parameters:

HANDLE	hChannel;	the channel handle.

ULONG* pPos [out] the current playing position, from 0 to 100 percent.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get current playing position.

Set Playing Jump

6.4.50 Set Jump Interval: HW_SetJumpInterval()

API:

int HW_SetJumpInterval(HANDLE hChannel, ULONG nSecond)

Parameters:

HANDLE hChannel; the channel handle.

ULONG nSecond; the time interval (second).

Return value:

Return 0 if successfully, or return error code.

Remarks:

Set jump time interval.

6.4.51 Start to Jump: HW_Jump()

API:

int HW_Jump(HANDLE hChannel, ULONG nDirection)

Parameters:

HANDLE hChannel; the channel handle.

ULONG nDirection; use the macro definition.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Jump forward or backward. You can use the macro definition:

JUMP_FORWARD, JUMP_BACKWARD.

Query of Decoding Time and Frame Information

Time Information

6.4.52 Get File's Total Time: HW_GetFileTime()

API:

int HW_GetFileTime(HANDLE hChannel, ULONG* pFileTime)

Parameters:

HANDLE	hChannel;	the channel handle.
ULONG	*pFileTime;	[out] file total time (ms).

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the file total time.

6.4.53 Get Current Frame's Time: HW_GetCurrentFrameTime()

API:

int HW_GetCurrentFrameTime(HANDLE hChannel, ULONG* pFrameTime)

Parameters:

HANDLE	hChannel;	the channel handle.
ULONG	pFrameTime;	[out] time of the current playing frame (ms).

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the time of the current playing frame.

6.4.54 Get File Absolute Time: HW_GetFileAbsoluteTime()

API:

int HW_GetFileAbsoluteTime(HANDLE hChannel, SYSTEMTIME *pStartTime, SYSTEMTIME *pEndTime)

Parameters:

HANDLE hChannel;	channel handle
SYSTEMTIME *pStartTime;	start time of playing file (The parameter millisecond in the SYSTEMTIME is invalid, it is always 0)
SYSTEMTIME *pEndTime;	end time of playing file (The parameter millisecond in the SYSTEMTIME is invalid, it

is always 0)

Return:

Return 0 if successfully, or return error code.

Remarks:

Get the start and end time of playing file(absolute time). The parameter millisecond in the SYSTEMTIME is invalid, it is always 0.

6.4.55 Get Current Frame's Absolute Time: HW_GetCurrentAbsoluteTime()

API:

int HW_GetCurrentAbsoluteTime(HANDLE hChannel, SYSTEMTIME *pTime)

Parameters:

HANDLE hChannel;	channel handle
SYSTEMTIME *pTime;	the current absolute time in the position where is playing back

Return:

Return 0 if successfully, or return error code.

Remarks:

Get the current absolute time in the position where is playing back(The parameter millisecond in the SYSTEMTIME is invalid, it is always 0)

6.4.56 Locate by Absolute Time: HW_LocateByAbsoluteTime()

API:

int HW_LocateByAbsoluteTime(HANDLE hChannel,SYSTEMTIME time)

Parameters:

HANDLE	hChannel;	channel handle
SYSTEMTIME	time;	the absolute time to be located

Return:

Return 0 if successfully, or return error code.

Remarks:

Locate the file by absolute time. This function takes effect after the file index is created successfully. To get the start and end absolute time, call API HW_GetFileAbsoluteTime().

Frame Information

6.4.57 Get Total Frame Number: HW_GetFileTotalFrames()

API:

int HW_GetFileTotalFrames(HANDLE hChannel, ULONG* pTotalFrames)

Parameters:

HANDLE hChannel; the channel handle.

ULONG *pTotalFrames; [out] the total frame number.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the file total frame number.

6.4.58 Get the Number of Played Frames: HW_GetPlayedFrames()

API:

int HW_GetPlayedFrames(HANDLE hChannel, ULONG *pDecVFrames)

Parameters:

HANDLE hChannel; the channel handle.

ULONG *pDecVFrames; [out] the frames number that have been played.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the frames number that have been played.

6.4.59 Get Current Frame Rate: HW_GetCurrentFrameRate()

API:

int HW_GetCurrentFrameRate(HANDLE hChannel, ULONG* pFrameRate)

Parameters:

HANDLE hChannel; the channel handle.

ULONG *pFrameRate; [out] the frame rate.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the current playing frame rate.

6.4.60 Get Current Frame Number: HW_GetCurrentFrameNum()

API:

int HW_GetCurrentFrameNum(HANDLE hChannel, ULONG* pFrameNum)

Parameters:

HANDLE hChannel; the channel handle.

ULONG *pFrameNum; [out] the frame number.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get the current playing frame number.

6.4.61 Locate By Frame Number: HW_LocateByFrameNumber()

API:

int HW_LocateByFrameNumber(HANDLE hChannel,UINT frmNum)

Parameters:

HANDLE	hChannel;	channel handle
UINT	frmNum;	frame number

Return:

Return 0 if successfully, or return error code.

Remarks:

Locate the file by frame number. This function takes effect after the file index is created successfully. Please call API HW_GetFileTotalFrames() to get the total frame number of the file.

Capture Datas

Capture Image

Note: Image capture is a special feature for DS-40xxMDI series decoding card

6.4.62 Get YV12 Image: HW_GetYV12Image()

API:

int HW_GetYV12Image(HANDLE hChannel, PBYTE pBuffer, ULONG nSize)

Parameters:

HANDLE	hChannel;	the channel handle.
PBYTE	pBuffer;	the buffer to save data. The size of buffer must be greater than or equal to (*pWidth)*(*pHeight)*3/2. In YV12 format, each pixel need 3/2 byte.
ULONG	nSize;	the size of the buffer.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Capture current displaying image, YV12 format.

6.4.63 Convert YV12 to Bmp File: HW_ConvertToBmpFile()

API:

int HW_ConvertToBmpFile(BYTE *pBuf, ULONG nSize, ULONG nWidth, ULONG nHeight, char *sFileName, ULON nReserved)

Parameters:

BYTE	* pBuf;	the buffer storing YV12 image.
ULONG	nSize;	the size of buffer.
ULONG	nWidth;	width of the image.
ULONG	nHeight;	height of the image.
char	* sFileName;	the bmp file name.
ULONG	nReserved;	reserved.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Convert the YV12 image into bitmap.

Record

6.4.64 Capture Stream and Save to File: HW_StartCapFile()

API:

int HW_StartCapFile(HANDLE hChannel, LPTSTR sFileName)

Parameters:

HANDLE	hChannel;	the channel handle.
LPTSTR	sFileName;	the file name to record the stream.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Capture the current data stream and save it into file.

6.4.65 Stop Capturing: HW_StopCapFile()

API:

int HW_StopCapFile(HANDLE hChannel)

Parameters:

HANDLE	hChannel;	the channel handle.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Stop capturing data stream.

6.4.66 Get Image Size: HW_GetPictureSize()

API:

int HW_GetPictureSize(HANDLE hChannel, ULONG* pWidth, ULONG* pHeight)

Parameters:

HANDLE	hChannel;	the channel handle.
ULONG	* pWidth;	[out] the width of the original image.
ULONG	* pHeight;	[out] the height of the original image.

Return value:

Return 0 if successfully, or return error code.

Remarks:

Get size of the original image in the current stream.

Capture YUV420 Data of Decode channel

Capturing original (YUV) data is desiged for DS-40xxMDI and DS-4304HDI-E.

Original Data Callback of Decode Channel

6.4.67 Register YUV420 Callback RegisterDecoderVideoCaptureCallback()

API:

```
int RegisterDecoderVideoCaptureCallback(
        DECODER_VIDEO_CAPTURE_CALLBACK
        DecoderVideoCaptureCallback,void *context)
```

Parameters:

DECODER_VIDEO_CAPTURE_CALLBACK DecoderVideoCaptureCallback;	Callback function
void *context;	context when this function is called

```
typedef void (*DECODER_VIDEO_CAPTURE_CALLBACK)
(UINT nChannelNumber, void *DataBuf, UINT width, UINT height,
 UINT nFrameNum, UINT nFrameTime, SYSTEMTIME *pFrameAbsoluteTime,
 void *context)
```

UINT	nChannelNumber;	handle of decode channel
void	*DataBuf;	buffer address
UINT	width;	width of image
UINT	height;	height of image
UINT	nFrameNum;	frame number of current frame grabbed

UINT nFrameTime ; relative time of current frame (millisecond)
SYSTEMTIME *pFrameAbsoluteTime;
absolute time of current frame
void *context; context.

Return:

Return 0 if successfully, or return error code.

Remarks:

Register a callback function to get decoded data.

6.4.68 Start Capturing Decoded Data: HW_SetDecoderVideoCapture()

API:

int HW_SetDecoderVideoCapture(HANDLE hChannel,BOOL bStart,
UINT param)

Parameters:

HANDLE hChannel; the decode channel handle
BOOL bStart; enable or disable
UINT param; reserved

Return:

Return 0 if successfully, or return error code.

Remarks:

Start capturing decoded data in YUV420 of decode channel

Original Data Callback of Display Channel

This callback function is desiged for DS-40xxMDI and DS-4304HDI-E.

6.4.69 Register YUV420 Callback: RegisterDisplayVideoCaptureCallback()

API:

int RegisterDisplayVideoCaptureCallback(IMAGE_STREAM_CALLBACK
DisplayVideoCaptureCallback,void *context)

Parameters:

IMAGE_STREAM_CALLBACK DisplayVideoCaptureCallback;
callback function
void *context; context

Callback function Explanation:

void (*IMAGE_STREAM_CALLBACK)(UINT channelNumber ,void *context)
UINT channelNumber; Display Channel Index
void *context; context provided when calling this function

Return:

Return 0 if successfully, or return error code.

Remarks:

Register callback function of get original image stream that is output to monitor, The application can get realtime YUV420 data (without any compression) .

6.4.70 Start Capturing YUV420 Data: SetDisplayVideoCapture()

API:

int SetDisplayVideoCapture(UINT nDisplayChannel, BOOL bStart,
UINT fps,UINT width,UINT height,unsigned char *imageBuffer)

Parameters:

UINT nDisplayChannel;	display channel
BOOL bStart;	TRUE/ Start Capture, FALSE/Stop Capture
UINT fps;	frame rate
UINT width;	image width, at present, it must be 704
UINT height;	image height, at present ,it must be 576(PAL) or 480(NTSC)
unsigned char *imageBuffer;	address of image buffer

Return:

Return 0 if successfully, or return error code.

Remarks:

Call this function to start or stop getting the original image stream that is output to monitor, the speed depends on the frequency of host's CPU.

File Index

6.4.71 Create File Index: HW_SetFileRef()

API:

int HW_SetFileRef(HANDLE hChannel, BOOL bEnable,
FILE_REF_DONE_CALLBACK FileRefDoneCallback)

Parameters:

HANDLE hChannel;	channel handle
BOOL bEnable;	enable to create file index or not
FILE_REF_DONE_CALLBACK FileRefDoneCallback;	the callback function to be called after the file index is createdsuccesfully

typedef void (*FILE_REF_DONE_CALLBACK)(UINT nChannel, UINT nSize)

UINT nChannel;	channel index
UINT nSize;	the size of file index.

Return:

Return 0 if successfully, or return error code.

Remarks:

Set file index.

6.4.72 Export File Index: HW_ExportFileRef()

API:

int HW_ExportFileRef(HANDLE hChannel,char *pBuffer,UINT nSize)

Parameters:

HANDLE hChannel;	channel handle
char * pBuffer;	the buffer to export file index
UINT nSiz;	buffer size, must be greater than the size of file index. The size of file index can be gotten from the callback function of HW_SetFileRef()

Return:

Return 0 if successfully, or return error code.

Remarks:

Export file index

6.4.73 Import File Index: HW_ImportFileRef()

API:

int HW_ImportFileRef(HANDLE hChannel,char *pBuffer,UINT nSize)

Parameters:

HANDLE hChannel	channel handle
char * pBuffer	the buffer to import file index
UINT nSize	buffer size

Return:

Return 0 if successfully, or return error code.

Remarks:

Import file index .

To import file index, the application software should stop creating file index in function HW_SetFileRef(), then import file index after calling function HW_OpenFile().

OSD

6.4.74 Set OSD Display Mode: SetDecoderOsdDisplayMode()

API:

int SetDecoderOsdDisplayMode(HANDLE hChannel, BOOL bEnableOsd, UINT nLuminance, UINT nFlash, COLORREF CharacterColor, COLORREF BackGroundColor, UINT nTranslucentMode, BOOL bAutoAdjustLum, UINT timeMode, int *param, int nLineCount, UINT **Format)

Parameters:

HANDLE hChannel;	channel handle
BOOL bEnableOsd;	enable or disable OSD
UINT nLuminance;	OSD brightness, value range: [0,255]
UINT nFlash;	OSD flash parameter. Bit0-7: stopping seconds, Bit8-15: display seconds. E.g. flash= (2<<8)|1 means Logo display for 2 seconds, stopping for 1s.
COLORREF CharacterColor;	colour of OSD, supports RGB format. If its value is '-1', it is of normal mode, and OSD is white.
COLORREF BackGroundColor;	background colour of OSD, supports RGB format. If its value is '-1', it is of normal mode, and the background is transparent.
UINT nTranslucentMode;	set to be translucent or not. 0- both foreground and background are not transparent, 1- foreground translucent and background not transparent, 2- foreground not transparent and background translucent, 3- both foreground and background translucent */
BOOL bAutoAdjustLum;	adjust brightness automatically or not. When set to adjust automatically, the defined brightness value is invalid
UINT timeMode;	0- system time, 1- decoding time
int param;	int type array, to set horizontal and vertical magnification of every OSD line. Bit0-7: vertical magnification, Bit8-15: horizontal magnification
int nLineCount;	lines of OSD display, max 8 lines
UINT ** Format;	describle the position of overlay and pattern of order.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

6.4.75 Set OSD Display Mode: SetDisplayOsdDisplayMode()

API:

SetDisplayOsdDisplayMode(UINT nDisplayChannel, BOOL bEnableOsd, UINT nLuminance, UINT nFlash, COLORREF CharacterColor, COLORREF BackGroundColor, UINT nTranslucentMode, BOOL bAutoAdjustLum, int *param, int nLineCount, UINT **Format)

Parameters:

UINT	nDisplayChannel;	display channel number
BOOL	bEnableOsd;	enable or disable OSD
UINT	nLuminance;	OSD brightness, value range: [0,255]
UINT	nFlash;	OSD flash parameter. Bit0-7: stopping seconds, Bit8-15: display seconds. E.g. flash= (2<<8)|1 means Logo display for 2 seconds, stopping for 1s.
COLORREF	CharacterColor;	colour of OSD, supports RGB format. If its value is '-1', it is of normal mode, and OSD is white.
COLORREF	BackGroundColor;	background colour of OSD, supports RGB format. If its value is '-1', it is of normal mode, and the background is transparent.
UINT	nTranslucentMode;	set to be translucent or not. 0- both foreground and background are not transparent, 1- foreground translucent and background not transparent, 2- foreground not transparent and background translucent, 3- both foreground and background translucent
BOOL	bAutoAdjustLum;	adjust brightness automatically or not. When set to adjust automatically, the defined brightness value is invalid
int	param;	int type array, to set horizontal and vertical magnification of every OSD line. Bit0-7: vertical magnification, Bit8-15: horizontal magnification
int	nLineCount;	lines of OSD display, max 8 lines
UINT	** Format;	describle the position of overlay and pattern of order.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

6.4.76 Enable/Disable OSD: SetOsd()

API:

int SetOsd(HANDLE hChannelHandle, BOOL Enable);

Parameters:

HANDLE	hChannelHandle;	channel handle
BOOL	Enable;	be enabled or not

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set OSD (Overlay String Display) mode. It can make the current system time(such as year , month , day ,hour ,minute and second) or custom string overlay with the real-time active video window, translucent processing is also permitted

Logo

6.4.77 Convert 24bit BMP to YUV422: LoadYUVFromBmpFile()

API:

int LoadYUVFromBmpFile(char *FileName, unsigned char *yuv, int BufLen, int *Width, int *Height);

Parameters:

char	*FileName;	file Nname
unsigned char	*yuv;	address of yuv buffer
int	BufLen;	the size of yuv buffer
int	*Width;	[out]the width of yuv picture
int	*Height;	[out]the height of yuv picture

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Convert the 24 bit BMP file to YUV422 data .The height and width of BMP file must be multiple of 16 pixels, the max size is 128*128 pixels. v4.3 SDK supports 256*128 pixels. From v6.0 SDK, DS-40xx series, DS-42xxHFVI and DS-43xx card support 4CIF Logo, and DS-4216HCI card support CIF.

6.4.78 Setup Logo for the Display Channel: SetDisplayLogo()

API:

int SetDisplayLogo(UINT nDisplayChannel, int x, int y, int w, int h, unsigned char *yuv)

Parameters:

UINT	nDisplayChannel;	display channel number
int	x,y,w,h;	coordinate of Logo bmp
unsigned char	*YUV;	image data of UYVY(yuv422) format

Remarks:

x and w should be multiple of 16, y and h should be multiple of 8.

6.4.79 Set Logo Mode for Display Channel: SetDisplayLogoDisplayMode()

API:

int SetDisplayLogoDisplayMode(UINT nDisplayChannel, COLORREF ColorKey, BOOL Translucent, int Twinkle)

Parameters:

HANDLE	nDisplayChannel;	display channel handle
COLORREF	ColorKey;	key color of Logo image
BOOL	Translucent;	whether translucent or not
int	Twinkle;	time setting of twinkle. It is formed as hex 0xXXYY, and XX is display time and YY is the time of stopping display. Both times is measured as second. When XX and YY are all 0 it can display normally;

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set Logo on the display channel of DS-40xxMDI/MDI+, or DS-4101HDI.

6.4.80 Stop Logo of Display Channel: StopDisplayLogo()

API:

int StopDisplayLogo(UINT nDisplayChannel)

Parameters:

UINT	nDisplayChannel;	display channel handle

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set Logo on the display channel of DS-40xxMDI/MDI+, or DS-4101HDI.

Output Configuration of HD Decoding Card

6.4.81 Set Video Output Mode: SetDisplayVideoMode()

API:

int SetDisplayVideoMode(UINT nDisplayChannel,
DISPLAY_ FORMAT displayFormat, DISPLAY_RESOLUTION displayResolution)

Parameters:

UINT nDisplayChannel; output channel
DISPLAY_ FORMAT displayFormatoutput; interface format
DISPLAY_RESOLUTION displayResolution; output resolution and frame rate

Return:

Return 0 if successfully, or return error code.

Remarks:

This API supports DS-4101HDI and DS-43xx series card.

Output Resolution and Frame Rate:

```
typedef enum
{
 DISPLAY_RESOLUTION_INVALID              = 0x00000000,
 DISPLAY_RESOLUTION_D1                   = 0x00000001,
 DISPLAY_RESOLUTION_XGA_60HZ             = 0x00000002,
 DISPLAY_RESOLUTION_SXGA_60HZ            = 0x00000004,
 DISPLAY_RESOLUTION_SXGA_960_60HZ        = 0x00000008,
 DISPLAY_RESOLUTION_720P_50HZ            = 0x00000010,
 DISPLAY_RESOLUTION_720P_60HZ            = 0x00000020,
 DISPLAY_RESOLUTION_1080I_50HZ           = 0x00000040,
 DISPLAY_RESOLUTION_1080I_60HZ           = 0x00000080,
 DISPLAY_RESOLUTION_1080P_24HZ           = 0x00000100,
 DISPLAY_RESOLUTION_1080P_25HZ           = 0x00000200,
 DISPLAY_RESOLUTION_1080P_30HZ           = 0x00000400,
 DISPLAY_RESOLUTION_1080P_50HZ           = 0x00000800,
 DISPLAY_RESOLUTION_1080P_60HZ           = 0x00001000,
 DISPLAY_RESOLUTION_UXGA_30HZ            = 0x00002000,
 DISPLAY_RESOLUTION_UXGA_60HZ            = 0x00004000
}DISPLAY_RESOLUTION;
```

output mode:

```
typedef enum
{
 DISPLAY_FORMAT_INVALID        = 0x00000000,
```

```
 DISPLAY_FORMAT_CVBS        = 0x00000001,
 DISPLAY_FORMAT_DVI         = 0x00000002,
 DISPLAY_FORMAT_VGA         = 0x00000004,
DISPLAY_FORMAT_HDMI         = 0x00000008,
 DISPLAY_FORMAT_YPbPr       = 0x00000010
}DISPLAY_FORMAT;
```

6.4.82 Get Video Output Mode: GetDisplayVideoMode()

API:

int GetDisplayVideoMode(UINT nDisplayChannel, DWORD *dwDisplayFormat, DWORD *dwDisplayResolution)

Parameters:

UINT	nDisplayChannel;	output channel
DWORD	*displayFormat;	output interface format
DWORD	*displayResolution;	output resolution and refresh rate

Return:

Return 0 if successfully, or return error code.

Remarks:

This API only supports DS-4101HDI, DS-43xxHFHI-E and DS-4304HDI-E.

6.4.83 Get Supported Display Resolution: GetDisplayFormatCapability()

API:

int GetDisplayFormatCapability (UINT nDisplayChannel, DISPLAY_FORMAT displayFormat, DWORD *dwResolutionCapabiltiy);

Parameters:

UINT nDisplayChannel;	output channel
DISPLAY_FORMAT displayFormat;	output mode
DWORD *dwResolutionCapabiltiy;	return resolution supported by the mode, represented by bit. See to DISPLAY_RESOLUTION for detail definition.

Remark:

This API only supports DS-4101HDI and DS-43xx series card.

DISPLAY_FORMAT_CVBS support resolution:

DISPLAY_RESOLUTION_D1

DISPLAY_FORMAT_VGA support resolution:

DISPLAY_RESOLUTION_XGA_60HZ;
DISPLAY_RESOLUTION_SXGA_60HZ;
DISPLAY_RESOLUTION_SXGA_960_60HZ;
DISPLAY_RESOLUTION_720P_60HZ;

DISPLAY_FORMAT_DVI, DISPLAY_FORMAT_HDMI support resolution:

DISPLAY_RESOLUTION_XGA_60HZ;
DISPLAY_RESOLUTION_SXGA_60HZ;
DISPLAY_RESOLUTION_SXGA_960_60HZ;
DISPLAY_RESOLUTION_720P_50HZ;
DISPLAY_RESOLUTION_720P_60HZ;
DISPLAY_RESOLUTION_1080I_50HZ;
DISPLAY_RESOLUTION_1080I_60HZ;
DISPLAY_RESOLUTION_1080P_24HZ;
DISPLAY_RESOLUTION_1080P_25HZ;
DISPLAY_RESOLUTION_1080P_30HZ
DISPLAY_RESOLUTION_1080P_50HZ
DISPLAY_RESOLUTION_1080P_60HZ
DISPLAY_RESOLUTION_UXGA_30HZ
DISPLAY_RESOLUTION_UXGA_60HZ

DISPLAY_FORMAT_YPbPr support resolution:

DISPLAY_RESOLUTION_720P_60HZ;
DISPLAY_RESOLUTION_1080I_60HZ

6.5 Video & Audio Output Configuration

Video and Audio output is supported by all the HIKVISION decode cards and certain models of HIKVISION compression cards, which contains “V” in their model names, e.g. DS-4216HFVI, DS-43xxHCVI-E, DS-43xxHFVI, etc.

For the HIKVISION cards which support video & audio output, users can output live data from 1 or multiple video & audio input channel(s) to monitors and speakers, using the video/audio output adapter in the card package.

For HIKVISION decoding card, users can output the live view data from compression cards plugged in the same PC or the decoded video and audio data via the ‘Video Out’ and ‘Audio Out’ on the card.

The output mode can be devided into **Internal Output** mode and **External Output** mode, distinguished by the difference of output source - output channel relation.

Internal Output

The output source* shall be from the same DSP as the output (display) channel.

External Output

The output source* is not required to be from the same DSP as the output (display) channel.

** The output source stands for the video or audio data that needs to be output and displayed on the monitors or speakers. It can be either encode channel from compression card or decode channel from decoding cards.*

Note: For V6.0 SDK, when the encode channel of DS-42xx is output by the output channel of other card, e.g. DS-40xxMDI/MDI+; or when the encode/decode channel of other card is output by the output channel of DS-42xx, then it only supports **single-screen** display mode.

For the DS-43xx series card with PCI-E interface, the capability of ‘output from other card’ depends on the PC motherboard. If the external data transmission is supported by the PCI-E interfaces that installed with the video/audio input and output card, then the ‘output from other card’ function works; otherwise this function is not supported.

Model	DS-42xxHFVI	DS-40xxMDI/MDI+	DS-4101HDI	DS-43xxHFVI-E/ DS-43xxHCVI-E/ DS-43xxHWI-E
Video Output Capability	V5.1 SDK: 1-ch single-screen internal video output V6.x SDK: 1-ch single-screen internal or external output, and only supports output once.	Up to xx-ch 16-screen-split* internal or external video output *When the encode channel of DS-42xx is output by DS-40xxMDI/MDI+, it only supports **single-screen**.	Up to xx-ch 16-screen-split* internal or external video output *When the encode channel of DS-42xx is output by DS-4101HDI, it only supports **single-screen**.	1-ch internal or external video output, up to 16 screen-split display
Video Source	Encode channel of DS-40xx /DS-42xx, or decode channel of DS-40xxMDI/MDI+ or DS-4101HDI	Encode channel of DS-40xx or DS-42xx, or decode channel of DS-40xxMDI/ MDI+ or DS-4101HDI	Encode channel of DS-40xx or DS-42xx, or decode channel of DS-40xxMDI/ MDI+ or DS-4101HDI	Encode channel or decode channel of DS-43xx series card
API	**SetEncoderVideoExtOutput()** **SetDecoderVideoExtOutput()**	**SetEncoderVideoExtOutput()** **SetDecoderVideoOutput()** **SetDecoderVideoExtOutput()**	**SetEncoderVideoExtOutput()** **SetDecoderVideoOutput()** **SetDecoderVideoExtOutput()**	**SetEncoderVideoExtOutput()** **SetDecoderVideoExtOutput()**
Audio Output Capability	Supports 1-ch internal or external output	Supports **xx-**ch internal or external output	Supports 1-ch internal output	Supports 1-ch internal or external output
Audio Source	Encode channel of DS-40xx /DS-42xx, or decode channel of DS-40xxMDI/MDI+ or DS-4101HDI	Encode channel of DS-40xx or DS-42xx, or decode channel of DS-40xxMDI/ MDI+ or DS-4101HDI	Encode channel of DS-40xx or DS-42xx, or decode channel of DS-40xxMDI/ MDI+ or DS-4101HDI	Encode channel or decode channel of DS-43xx series card
API	**SetEncoderAudioOutput()** **SetEncoderAudioExtOutput()** **SetDecoderAudioExtOutput()**	**SetEncoderAudioExtOutput()** **SetDecoderAudioOutput()** **SetDecoderAudioExtOutput()**	**SetEncoderAudioExtOutput()** **SetDecoderAudioOutput()** **SetDecoderAudioExtOutput()**	**SetEncoderAudioOutput()** **SetEncoderAudioExtOutput()** **SetDecoderAudioExtOutput()**

Model	DS-43xxHFHI-E	DS-4308MDI-E	DS-4304HDI-E
Video Output Capability	1-ch HDMI output, up to 16 screen-split display	8-ch BNC output, up to 16 screen-split display	4-ch HDMI output, up to 16 screen-split display
Video Source	Encode channel of DS-43xx series card, or decode channel of DS-43xx series card	Encode channel of DS-43xx series card, or decode channel of DS-43xx series card	Encode channel of DS-43xx series card, or decode channel of DS-43xx series card
API	**SetEncoderVideoExtOutput()** **SetDecoderVideoExtOutput()**	**SetEncoderVideoExtOutput()** **SetDecoderVideoOutput()** **SetDecoderVideoExtOutput()**	**SetEncoderVideoExtOutput()** **SetDecoderVideoOutput()** **SetDecoderVideoExtOutput()**
Audio Output Capability	1-ch HDMI output	8-ch BNC output	4-ch HDMI output
Audio Source	Encode channel of DS-43xx series card, or decode channel of DS-43xx series card	Encode channel of DS-43xx series card, or decode channel of DS-43xx series card	Encode channel of DS-43xx series card, or decode channel of DS-43xx series card
API	**SetEncoderAudioOutput()** **SetEncoderAudioExtOutput()** **SetDecoderAudioExtOutput()**	**SetEncoderAudioExtOutput()** **SetDecoderAudioOutput()** **SetDecoderAudioExtOutput()**	**SetEncoderAudioExtOutput()** **SetDecoderAudioOutput()** **SetDecoderAudioExtOutput()**

6.5.1 Internal Audio Output of Decode Channel: SetDecoderAudioOutput()

API:

int SetDecoderAudioOutput(UINT nDecodeChannel, BOOL bOpen, UINT nOutputChannel)

Parameters:

UINT	nDecodeChannel;	Decode channel index
BOOL	bOpen;	Enable/ disable
UINT	nOutputChannel;	The index of display channel

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nDecodeChannel is overflowed, or an incorrect nOutputChanne number; other errors will cause the function to return ERR_KERNEL.

Remarks:

Set audio output. Output the decoded audio of channel **nDecodeChannel** to the display channel **nOutputChannel** on the decoding card.

If there is already audio output from another decode channel on this **nOutputchannel**, the SDK will automatically stop this output first.

6.5.2 External Audio Output of Decode Channel: SetDecoderAudioExtOutput()

API:

int SetDecoderAudioExtOutput(UINT nDecodeChannel, UINT nPort, BOOL bOpen, UINT nOutChannel, UINT nReserved);

Parameters:

UINT nDecodeChannel	Decode channel index number;
UINT nPort	0 or 1, each audio channel can be output twice.
BOOL bOpen	Enable/ disable
UINT nOutChannel	The index of display channel, start from 0
UINT nReserved	Reserved.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code.

Remarks:

This API is used to set audio ouput, and it supports to set No.nPort audio data of decode channel No. nDecodeChannel to output from any No.nOutChannel display channel in the system.

6.5.3 Internal Audio Output of Encode Channel: SetEncoderAudioOutput()

API:

int SetEncoderAudioOutput(UINT nEncodeChannel, BOOL bEnable,UINT nOutputChannel)

Parameters:

UINT	nEncodeChannel;	The index of encode channel
BOOL	bEnable;	Enable/ disable
UINT	nOutputChannel;	Display channel, should be set as 0

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Note: For DS-42xx & DS-43xx series card only.

6.5.4 External Audio Output of Encode Channel: SetEncoderAudioExtOutput()

API:

int __stdcall SetEncoderAudioExtOutput(UINT nEncodeChannel,
UINT nPort,BOOL bOpen,UINT nOutChannel,UINT nReserved);

Parameters:

UINT nEncodeChannel;	The index of encode channel
UINT nPort;	0 or 1, every encode channel supports twice of audio output.
BOOL bOpen;	Enable/ disable
UINT nOutChannel;	Display channel, should be set as 0.
UINT nReserved;	Reserved.

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

6.5.5 Internal Video Output of Decode Channel: SetDecoderVideoOutput()

API:

unsigned int SetDecoderVideoOutput(UINT nDecodeChannel,UINT nPort,
BOOL bOpen, UINT nDisplayChannel,UINT nDisplayRegion,UINT nReserved)

Parameters:

UINT	nDecodeChannel;	The index of decode channel
UINT	nPort;	Output port of decode channel (0 or 1)
BOOL	bOpen;	Enable/ disable
UINT	nDisplayChannel;	Display channel index, starts from 0

UINT	nDisplayRegion;	Display region of current display channel
UINT	nReserved;	Reserved

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nDecodeChannel is overflow,
nPort is greater than 1,
nDisplayChannel is overflow:
nDisplayRegion is invalid .

Other error will make the function return ERR_KERNEL.

Remarks:

Set video output of decode channel. Output the video of channel **nDecodeChannel** to the region **nDisplayRegion** of display channel **nDisplayChannel**. The **nDisplayChannel** shall be in the same DSP as the **nDecodeChannel**.

6.5.6 External Video Output of Decode Channel:

SetDecoderVideoExtOutput()

API:

unsigned int SetDecoderVideoExtOutput(UINT nDecodeChannel,UINT nPort, BOOL bOpen, UINT nDisplayChannel, UINT nDisplayRegion, UINT nReserved)

Parameters:

UINT	nDecodeChannel;	Decode channel index
UINT	nPort;	Output port of decode channel (0 or 1)
BOOL	bOpen;	Enable/ disable
UINT	nDisplayChannel;	Display channel index, starts from 0
UINT	nDisplayRegion;	Display region of current display channel
UINT	nReserved;	Reserved

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nDecodeChannel is overflow, nPort is greater than 1, nDisplayChannel is overflow, nDisplayRegion is invalid .Other error will make the function return ERR_KERNEL.

Remarks:

Set external video output of decode channel. Output the video of channel **nDecodeChannel** to the region nDisplayRegion of display channel **nDisplayChannel** (start from 0).

6.5.7 Video Output of Encode Channel: SetEncoderVideoExtOutput()

API:

unsigned int SetEncoderVideoExtOutput(UINT nEncodeChannel,UINT nPort, BOOL bOpen,UINT nDisplayChannel,UINT nDisplayRegion,UINT nFrameRate)

Parameters:

UINT	nEncodeChannel;	Index of encode channel
UINT	nPort;	Output port of encode channel (must be 0 or 1)
BOOL	bOpen;	Open/Close
UINT	nDisplayChannel;	Display channel index, starts from 0
UINT	nDisplayRegion;	Display region of current display channel
UINT	nFrameRate;	Frame Rate

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

ERR_INVALID_DEVICE: nEncodeChannel is overflowed,
nPort is greater than 1,
nDisplayChannel is overflowed,
nDisplayRegion is invalid.

Other error will make the function return ERR_KERNEL

Remarks:

Set video output of encode channel (matrix output). Output the live video of channel nEncodeChannel to the region nDisplayRegion of display channel nDisplayChannel .The parameter nPort must be 0 or 1. (Each encode channel supports two video output at the same time maximally).If the parameter bOpen is equal to 0, and then nDisplayChannel and nDisplayRegion are meaningless.

This API is for combination usage of DS-40xxHCI and DS-40xxMDI card, or video output of DS-42xxHFVI, DS-43xx series card. But DS-42xxHFVI card only support output single channel and nPort should be 0.

Note: Cross-card video output of DS-43xx series card depends on the PC motherboard compatibility.

6.6 Get Channel Capability Set

Intelligent Capability:

```
typedef enum _VCA_CAPABILITY_
{
    NONE_CAPABILITY          = 0x0,    //support no intelligent ability
    BEHAVIOR_CAPABILITY      = 0x1,    //support behavior analysis
    PLATE_CAPABILITY         = 0x2,    //support license plate recognition
    FACE_CAPABILITY          = 0x4     //support face detection
```

```
}VCA_CAPABILITY;
```

Type of Intelligent Channel:

```
typedef enum _VCA_CHANNEL_TYPE_
{
    NORMAL_CHAN                 = 0,   //channel in unintelligent card
    VCA_CHAN_ON_VCA_BOARD   = 1,   //intelligent channel in intelligent card
    NORMAL_CHAN_ON_VCA_BOARD = 2,
                                       //unintelligent channel in intelligent card
}VCA_CHANNEL_TYPE;
```

Channel Type:

```
typedef enum _CHANNEL_TYPE_
{
    ENCODE_CHANNEL      = 1,      //encode channel
    DECODE_CHANNEL      = 2,      //decode channel
    DISPLAY_CHANNEL     = 3,      //display channel
}CHANNEL_TYPE;
```

DEINTERLACE Mode:

```
typedef enum _DEINTERLACE_MODE_
{
    NO_DEINTERLACE          = 0x0,      //don't support DEINTERLACE
    DEINTERLACE_MODE1       = 0x1,      //Mode 1
    DEINTERLACE_MODE2       = 0x2,      //Mode 2
    DEINTERLACE_MODE3       = 0x4,      //Mode 3
} DEINTERLACE_MODE;
```

Live audio:

```
typedef enum   _AUDIO_PREVIEW_
{
    AUDIO_PREVIEW_NOT_SUPPORT     =0x00000000,
                    //don't support live audio
    AUDIO_PREVIEW_HARD       =0x00000001,
                    //live audio with connecting audio cable
    AUDIO_PREVIEW_SOFT       =0x00000002,
                    //live audio in PCI mode, and without connecting audio cable
} AUIDO_PREVIEW;
```

Image Resolution:

```
typedef enum   _CAP_PICTURE_FORMAT_
{
   CAP_PICTURE_FORMAT_INVALID          = 0x00000000,
   CAP_PICTURE_FORMAT_QCIF             = 0x00000001,
```

```
                    //QCIF. The definition is only used in capability set.
    CAP_PICTURE_FORMAT_CIF              = 0x00000002,
                    //CIF. The definition is only used in capability set.
    CAP_PICTURE_FORMAT_2CIF             = 0x00000004,
                    //2CIF. The definition is only used in capability set.
    CAP_PICTURE_FORMAT_DCIF             = 0x00000008,
                    //DCIF. The definition is only used in capability set.
     CAP_PICTURE_FORMAT_4CIF            = 0x00000010,
                    //4CIF. The definition is only used in capability set.
     CAP_PICTURE_FORMAT_VGA             = 0x00000020,
                    //VGA. The definition is only used in capability set.
}CAP_PICTURE_FORMAT;
```

Image Resolution (extended):

```
typedef enum    _CAP_PICTURE_FORMAT_EX_
{
    CAP_PICTURE_FORMAT_SVGA         = 0x00000001,
        /* 800*600 resolution, the definition is only used in capability set */
    CAP_PICTURE_FORMAT_XGA          = 0x00000002,
        /* 1024*768 resolution, the definition is only used in capability set */
    CAP_PICTURE_FORMAT_SXGA         = 0x00000004,
        /* 1280*1024 resolution, the definition is only used in capability set */
    CAP_PICTURE_FORMAT_SXGA_960   = 0x00000008,
        /* 1280*960 resolution, the definition is only used in capability set */
    CAP_PICTURE_FORMAT_720          = 0x00000010,
        /* 1280*720 resolution, the definition is only used in capability set */
    CAP_PICTURE_FORMAT_1080         = 0x00000020,
        /* 1920*1080 resolution, the definition is only used in capability set */
    CAP_PICTURE_FORMAT_UXGA         = 0x00000040
        /* 1600*1200 resolution, the definition is only used in capability set */
}CAP_PICTURE_FORMAT_EX;
```

Channel Capacity Set:

```
typedef struct _CHANNEL_CAPABILITY_EX_
{
    BOARD_TYPE_DS boardType;
        /* type of the card which the channel belongs to */
    CHANNEL_TYPE    channelType;
        /* channel type */
    VCA_CHANNEL_TYPE vcaChanType;
        /* intelligent channel type */
    DWORD       dwVcaCapability;
        /* intelligent Capability, see to VCA_CAPABILITY */
    DWORD       dwStreamPackType;
```

```
        /* the stream packaging format supported by the main channel.
           It is invalid and reserved, and set to 0. */
    DWORD dwSubStreamPackType;
        /* the stream packaging format supported by the Subchannel.
           It is invalid and reserved, and set to 0. */
    DWORD     dwVideoCodec;
        /* video encode algorithm supported by the main channel, not in use now.
           It is invalid and reserved, and set to 0. */
    DWORD       dwSubVideoCodec;
        /* video encode algorithm supported by the subchannel, not in use now.
           It is invalid and reserved, and set to 0. */
    DWORD           dwAudioCodec;
        /* audio encode algorithm supported by the main channel, not in use now.
           It is invalid and reserved, and set to 0. */
    DWORD           dwSubAudioCodec;
        /* audio encode algorithm supported by the Subchannel.
           It is invalid and reserved, and set to 0. */
    DWORD          dwInputVideoPosition;
       /* whether to support the adjustment of the location of the video input,
          DS-42xx doesn't support it. */
    DWORD         dwMbcsOsdMode;
       /* the related functions supported by Multi-byte character set OSD,
          0: supports characters of the 8bit gray adjustment arrange from 0~225,
          1: only supports white and black characters,
          2: not support this kind of OSD mode (Only encode channel supports this
             OSD mode).
          DS-42xx card only supports black and white brightness adjustment. */
    DWORD       dwUnicodeOsdMode;
       /* the related functions supported by Unicode OSD,
          0: supports the character brightness adjustment, supports setting characters
             and background color, and supports setting translucent effect to the
             character and background;
          1: only supports black and white character, and doesn't support brightness
             adjustment, doesn't support setting the characters and background color,
             and only supports setting translucent effect to characters.
             DS-42xx card supports black and white brightness adjustment */
    DWORD       dwImageStream;
       /* whether to support capturing original Image steram. */
    DWORD       dwEncoderPIP;
       /* whether to support the encoding preview function of 'picture in picture'.*/
    DWORD       dwAdaptiveMotionDetection;
       /* whether to support self-adaptive motion detection, for DS-40xx & DS-43xx
          card */
    DWORD       dwEncoderAudioOutput;
```

```
    /* whether to support internal audio output of encode channel */
DWORD       dwEncoderAudioExtOutput;
    /* whether to support external audio output of encode channel */
DWORD        dwAudioPreview;
    /* the type supports live audio,
     0: doesn't support,
     1: supports live audio with connecting audio cable,
     2: supports live audio in PCI mode, without connecting audio cable,
     3: supports both the two types */
DWORD       dwSubVideoCapture;
    /* whether to support the subchannel encoding */
DWORD      dwEncoderPictureFormat;
    /* the supported main channel encoding resolution. DS-40xxHCSI card and
       the DS-40xxHSI card in non-expansion mode, only supports CIF or even
       lower resolution. DS-40xxHSI card can support the 4CIF/DCIF/2CIF/CIF/
       QCIF in expansion mode, DS-4216HCI supports the resolution up to CIF,
       and DS-42xxHFVI card supports 4CIF/DCIF/2CIF/CIF/QCIF.*/
DWORD       dwEncoderPictureFormatEx;
     /* the supported main channel encoding resolution, and is invalid.
        Corresponds to the type of CAP_PICTURE_FORMAT_EX */
DWORD       dwSubEncoderPictureFormat;
     /* the supported subchannel encoding resolution, DS-42xx card only
        supports the CIF encoding resolution or lower. Corresponds to the types
        of CAP_PICTURE_FORMAT */
DWORD       dwSubEncoderPictureFormatEx;
     /* the supported subchannel encoding resolution, DS-42xx card supports the
        CIF encoding resolution or lower. Corresponds to the type of
        CAP_PICTURE_FORMAT_EX */
DWORD       dwWatermark;
     /* support watermark or not, DS-42xx card doesn't support it */
DWORD       dwStreamCRC;
     /* support CRC Checksum or not, DS-42xx card doesn't support it */
DWORD       dwStreamEncrypt;
     /* whether to support the encoding stream encryption */
DWORD  dwEncoderParam;
     /* whether to support the adjustment of the main channel encoding
      framework and encoding complexity */
DWORD          dwSubEncoderParam;
     /* whether to support the adjustment of sub-channel encoding framework
      and encoding complexity */
DWORD       dwEncoderVideoOutput;
     /* whether to support the encode channel video output, non-matrix. */
DWORD          dwEncoderVideoExtOutput;
     /* whether to support the encode channel video matrix output.*/
```

```
    DWORD       dwWatchDog;
        /* whether to support watchdog, only DS-40xxHCSI cards supports. */
    DWORD           dwAlarmIn;
        /* whether to support the alarm input, only DS-40xxHCSI card supports. */
    DWORD           dwAlarmOut;
        /* whether to support the alarm output, only DS-40xxHCSI card supports.*/
    DWORD       dwDeInterlace;
        /* supported the deinterlace mode, see to DEINTERLACE_MODE */
    DWORD       dwDrawEncoderLineRect;
        /* whether to support the encode channel to draw line an rect on video
           picture by DSP */
     DWORD          dwDisplayFormat;
        /* supported video output formats, the specific definition refer to
          DISPLAY_FORMAT. */
     DWORD          dwReserved[20];   /* reserved */
}CHANNEL_CAPABILITY_EX;
```

Channel capacity set (CHANNEL_CAPABILITY_TYPE), corresponding to the members of CHANNEL_CAPABILITY_EX.

```
typedef enum _CHANNEL_CAPABILITY_TYPE_
{
    NormalCapability        = 1,
        /* describe the capability of functions, these functions are outside of the
           functions which are corresponding to the capability of
           CHANNEL_CAPABILITY_EX structure */
    VcaChanType             = 2,
        /* intelligent channel type, corresponding to vcaChanType */
    VcaCapability           = 3,
        /* intelligent capability, corresponding dwVcaCapability */
    StreamPackType          = 4,
        /* the encoding package format supported by main stream channel,
           corresponding to dwStreamPackType */
    SubStreamPackType       = 5,
        /* the encoding package format supported by sub-channel, corresponding to
           dwSubStreamPackType */
    VideoCodec              = 6,
        /* the video encoding algorithm supported by main stream channel,
           corresponding to dwVideoCodec */
    SubVideoCodec           = 7,
        /* the video encoding algorithm supported by sub-channel, corresponding to
           dwSubVideoCodec */
    AudioCodec              = 8,
        /* the audio encoding algorithm supported by main stream channel,
           corresponding to dwAudioCodec */
```

```
SubAudioCodec           = 9,
    /* the audio encoding algorithm supported by the sub-channel,
       corresponding to dwSubAudioCodec */
InputVideoPosition      = 10,
    /* whether to support the adjustment of the location of the video input,
       corresponding to dwInputVideoPosition */
MbcsOsdMode     = 11,
    /* multi-byte character set OSD related functions, corresponding to
       dwMbcsOsdMode */
UnicodeOsdMode          = 12,
    /* Unicode OSD related functions, corresponding to dwUnicodeOsdMode */
ImageStream     = 13,
    /* whether to support capturing original image stream, corresponding to
       dwImageStream */
EncoderPIP          = 14,
    /* whether to support picture in picture function, corresponding to
       dwEncoderPIP */
AdaptiveMotionDetection     = 15,
    /* whether to support self-adaptive motion detection, corresponding to
       dwAdaptiveMotionDetection */
EncoderAudioOutput              = 16,
    /* whether to support the encode-channel audio output, non-matrix,
       corresponding to dwEncoderAudioOutput */
EncoderAudioExtOutput           = 17,
    /* whether to support encode-channel audio matrix output, corresponding to
       dwEncoderAudioExtOutput */
AudioPreview                    = 18,
    /* supported live audio, corresponding to dwAudioPreview */
SubVideoCapture                 = 19,
    /* whether to support the sub-channel encoding, corresponding to
       dwSubVideoCapture */
EncoderPictureFormat            = 20,
    /* the supported main channel encoding resolution, corresponding to
       dwEncoderPictureFormat */
EncoderPictureFormatEx      = 21,
    /* the supported main channel encoding resolution, corresponding to
       dwEncoderPictureFormatEx */
SubEncoderPictureFormat     = 22,
    /* the supported sub-channel encoding resolution, corresponding to
       dwSubEncoderPictureFormat */
SubEncoderPictureFormatEx   = 23,
    /* the supported sub-channel encoding resolution, corresponding to
       dwSubEncoderPictureFormatEx */
Watermark                       = 24,
```

```
        /* whether to support watermark, corresponding to dwWatermark */
    StreamCRC                   = 25,
        /* whether to support CRC Checksum, corresponding to dwStreamCRC */
    StreamEncrypt               = 26,
        /* whether to support the stream encryption, corresponding to
           dwStreamEncrypt */
    EncoderParam                = 27,
        /* whether to support the adjustment of the main channel encoding
       framework and encoding complexity, corresponding to dwEncoderParam */
    SubEncoderParam             = 28,
        /* whether to support the adjustment of sub-channel encoding framework
           and encoding complexity, corresponding to dwSubEncoderParam */
    EncoderVideoOutput          = 29,
        /* whether to support the encode channel video output, non-matrix,
           corresponding to dwEncoderVideoOutput */
    EncoderVideoExtOutput       = 30,
        /* whether to support the encode channel video matrix output,
           corresponding to dwEncoderVideoExtOutput */
    WatchDog                    = 31,
        /* whether to support watchdog, corresponding to dwWatchDog */
    AlarmIn                     = 32,
        /* whether to support alarm input , corresponding to dwAlarmIn */
    AlarmOut                    = 33,
        /* whether to support alarm output, corresponding to dwAlarmOut */
    DeInterlace                 = 34,
        /* the Supported deinterlace mode, corresponding to dwDeInterlace */
    DrawEncoderLineRect         = 35,
        /* whether to support encode channel to draw line and rect on video picture
           by DSP, corresponding to dwDrawEncoderLineRect */
    DisplayFormat               = 36
        /* the supported video output format, corresponding to dwDisplayFormat */
} CHANNEL_CAPABILITY_TYPE;
```

6.6.1 Get Channel Capacity: GetChannelCapability()

API:

int GetChannelCapability(CHANNEL_TYPE chanType, UINT chan, CHANNEL_CAPABILITY_EX * pCapability);

Parameters:

CHANNEL_TYPE	chanType;	channel type
UINT	chan;	channel index
CHANNEL_CAPABILITY_EX	*pCapability;	channel capability set

Return value:

ERR_KERNEL

Remarks:

Get the capability set of specified channel.

6.6.2 Get Capacity by API Name: CheckFunctionChannelCapability()

API:

DLLEXPORT_API int CheckFunctionChannelCapability (CHANNEL_TYPE chanType, UINT chan, const char *functionName, FUNCTION_CHANNEL_CAPABILITY *pFuncChanCap);

Parameters:

CHANNEL_TYPE chanType; channel type
UNIT chan; channel index
char* functionName; the name of API
FUNCTION_CHANNNEL_CAPABILITY *pFunChanCap;
channel capability

```
typedef struct _FUNCTION_CHANNEL_CAPABILITY_
{
    CHANNEL_CAPABILITY_TYPE dwChanCapType1
                                        //channel capability type 1
    CHANNEL_CAPABILITY_TYPE dwChanCapType2
                                        //channel capability type 2
    DWORD    dwChanCapValue1    //channel capability value 1
    DWORD    dwChanCapValue2    //channel capability value 2
    DWORD    dwReserved[8]      //reserved
}FUNCTION_CHANNEL_CAPABILITY;
```

Return value:

ERR_KERNEL

Remarks:

This API will output the channel capability according to the input API name, and save the capability information in the structure FUNCTION_CHANNEL_CAPABILITY.

6.7 Smart Motion Detect（SMD） API

6.7.1 Register callback function:RegisterSMDCallback()

API：

int __stdcall RegisterSMDCallback(SMDResultCallBack pSMDFunc, void* pContext);

Parameters:

SMDResultCallBack pSMDFunc； SMD callback function pointer

void* pContext; user data

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

SMD Callback Function:

typedef void (*SMDResultCallBack)(int chan, int ruleId, void* context);

Parameters:

int chan ; alert channel

int ruleId; alert rule id

void* context; user data

6.7.2 Open/Close SMD: SetupSMD()

API:

int __stdcall SetupSMD(HANDLE hChannelHandle, int param);

Parameters:

HANDLE hChannelHandle; channel handle

int param; SMD param;

the first bit means Open/Close SMD

param = 0 : Close SMD

param = 1 : Open SMD and not save SMD info

the 16th bit means Whether PS stream including SMD info or not

param = 0x10001 means open SMD and save SMD info in encoded stream

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

Remarks:

for all Ds43xx series encode card but Ds4316HCI-E, one dsp can open one SMD channel; for example: Ds4316HW-E has two dsps, one is for 0-7 encode channel, the other is for 8-15channel;then we can get one channel open SMD from 0-7 channel, and get another channel open SMD from 8-15channel;

Only PS pack type and main stream support saving SMD info in cap file, we can replay the cap file with hikvision VSplayer.

6.7.3 Set SMD Rule Info: SetSMDRule()

API:

int __stdcall SetSMDRule(HANDLE hChannelHandle, VCA_SMD_RULE* pSmdRule, UINT ruleNum, int reserved);

Parameters:

HANDLE hChannelHandle； channel handle
VCA_SMD_RULE* pSmdRule; rule detail info
UINT ruleNum; rule num
int reserved; reserved

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

Remarks:

If more than one rule is to set, pSmdRule param should be the first pointer of the VCA_SMD_RULE array;

Note:

If the event type is corss direction, the vertexNum of alert region must be 2;

If the event type is intrustion, the vertexNum of alert region must be 4;

Cross Direction

```
typedef enum
{
    SMD_BOTH_DIRECTION,     both direction
    SMD_LEFT_TO_RIGHT,      from left to right
    SMD_RIGHT_TO_LEFT       from right to left
}VCA_SMDCROSS_DIRECTION;
```

Event Param

```
typedef union
{
    int param[6];               reversed
    int intrusionDelay;         0--2500 frames, alert until intrusion happen for {delay} frames
    VCA_SMDCROSS_DIRECTION crossDirection;
}SMD_PARA_SETS;
```

Event type

```
typedef enum _SMD_EVENT_TYPE_
{
    SMD_EVENT_TRAVERSE_PLANE    = 0x01,         TRAVERSE_PLANE
    SMD_EVENT_INTRUSION         = 0x08,         INTRUSION
}SMD_EVENT_TYPE;
```

SMD rule detail info

```
typedef struct
{
    UINT                ruleID;         rule ID,  [1,8]
    SMD_EVENT_TYPE      eventType;      event type
    SMD_PARA_SETS       paraSets;       event param
    POLYGON_F           polygon;        alert region
```

```
    UINT                    reserved[8];
}VCA_SMD_RULE;

typedef struct
{
    float x;            x coordinate:[0,1]
    float y;            y coordinate:[0,1]
}POINT_F;

typedef struct
{
    unsigned int vertexNum;             vertex num
    POINT_F point[10];                  coordinate
}POLYGON_F;
```

6.8 Video Quality Diagnostics（VQD） API

6.8.1 Start VQD Polling:StartVQD()

API:

int __stdcall StartVQD(UINT vqdInterval);

Parameters:

UINT vqdInterval; VQD polling interval,in seconds.
minimum is 10s,maximum is 7*24 hour(604800s).

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

Remarks:

For all DS43xx series encode card but DS4316HCI-E.

Start VQD Polling,set polling interval.

All the DS43xx encode channels(except DS4316HCI-E) will do VQD polling during every polling interval time.The result of diagnostics will be put on stream data with PktVQDScore frame type.

PktVQDScore frame data structure :

```
typedef struct _HIK_VQD_SCORE
{
    int   blur_score;       //Blur detection results: 0-100
    int   luma_score;       //Luma detection results: 0-100
    int   chroma_score;     //Chroma detection results: 0-100
    int   res[4];           //reserved
}HIK_VQD_SCORE;
```

The score represent the corresponding evaluation of the video.

For example,blur detection score represent the blur degree of video.

blur more serious with higher scores.Score range is 0 to 100.

Score threshold should be set according to the actual application,

The recommended default value is:

Blur detection:60 Luma detection:75 Chroma detection:64

6.8.2 Stop VQD Polling:StopVQD()

API:

int __stdcall StopVQD(l);

Parameters:

--

Return value:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on error code.

Remarks:

Stop VQD Polling.

7 Appendix Old/Expired API

In this chapter there are definition and description of APIs in the previous SDK version which are invalid or expired , and can be replaced by other APIs in the SDK.

7.1.1 Get Total Channel Number: GetTotalChannels()

API:

int GetTotalChannels()

Parameters:

--

Return:

The function gets the available number of encode channels.

Remarks:

Each DS-4004HCI has 4 encode channels.

Each DS-4008HCI has 8 encode channels.

Each DS-4016HCI has 16 encode channels.

Each DS-4002MDI has 4 decode channels and 2 display channels.

Each DS-4004MDI has 8 decode channels and 4 display channels.

To get the available number of decode channels, please call API GetBoardDetail() instead.

7.1.2 Get Total Channel Number: GetTotalDSPs()

API:

unsigned int GetTotalDSPs ()

Parameters:

--

Return:

The function also gets the available number of encode channel.

Remarks:

The function doesn't return the available number of DSP.

Call function GetDspCount() to get available number of DSP.

Please call API GetDSPCount() instead.

7.1.3 Set OSD Time (For Checking Time): SetupDateTime()

API:

int SetupDateTime(HANDLE hChannelHandle, SYSTEMTIME *now)

Parameters:

HANDLE	hChannelHandle;	Channel handle
SYSTEMTIME	*now;	pointer to the system time or net time(If it is set as NULL,that means the SDK will check time with the local system)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set the time of OSD (can be used for check the time with net). After calling this function: SetOsd( ) won't take effect .

This function is expired after version 4.0 SDK.

7.1.4 Get Last Error: GetLastErrorNum()

API:

int GetLastErrorNum(HANDLE hChannelHandle, ULONG *DspError, ULONG *SdkError);

Parameters:

HANDLE	hChannelHandle;	Channel handle
ULONG	*DspError;	DSP Error
ULONG	*SdkError;	SDK Error(defined in section 4)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get the error report of SDK and DSP. This function only works for DS-40xxHI series card.

7.1.5 Set Video Standard: SetVideoStandard()

API:

int SetVideoStandard(HANDLE hChannelHandle,VideoStandard_t VideoStandard)

Parameters:

HANDLE	hChannelHandle;	channel handle
VideoStandard_t	VideoStandard;	video standard(defined in section 3.3)

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set Video Standard for DS-40xxHI series card;

For DS-40xxHCI/DS-40xxHSI/DS-40xxHCSI card, this function is not necessary to be called, as the SDK can recognize the video standard of input video

automatically. Relative APIs: SetDefaultVideoStandard(), SetDefaultHDVideoStandard()

7.1.6 Reset DSP: ResetDSP()

API:

int ResetDSP(int DspNumber)

Parameters:

int DspNumber; DSP Index

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Reset one of the DSP on the cards which is indexed by DspNumber, This function can be called when some DSP is deadlock or malfunction. It is suggested that channel that related to this DSP should be closed before call this function and open again after it.

This function is expired.

7.1.7 Set Watch Dog: SetWatchDog()

API:

int SetWatchDog(UINT boardNumber,BOOL bEnable)

Parameters:

UINT boardNumber; card number Index

BOOL bEnable; enabled or not

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Set watch dog. For the DS-4016HCSI card,it provides 4 pinout on the front.The customer need connect the PC RESET SW leads to two of the 4 pinout and connect the other 2 pinout to the Reset connector on the motherboard.

This function is expired.

7.1.8 Get Special Capability of Card: GetCapability()

API:

int GetCapability(HANDLE hChannelHandle,
CHANNEL_CAPABILITY *Capability);

Parameters:

HANDLE	hChannelHandle;	channel handle
CHANNEL_CAPABILITY	*Capability;	defined in Section

```
typedef struct tagChannelCapability{
    UCHAR bAudioPreview;        //Live audio
    UCHAR bAlarmIO;             //Alarm IO, invalid
    UCHAR bWatchDog;            //Watch Dog, invalid
}CHANNEL_CAPABILITY, *PCHANNEL_CAPABILITY;
```

Return:

If the function succeeds, the return value is 0.

If the function fails, the return value is the error code defined on section 3.

Remarks:

Get special capability of card