DS-4xxx Encoding/Decoding Card

SDK Documentation

Version 5.1

(For Linux)

Hangzhou Hikvision Digital Tech. Co., Ltd.

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:
overseabusiness@hikvision.com

Find out more about HIKVISION at www.hikvision.com

CONTENTS

1. Hikvision Card Overview

Hikvision H.264 encoding and decoding card is a special production, which is designed for digital surveillance market. It uses high-performance Video compression of H.264 standard with OggVorbis Audio coding algorithm to accurately achieve video and audio Real-time coding (CIF 25 f/s PAL or 30 f/s NTSC) based on hardware completely. It also has the function such as dynamic bit rate, controllable frame rate, frame mode, dynamic image quality control, and real-time audio preview and alarming on Video signal loss, and can adjust any channel's parameters independently with stable and reliable performance. Compared with MPEG-I products, it can greatly save storage space and more suitable for broadband or narrowband network transmission with the same image quality, so it is one of the best choices for digital surveillance products.

Hikvision DS-400xMDI matrix decode board is the key product to build digital video matrix and network matrix. It can be used for decode H.264 streams compressed by Hikvision encoding products. It is a bridge between digital and analog system, and strongly support concentrative digital surveillance solution.

The SDK of Hikvision DS-4000 series card is made up of encoding system SDK, network SDK and player SDK. This manual especially describes encode system SDK and decode system SDK, as to the other SDK you can refer to the relevant documents. This SDK is the local software interface program, which is designed for one or multiple channel boards of this series, to provide for internet application developers in the form of dynamic data base. It also has Demo (dsdemo and mddemo) and corresponding source code, which can effectively decrease the period of development applications.

When using, software developer should especially notice that they can modify all the parameters like resolution, stream code, frame structure except the stream code (complex stream, video stream). Namely, it can transform frame rate (SetIBPMode(…)) and quantization coefficient(SetDefaultQuant) in the course of compression, while no need of stopping or starting compression but still within a file record. The player can automatically identify parameters such as frame rate and can play normally according to current compressed frame rate.

Compressed bit rate can be controlled by dynamically modifying the quantization coefficient (I, B, P). If the bit rate is too high, increase the coefficient; whereas, decrease it. Certainly, the coefficient doesn't need to be decreased if enough.

Moving detection of DS-4000 series compression card is independent from compression. It can be done while no encoding. It is valuable that the frame rate can be transformed. When moving, record as high frame rate (30F/S); whereas, record as low frame rate. Recording in the same file can greatly save hard disk space.

SetLogo(…) not only can be used as LOGO but also to envelop some image area.

DS4008HSI, DS4016HSI support 8 channels and 16 channels CIF/QCIF video and audio real-time compression respectively, but not support 4CIF、2CIF、DCIF, and sub channel is also not supported.

SDK interface is completely the same as those of DS-400xM/DS-400xH serial boards and more functions are added. Developed internet applications can be migrated very fast.

DS-42xx Series card has 3 models: DS-4216HCI、DS-4208HFVI、DS-4216HFVI, which have good performance and low power consumption.

DS-4216HCI supports 16 channels CIF/QCIF real-time encoding, or 4CIF/2CIF non real-time recording, and 16 sub channels CIF/QCIF encoding (if main channels are 4CIF non real-time recording, sub channels are invalid).

DS-4208HFVI supports 8 channels 4CIF/2CIF/CIF/QCIF real-time encoding and 8 sub channels CIF/QCIF encoding.

DS-4216HFVI supports 16 channels 4CIF/2CIF/CIF/QCIF real-time encoding and 16 sub channels CIF/QCIF encoding.

Note: Reboot your computer after the driver installed.

Demo program included in our SDK must run on XWINDOW.

2. SDK Version Update

Version 5.1 Released, July 2009

1. Compatible with DS-42xx card and DS-40xx card.
2. DS-42xx Card supports Audio Preview via PCI slot, which means audio preview cable isn't needed any more.
3. DS-42xxHFVI supports TV-Out function by invoking SetEncoderVideoExtOutput() and SetEncoderAudioOutput()
4. DS-4216HCI supports 4CIF/2CIF not real-time recording. While using 4CIF not real-time recording, all sub channels are disabled. .

Notes:

Some API has different effects on DS-42xx and DS-40xx card; please pay attention to the API description.

VER 4.3 (Build 20080422)

1. Support encoding 256 channels in one PC.
2. DS-4000MDI supports capturing decoded data by invoking: HW_SetDecoderVideoCapture().
3. Support configuring the frame rate of local video matrix output by invoking SetEncoderVideoExtOutput()
4. Lower decoding delay of DS-4000MDI card.
5. 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 grabbing YUV image, JPEG image, raw stream, and matrix output, but it doesn't support sub stream.
6. Support encoding Audio stream only.
7. Improve the sense of video signal detection
8. Support processing the color crosstalk on HCI,HCI+ card
9. Improve the performance of encoding and decoding.

Bug fixed:

1. Fixed the bug that if the application starts again before the SDK exits completely after turn off customer's application, the PC may be freezing.
2. The DS-400xMDI with previous SDK will output the first two Audio channels after launching. Version 4.3 SDK won't output audio by default.
3. Fixed GetSoundLevel() doesn't work on first 12 channels of DS-4016HCSI card in v4.2 SDK
4. Fixed the image of DS-400xMDI card may be overlapped, if use multithreading to input data for decoding.
5. Fixed DS-400xMDI card may can't decode the data at the end of file.
6. Fixed bugs that when grab BMP, the image maybe dislocation, and when grab Jpg image, the operation may be timeout and can't be resumed.
7. Fixed image decoded by DS-400xMDI will be corrupted, including NTSC QCIF image.

New Function:

1. RegisterDecoderVideoCaptureCallback()
2. HW_SetDecoderVideoCapture()

VER 4.2

1. Add new API StartVideoPreviewEx() to extend the function StartVideoPreview() and provide more display mode for board DS-40xxHC.
2. Support playing multi-encode files for board DS-40xxMDI
3. Add new API HW_SetDecoderPostProcess() to avoid the flash while decoding.
4. Add new API HW_InputDataByFrame() which is the similar with HW_InputData () and support displaying I frame in real-time
5. Update the decoder of MDI board, and improve the quality of image
6. Improve the thread-safety of SDK
7. support 4008HSI and 4016HSI PCI-Card

VER 4.1

1. Support the new board DS40xxHC+
2. Improve the image quality for the image, special for 4CIF format
3. Add the file index for DS400xMD, Then could get the begin time and end time for the record file. And could locate playing by time or frame number.

VER 4.0

1. Support new board DS4016HCSI, DS4016HCI,DS4002MDI,DS4004MDI
2. Use new mode to realize video preview. It peel off the SDL from the SDK, And the SDK provide the video data directly. Then the application could carry out the preview by it by the SDL or other more efficient ways. Please refer to the demo “dsdemo”. It is use SDL to realize the preview. Please note, in demo the parameter of SDL_VIDEO_YUV_HWACCEL was set as 1,it could cause low cpu load and more wonderful picture. But it is restricted by the system. In some system as some video card, there will case the image flash. Then this parameter must change to 0.
3. Add the new API SetupMotionDetectionEx() to predigest call and advance the efficiency. Now realize motion detect only by calling SetupMotionDetectionEx, StartMotionDetection. After call this API, the SDK will not provide the frames type as PktMotionDetection. You could get the motion status by callback function MotionDetectionCallback() and parameter bMotionDetected.
4. Add the new API SetOsdDisplayModeEx(). That could support up to 8 lines OSD. And fixed the precision of the OSD. Now the user could not call SetupDateTime() to adjust the time.
5. Add the new API GetJpegImage() to get the JPEG image directly.
6. There have watchdog In DS4016HCSI. You could call SetWatchDog() to enable it.

The DS4002MD and DS4004MD: **H.264 High Resolution Video & Audio Matrix Decode Board**

Decode function:

1. Each DS4002MDI can realize 4 channels QCIF,CIF,2CIF or 3 channels DCIF, 2 channel

4CIF real time decoding (DS-4004D does not support 4CIF resolution decoding).

2. Audio output: 2 channels, can choose any 2 of the 4 decoding channels.
3. Video output: 2 channels, each video output can be max divided into 16 windows.
4. Audio preview: each DS4002MDI support 1 channel audio preview output.
5. Software: from version3.0 SDK, Hikvision provide the support to DS-4002MDI.
6. Support the mixed insert of HI series board, HCI series board and MDI board in one PC.
7. Within one SDK, it can simultaneously support HI, HCI and MDI board.
8. Most of the API in decoding part can be compatible with the SDK of previous Hikvision's decode card DS-4004D.
9. At present one PC can max support 16 pieces of DS4002MDI board, that is to say 64-channel decoding and 32-channel video output.
10. Basic decoding capability (the possible DSP resources occupied when decoding one channel):

    CIF: 12% (512Kb); 16% (2Mb)

    2CIF: 30% (1Mb)

    DCIF: 28% (768Kb)

    4CIF: 50% (1Mb); 60% (3Mb)

    Explanation: the test file above is the stable image under fixed bit rate. At present the further optimization is on process, the performance will be improved in the afterward versions.

Video Matrix:

1. Video input: the real time video captured by HCI board, the video after decode of MDI board (local files or real time stream transfer from the network).
2. Video output: output channel of MDI channel. Video output support screen partition, each video output can be divided into max 16 windows, and picture switch of the video matrix use the windows as the unit.
3. Matrix control: as to all the HCI boards and MDI boards in one PC, each encoding channel of HCI board and each decoding channel of MDI board can display its video in any window of any channel of any MDI boards.

Basic parameters of Matrix:

- Each DS4002MDI support 2 channel matrix output, the resolution for each output is 4CIF.
- Each encoding channel of HCI board can simultaneously support 1-channel display card preview and 1-chanel matrix output.
- Each decoding channel of MDI board can simultaneously support 1-channel display card output and 2-channel matrix output.
- Each video output support picture-in-picture function, the position of the each window can be adjusted dynamically.
- The summation of total surface of the each video output cannot exceed 4CIF + QCIF, that is to say it support one 4CIF full screen output plus one QCIF picture-in-picture output.

3. Data Structure Definition

3.1 Definition for frame type

PktError	illegal frame data
PktSysHeader	System header
PktIFrames	I frame
PktPFrames	P frame
PktBBPFrames	BBP frame
PktAudioFrames	Audio frame
PktMotionDetection	Motion detection frame
PktSFrames	Frame types transferred during capturing I frame
PktSubIFrames:	when in double decoding, I frame in subchannel
PktSubPFrames:	when in double encoding, P frame in subchannel
PktSubBBPFrames:	when in double encoding, BBP frame in subchannel
PktSubSysHeader:	when in double encoding, system header in subchannel

3.2 Definition for video standard

StandardNone	No video signal
StandardNTSC	NTSC format
StandardPAL	PAL format

3.3 Definition for extraordinary ability

```
typedef struct tagChannelCapability{
    UCHAR bAudioPreview;        Audio preview
    UCHAR bAlarmIO;             Alarming signal
    UCHAR bWatchDog;            Watch dog
}CHANNEL_CAPABILITY, *PCHANNEL_CAPABILITY;
```

3.4 Frame data Stat.

```
typedef struct tagFramsStatistics{
    ULONG VideoFrames;          Video frame
    ULONG AudioFrames;          Audio frame
    ULONG FramesLost;           Lost frame
    ULONG QueueOverflow;        Buffer overflow
}FRAMES_STATISTICS, *PFRAMES_STATISTICS;
```

3.5 The preview rectangle

```
typedef struct tagRect{
    short RectTop;
```

```
    short RectBottom;
    short RectLeft;
    short RectRight;
}RECT;
```

3.6 Edition information

```
typedef struct tagVersion{
    ULONG DspVersion, DspBuildNum;          DSP edition and BUILD mark
    ULONG DriverVersion, DriverBuildNum;    Drive edition and BUILD mark
    ULONG SDKVersion, SDKBuildNum;          SDK edition and BUILD mark
}VERSION_INFO, *PVERSION_INFO;
```

3.7 Definition of motion detection data

DS40xxHCI serial boards offer motion intensity information to deal with motion detection. When setting the motion detection areas, use 32x32 as one unit, resolution using 4CIF (704x5760) there re 22 units in one row (704/32). There are 18 lines (576/32) when in PAL format, 15 lines (480/32) in NTSC format, no matter what kind of encoding format. Through the test, using this way the sensibility and the reliability have been developed compared with the H serial board, and simplify the return data. The value of return is 18 DWORD, the corresponding height of the screen is 576/32=18 lines (in PAL). The corresponding width of 0-21 unit of the DWORD is 704/32=22 rows. Among them, 1 means motion and o still, and can also call the original analyses result of MotionAnalyzer().

4. Encoding Card SDK

4.1. Order of Function Call

A.

Set the default video standard	**SetDefaultVideoStandard()**

B.

Init system	**InitDSPs()**

C.

Get total encode channels	**GetTotalChannels()**
Open encode channel	**ChannelOpen()**
Register the call back to get the raw stream data	**RegisterImageStreamCallback()**
Start video preview	**StartVideoPreview ()**

D.

Set OSD	
Set the mode for OSD(just support 2lines)	**SetOsdDisplayMode()**
Set the mode for OSD(support 8 lines)	**SetOsdDisplayModeEx ()**
Start OSD	**SetOsd()**
set Logo	
Convert 24 bits bmp file to yuv data	**LoadYUVFromBmpFile()**
Set logo display mode	**SetLogoDisplayMode()**
Set the logo for data and location	**SetLogo()**
Set the mask	
Enable the mask	**SetupMask()**

E.

Set the encode format for master channel：	**SetEncoderPictureFormat()**
Set the stream type for master channel：	**SetStreamType()**
Set the image quality：	**SetDefaultQuant()**
Set the structon for encode and frame rate:	**SetIBPMode()**
Set the max bit rate：	**SetupBitrateControl()**
Set the mode for stream：	**SetBitrateControlMode()**
Set the parameter for the video as luminance, saturation, contrast,：	**SetVideoPara()**

F. Motion detection mode 1

Set the precision for the detect：	**AdjustMotionDetectPrecision()**
Set the area and amount：	**SetupMotionDetection()**
Start motion detect：	**StartMotionDetection()**
Analyzer：	**MotionAnalyzer()**

G. Motion detection mode 2

Set motion detection：	**SetupMotionDetectionEx()**

Start:	**StartMotionDetection()**

H.

Get original image:	**GetOriginalImage()**
Save to file as bmp:	**SaveYUVToBmpFile()**
Get the PEG:	**GetJpegImage()**

I.

Get the audio level for the scene:	**GetSoundLevel()**
Set the audio preview:	**SetAudioPreview()**

J.

Get the video signal:	**GetVideoSignal()**
Get the vision for the sdk:	**GetSDKVersion()**
Get the video parameter:	**GetVideoPara()**
Get the version and the SN for the board:	**GetBoardInfo()**
Get the statistics for the frames:	**GetFramesStatistics()**
Get the detail info for the board:	**GetBoardDetail()**
Get the detail info for the dsp:	**GetDspDetail()**

K.

Start video encode:	**StartVideoCapture()**

L.

Start get raw image data:	**SetImageStream()**

M.

Set the encode type for the sub channel:	**SetSubStreamType ()**
Set the picture format for the sub channel:	**SetSubEncoderPictureFormat()**
Switch between the sub channel and the master:	**SetupSubChannel()**
(The sub channel could have diffident frame rate and encode coefficient)	
Start video encode from the sub channel:	**StartSubVideoCapture()**

N. Exit

Stop getting raw stream data:	**SetImageStream()**
Stop motion detect:	**StopMotionDetection()**
Stop video encode from the master channel:	**StopVideoCapture()**
Stop video encode from the sub channel:	**StopSubVideoCapture()**
Stop video preview:	**StopVideoPreview()**
Stop the encode channel:	**ChannelClose()**
Release resource:	**DeInitDSPs()**

The SetStreamType() and SetSubStreamType () could not be called while encoding. Others setting as video parameters，OSD、Logo、 image format、frame rate、bit rate、coefficient all could be called while encoding.

4.2. API of Encoding Card

4.2.1. int GetLastErrorNum()

Explanation:

Get the last error number when there is exception.

Return value:

The error number is described in ds40xxsdk.h

4.2.2. int InitDSPs();

Explanation:

This API will initialize all cards. It should be invoked when application starts.

Return value:

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

When it returns 0, if there is compression card in computer, 0 means initialization is failed, if there is only decoding card, 0 is normal, HW_InitDecDevice() is suggested for initializing decoding card. DeInitDSPs() is the corresponding API.

4.2.3. int DeInitDSPs();

Explanation:

To close the functions in every board and must be called before exiting application;

Return value:

0—success; -1 – failed;

4.2.4. int ChannelOpen(int ChannelNum, STREAM_READ_CALLBACK streamReadCallback);

Parameter:

int ChannelNum: // channel No. (0-n)

STREAM_READ_CALLBACK StreamReadCallBack: // Data callback function

void StreamReadCallBack(int ChannelNum, void * DataBuf, int FrameType ,
int Length, int FrameNum);

int ChannelNum //channel num (0-n)

void * DataBuf //pointer of frame data

int FrameType //frame type defined in section 3.1

int Length //frame length

The callback function will return encoded frame and Motion detection Frame.

Explanation:

Open channels and get operation handle. All operations related with this channel need use this handle;

Return value:

>=0 —success and it is channel handle; -1 – failed;

4.2.5. int ChannelClose(int channelHandle) ;

Parameter:

int channelHandle channel handle;

Explanation:

Close channel and release correlative resource;

Return value:

0—success; -1 – failed;

4.2.6. int GetTotalChannels();

Explanation:

Get total valid channel number in system.

Return value:

If return value is less than the number of channels installed in system, it is means those initializations of DSP are failed.

4.2.7. int GetTotalDSPs();

Explanation:

Get the number of DSP in system.

Return value:

If return value is less than the number of DSP installed system, it is means those some initializations of DSP are failed.

4.2.8. int GetBoardCount()

Explanation:

Get the number of card in system.

Return value:

The number of the cards

4.2.9. int GetBoardDetail (UINT boardNum,DS_BOARD_DETAIL *pBoardDetail)

Parameter:

UINT boardNum the index of the board

DS_BOARD_DETAIL *pBoardDetail the info of the board

```
typedef struct
{
    BOARD_TYPE_DS type;          //the type of the board
    BYTE sn[16];                 //the serial number
    UINT dspCount;               //the DSP count in the board
    UINT firstDspIndex;          //the index of the first DSP
    UINT encodeChannelCount;     //the encode channel count in the board
    UINT firstEncodeChannelIndex;  //index of first encode channel
    UINT decodeChannelCount;       //the decode channel count in the board
    UINT firstDecodeChannelIndex;  //the index for the first decoding channel
    UINT displayChannelCount;    //the display channel count
```

```
        UINT firstDisplayChannelIndex;      // the index of the first display channel
        UINT reserved1;
        UINT reserved2;
        UINT reserved3;
        UINT reserved4;
    }DS_BOARD_DETAIL;
```

Explanation:

Get the detail information for the board

Return value:

0 – success; -1 - fail;

4.2.10. int GetDspDetail (UINT dspNum,DSP_DETAIL *pDspDetail)

Parameter:

UINT dspNum DSP index for the dsp

DS_BOARD_DETAIL * pDspDetail the detail info for the dsp

```
typedef struct
{
    UINT encodeChannelCount; //the encoding channel count in the dsp
    UINT firstEncodeChannelIndex; //the index of the first encoding channel
    UINT decodeChannelCount;        //the decode channel count in the dsp
    UINT firstDecodeChannelIndex; // the index of the first decoding channel
    UINT displayChannelCount;       // the display channel count in the dsp
    UINT firstDisplayChannelIndex; // the index of the first display channel
    UINT reserved1;
    UINT reserved2;
    UINT reserved3;
    UINT reserved4;
}DSP_DETAIL;
```

Explanation:

Get the detail information for the DSP

Return value:

0 – success; -1 - fail;

4.2.11. int GetEncodeChannelCount()

Explanation:

Get the encode channel count in the system.

Return value:

The total number of encoding channels;

4.2.12. int GetDecodeChannelCount()

Explanation:

Get the decode channel count in the system.

Return value:

The total number of decoding channels;

4.2.13. int GetDisplayChannelCount()

Explanation:

Get the all display channel count in the system.

Return value:

The total number of display channels;

4.2.14. int GetBoardInfo(int hChannelHandle, UINT *BoardType, char *SerialNo);

Parameter:

int channelHandle	channel handle
Int *BoardType	the type of the board
char *SerialNo	ID number of card: Content is ASCII number of SN, SerialNo[0] corresponding to the highest, SerialNo[11] corresponding to the lowest. E.g.: "4 0 0 0 0 1 0 0 2 3 4 5" corresponding to array 4,0,0,0,1,0,0,2,3,4,5.

Return value:

0 – success; -1 - fail;

Note:

Board as follow:

DS400XM	0,
DS400XH	1,
DS4004HC	2,
DS4008HC	3,
DS4016HC	4,
DS4001HF	5,
DS4004HF	6,
DS4002MD	7,
DS4004MD	8,
DS4016HCS	9,
DS4002HT	10,
DS4004HT	11

4.2.15. int GetCapability(int hChannelHandle, CHANNEL_CAPABILITY *Capability);

Parameter:

int channelHandle	channel handle
CHANNEL_CAPABILITY *Capability	refer to section 3.3

Explanation:

Get the information of special function of the board;

Return value:

0—success; -1 – failed;

4.2.16. int StartVideoPreview(int hChannelHandle, PREVIEWCONFIG* pPreviewConf,, UINT useSyncSem);

Parameter:

int channelHandle	channel handle;
PREVIEWCONFIG* pPreviewConf	the struct refer to struct RECT
UINT useSyncSem	1 means use the semaphore to synchronize the preview.The application initialize a semaphore，then provide it to sdk by pPreviewConf-〉SyncSem. And the sdk will post this semaphore after a new frame is ready. 0 means use the timer to synchronize the preview

Explanation:

Start video preview. Please create the sdl surface by returned pPreviewConf->w and pPreviewConf->h. Then you could copy video data from pPreviewConf->dataAddr.

Return value:

0—success; -1 – failed;

4.2.17. int StopVideoPreview(int channelHandle);

Parameter:

int channelInfo channel handle;

Explanation:

Stop video preview;

Return value:

0—success; -1 – failed;

4.2.18. int SetVideoPara(int channelHandle, int Brightness, int Contrast, int Saturation, int Hue);

Parameter:

int channelHandle	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);

Explanation:

set video parameters;

Return value:

0—success; -1 – failed;

4.2.19. int GetVideoPara(int channelHandle, VideoStandard_t *VideoStandard, int *Brightness, int *Contrast, int *Saturation, int *Hue);

Parameter:

int channelHandle		window handle;
VideoStandard_t	*VideoStandard	video format (refer to section 2.3);
int *Brightness		pointer of Brightness value (0--255);

	int *Contrast	pointer of Contrast valure (0--127);
	int *Saturation	pointer of Saturation value (0--127);
	int *Hue	pointer of Hue value (0--255);

Explanation:

To get video parameter

Return value:

0—success; -1 – failed;

4.2.20. void GetSDKVersion (PVERSION_INFO VersionInfo);

Parameter:

PVERSION_INFO	VersionInfo	pointer of VERSION_INFO;

Explanation:

get the SDK version. It is consist of 16 bits BCD code , the high 8 bits means major version ,the back 8 bits means senior version , and the following 32 bits means BUILD number which indicating the time that the SDK is modified latest;

4.2.21. int SetStreamType (int channelHandle, int type);

Parameter:

int channelHandle	channel handle
int type	stream type, see the macro definition as follows:

Macro definition:

```
#define  STREAM_TYPE_VIDEO    1        //video stream
#define  STREAM_TYPE_AUDIO    2        //audio stream
#define  STREAM_TYPE_AVSYNC   3        //video&audio synchronous stream
```

Explanation:

DS-42xx card do not support STREAM_TYPE_AUDIO mode

Set stream type;

Return value:

0—success; -1 – failed;

4.2.22. int GetStreamType(int channelHandle, int *StreamType);

Parameter:

int channelHandle	channel handle
int *StreamType	point to the stream type

Explanation:

To get stream type;

Return value:

0 – success; -1 - fail;

4.2.23. int SetSubStreamType (HANDLE hChannelHandle, int type)

Parameter:

int channelHandle	channel handle
int type	stream type as the master channel

Explanation:

DS-42xx card do not support STREAM_TYPE_AUDIO mode

To set the stream type from the sub channel;

Return value:

0—success; -1 – failed;

4.2.24. int GetSubStreamType(int hChannelHandle, int*StreamType)

Parameter:

int hChannelHandle channel handle

int*StreamType the point to the stream type

Explanation:

To get the stream type from the sub channel;

Return value:

0—success; -1 – failed;

4.2.25. int StartVideoCapture(int channelHandle);

Parameter:

int channelHandle channel handle

Explanation:

To startup video capture. The users' program can process the data stream directly by using callback parameter of StreamDirect ReadCallback. Or you ca do it just like H serial boards that is: user's program to read the data stream using ReadStreamData after knowing the registered message sending to the user's program RegisterMeddageNotifyHandle by SDK.

Return value:

0—success; -1 – failed;

4.2.26. int StopVideoCapture(int channelHandle);

Parameter:

int channelHandle channel handle

Explanation:

Stop data compress;

Return value:

0—success; -1 – failed;

4.2.27. int StartSubVideoCapture(int channelHandle)

Parameter:

int channelHandle channel handle

Explanation:

To start the sub-channel video capture

Return value:

0—success; -1 – failed;

4.2.28. int StopSubVideoCapture(int channelHandle)

Parameter:

int channelHandle channel handle

Explanation:

To stop the sub-channel video capture

Return value:

0—success; -1 – failed;

4.2.29. int SetIBPMode(int channelHandle, int KeyFrameIntervals, int BFrames, int PFrames, int FrameRate);

Parameter:

int channelHandle	channel handle
int KeyFrameIntervals	key frame interval (default is 100)
int Bframes	number of B frame (default is 2)
	DS-42xx card does not support B frame
int Pframes	number of P frame
int FrameRate	frame ratio (default is 25)

Explanation:

To set frame structure, key frame interval, number of B frame and frame rate. The value of key frame interval can be not less than 12 , number of B frame can be 0, 1,2 , number of P frame is set invalid at present, the range of frame rate is from 1 to 25 , and these value can be set during capturing ;

Note:

Explanation of key frame interval: Key Frame is the image frame, which is compressed within fames in the encoding stream. Its characters are the good image definition while needing big data capacity, and usually used as the original reference of frames interval encoding. Key frame interval is the numbers of the frames between the continuous frames encoding.

Return value:

0—success; -1 – failed;

4.2.30. int SetDefaultQuant(int channelHandle, int IQuantVal, int PQuantVal, int BQuantVal);

Parameter:

int channelHandle	channel handle
int IquantVal	I frame quantization parameters
int PquantVal	P frame quantization parameters
int BQuantVal	B frame quantization parameters

Explanation:

To set video encode quantization parameters. It is used in adjusting image 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; The normal rules is the I frame and P frame is set as the same, while B frame is 3 to 5 bigger than them.

Note:

Explanation of quantitative coefficient: quantitative coefficient is the parameter, which greatly affects the encoding, image quality and bit rate under MPEG standard. The lower the quantitative coefficient is, the better the quality and the higher the bit rate. On the contrary, the worse the quality is and the lower the bit rate.

Return value:

0—success; -1 – failed;

4.2.31. int SetEncoderPictureFormat(int channelHandle, PictureForamt_t pictureFormat) ;

Parameter:

Int channelHandle	channel handle
PictureForamt_t pictureFormat	the format of coding image (4CIF, 2CIF, CIF, QCIF CIFQ)

Explanation:

Set the encoding format of the master channel.

Return value:

0— success; -1 – failed;

Remarks

DS-42xx Card doesn't support DCIF resolution.

4.2.32. int SetSubEncoderPictureFormat(int channelHandel, PictureFormat_t pictureFormat)

Parameter:

int channelHandle	channel handle
PictureForamt_t pictureFormat	the format of coding image (4CIF, 2CIF, CIF, QCIF DCIF)

Explanation:

Set the encoding format of the sub channel.

Return value:

0— success; -1 – failed;

Remarks

DS-42xx Card doesn't support DCIF resolution.

4.2.33. int SetupBitrateControl(int channelHandle, int MaxBps) ;

Parameter:

int channelHandle	channel handle;
int MaxBps	the most baud rate (more than 100000)

Explanation:

It can be used to set the maximum baud rate. If the parameter of MaxBps is set to 0 then bit-rate control is closed. When the parameter of MaxBps is set as a certain baud rate, as the encoding data stream exceeds this value, DSP will automatically adjust encoding parameter not to exceed the max baud rate. While if the data stream is smaller than the maximum baud rate then DSP don't bother it; the adjustable error is <10%;

Return value:

0—success; -1 – failed;

4.2.34. int SetBitrateControlMode(int channelHandle, BitrateControlType_t brc)

Parameter:

int channelHandle	channel handle

BitrateControlType_t	brc	bitrate control mode, brVBR and br CBR

Explanation:

This function should be cooperated with SetupBitrateControl function, When brCBR is Selected and SetBitrateControl is called with specified bitrate, the encode 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 bit rate will be the value set by SetBitrateControl and the picture quality is adjust automatically to maintain constant bitrates.

For DS-42xxHCI/HFVI Card, if the stream is video only, the minimum bit rate should be bigger than 32*1024bps, and if the stream is Video&Audio the minimum bitrate should be bigger than 96*1024bps

Return value:

0—success; -1 – failed;

4.2.35. int SetVideoStandard(int channelHandle, VideoStandard_t videoStandard)

Parameter:

int	channelHandle	channel handle;
VideoStandard_t	videoStandard	video standard

Explanation:

To set current video standard to the specified type, it's is unnecessary to call the function if we boot the system under the condition that the video camera is connected. But it is necessary to call this function if we boot the system without connecting the cameras or we change the different format cameras in the process.

Return value:

0—success; -1 – failed;

4.2.36. int SetDefaultVideoStandard(VideoStandard_t VideoStandard)

Parameter:

VideoStandard_t VideoStandard	video format

Explanation:

To set the system default video format, the defaulted is PAL.

If there is no any video input in all video input channel in system, thus this channel will process as to the defaulted format

The format for all video output channel will adopt the defaulted format when the system startup.

Note:

this function can only be run before system initialization (calling InitDSPs), or it is invalid.

4.2.37. int SetVideoDetectPrecision(int hChannelHandle,unsigned int value)

Parameter:

int	hChannelHandle	channel handle;
int	value	precision

Explanation:

To set the precision of the signal detection. The range is 0-100, default is 20. If the video signal is too weak to change high frequency, there will cause display" No Signal". To get rid of this, you could change the precision. The value is big, the precision is low.

Return value:

0 – success; -1 - fail;

4.2.38. int GetVideoSignal(int channelHandle);

Parameter :

int channelHandle channel handle

Explanation :

To set the information of connect video signal. It can be used in alarming for video lost;

Return value:

1 - No video signal

0 - valid video signal

-1 – Invalid parameter

4.2.39. int SetInputVideoPosition(int hChannelHandle, unsigned int x, unsigned int y)

Parameter:

int channelhandle	channel handle
unsigned int x	the X axis of the coordinate, the defaulted value is 8
unsigned int y	the Y axis of the coordinate, the defaulted value is 2

Explanation:

to set the position of video input, some camera preview may have some black line in the left. (x,y) is original picture coordinate of top left camera input in system processing pictures. 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 is not matching 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.

DS-42xx card does not support this function.

Return value:

0—success; -1 – failed;

4.2.40. int SetOsdDisplayMode (int channelHandle, int brightness, int translucent, int twinkInterval,unsigned short *format1, unsigned short *format2););

Parameter:

int channelHandle	channel handle
int Brightness	display brightness of OSD , 255 means brightest and 0 means darkest **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.**
int translucent	whether translucent when overlay OSD string over active video.
int twinkleInterval	when the value is 1, brightness of OSD will be adjusted according to brightness of background. When background is too bright, brightness of OSD will be lower

automatically, darker, OSD will be brighter, and the twinkle function is closed

unsigned short *Forma1, Format2 — strings overlay to describe the position and sequence of character, the description about them as follows:

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

X and Y means the initiative position of this string in the normal CIF image, and X must be the multiple of 16, and Y can be set in the extent of image height: (0-575) PAL, (0-479) NTSC. CHARN is a parameter of USHORT, it can be ASCII or GB code Characters. When want to display the current time, you can point it as the fixed constant value, and they are as follows:

_OSD_YEAR4	show year time by length of 4, for example: 2004
_OSD_YEAR2	show year time by length of 2, for example: 02
_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 daytime 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 : 星期二
_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 that we must set NULL in the end of format strings, otherwise there will show some error contents.

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 four line CIF image.

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

```
unsigned short Format[] = {16, 19, 'O', 'f', 'f', 'i', 'c', 'e', '\0'};
```

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

```
unsigned short Format[]=
        {8,3,_OSD_YEAR4,':'OSD_MONTH2,':',_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:

```
unsigned short FormatNoDisplay [] = {0, 0, '\0'};
```

Return value:

0—success; -1 – failed;

4.2.41. int SetOsd(int channelHandle, int Enable);

Parameter:

int channelHandle	channel handle
int Enable	enables

Explanation:

Enable or disable the OSD. It can make the currently 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.

Return value:

0—success; -1 – failed;

4.2.42. int SetupDateTime(int channelHandle, SYSTEMTIME *now)

Note: Invalid now

4.2.43. int SetOsdDisplayModeEx(int hChannelHandle,int brightness,int Translucent,int param,int nLineCount,unsigned short **Format)

Parameter:

int hChannelHandle channel handle

int brightness the brightness of the OSD , 255 means brightest and 0 means darkest

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.

int translucent whether translucent when overlay OSD string over active video.

int twinkleInterval when the value is 1, brightness of OSD will be adjusted according to brightness of background. When background is too bright, brightness of OSD will be lower automatically, darker, OSD will be brighter, and the twinkle function is closed

unsigned short *Forma1, Format2 strings overlay to describe the position and sequence of character, the description about them as follows:

int param Bit 0： color auto change as the background；

Bit 16-23： the multiple of vertical zoom

Bit 24-31： the multiple of horizontal zoom

In OSD, the origial size of the ASCII char is **8×16**

For different encode format, the osd will be different. The multiple could be set as the follow diagram:

the multiple of horizontal zoom	the multiple of vertical zoom	Adaptive format	Remark
1	1	4CIF	It will be blurred in other format
1	2	2CIF	It will be blurred when lower than2CIF
2	2	CIF、DCIF	It will be blurred when format is QCIF
4	4	QCIF	It will be bigger in other format
0		Auto set（default）	
others		Set as 2 and 2	

Note: For the location will be changed as the picture format. If the location is changed, some char will exceed the range of the image then it will be not display. But the call will be return ok.

int	nLineCount	the lines of the OSD，the unsigned max is 8
short	**Format	multiple osd，set as the SetOsdDisplayMode.Just is the pointer to the string.

Explanation:

This is the expand of the SetOsdDisplayMode. It could support 8 lines.

Return value:

0—success; -1 – failed;

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

Parameter :	char *FileName	file name
	unsigned char *yuv	image pointer of YUV format
	int BufLen	size of YUV buffer
	int *Width	width returned by YUV image
	int *Height	height returned by YUV image

Explanation : To transfer the 24 bits bmp file to YUV format data, among them the width and length of the BMP should be the multiple of 16, and the max support 128*128 pixels.

Return value:

0—success; -1 – failed;

4.2.45. int SetLogoDisplayMode(int channelHandle, unsigned short ColorKeyR, unsigned short ColorKeyG, unsigned short ColorKeyB, unsigned short bTranslucent, int TwinkleInterval);

Parameter :

int channelHandle channel handle;

unsigned short ColorKeyR, ColorKeyG,, ColorKeyB, color of LOGO image will be completely translucent during display

unsigned short bTranslucent whether do it translucently processing about LOGO image

int TwinkleInterval Set the time about flash . It can be expressed by hex 0xXXYY , and XX is display time and YY is the time of stopping display . When XX and YY all are 0 it can display normally

Explanation:

To set the display mode for the logo.

Return value:

0—success; -1 – failed;

4.2.46. int SetLogo(int channelHandle, int x, int y, int w, int h, unsigned char *yuv);

Parameter :	int channelHandle	channel handle
	int x	top left corner position x(0-703)
	int y	top left corner position y (0-575)
	int w	width (0-128) (the size of it must be the same as the width of the original image)
	int h	height (0-128)

unsigned char *yuv	image pointer of YUV format(YUV422planar)

Explanation:

To set the position and data of logo screen image. User program can call the function LoadYUVFromBmpFile to get YUV data from 24-bits color bmp file (refer to section 4.44). And translucent processing is performed by DSP.

Return value:

0—success; -1 – failed;

4.2.47. int StopLogo(int channelHandle);

Parameter:

int channelHandle	channel handle

Explanation:

To stop logo display;

Return value:

0—success; -1 – failed;

4.2.48. int SetupMask(int channelHandle, RECT *rectList, int iAreas)

Parameter:

int channelHandle	hannel handle
RECT * rectList	the array of the rect
int iAreas,	the count of the rect

Explanation:

Set the video mask. There have up to 32 rect mask. The range of the rect is (0,0,703,575), the width of the rect must align with 16 and the high must align with 8.

Return value:

0—success; -1 – failed;

4.2.49. int StopMask(int channelHandle)

Parameter:

int channelHandle	channel handle

Explanation:

Stop the mask.

Return value:

0—success; -1 – failed;

4.2.50. int AdjustMotionDetectPrecision (int channelHandle, int iGrade, int iFastMotionDetectFps, int iSlowMotionDetectFps);

Parameter:

int channelHandle	channel handle
int iGrade	sensitiveness grade of motion analysis (0-6)
int iFastMotionDetectFps	frame interval of high speed motion detection (0-12) , the value 0 is means there is no need of high motion detection and usually it is 2

int iSlowMotionDetectFps	frame interval of low speed motion detection (>13) , the value of 0 is means there is no need of low speed motion detection

Explanation:

To adjust motion synthesize sensitiveness, and can adjust the sensitiveness of motion detection dynamically during encoding. It also decides the sensitiveness of whole DSP motion synthesizes. It is different from the parameter iThreshold of MotionAnalyze function, the latter mainly used in analyzing some area's motion by host computer. The grade 0 is the most sensitive and grade 6 is the most insensitive. The recommended value is 2;

Notes: The adaptive function described below is invalid in DS-42xx card.

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

Return value:

0—success; -1 – failed;

4.2.51. int SetupMotionDetection(int channelHandle, RECT *rectList, int iAreas) ;

Parameter:

int channelHandle	channel handle
RECT *rectList	rectangle array
int iAreas	number of rectangle （The Max value is 100）

Explanation:

Set motion detection areas. When receive the data frame of marcblock's movement information (PktMotion Detection), call function MotionAnalyzer which can analyze every needed detection areas that is set by SetupMotionDetection. If the threshold of some areas (iThreshold in MotionAnalyzer function) is reached, the finally result will be marked in returned array (iResult in MotionAnalyze function); the rectangle range of DS-40xxHCI is (0, 0, 703, 575).

Return value:

0—success; -1 – failed;

4.2.52. int StartMotionDetection(int channelHandle);

Parameter :

int channelHandle	channel handle

Explanation:

To startup motion detection. Motion detection information can be transmitted by data stream. When we find the frame type is PktMotionDetection we can use MotionAnalyze function to analyze the movement information, and the result is returned by parameter iResult in MotionAnalyzer. We can also analyze by ourself according to the data format given by the SDK refer to section 2.5 for the motion information format;

Notes: Motion detection and the encoding are independent from each other; the user program can run motion detection under the condition of no running encoding program.

Return value:

0—success; -1 – failed;

4.2.53. int StopMotionDetection (int channelHandle);

Parameter:

int channelHandle channel handle

Explanation:

Stop motion detection;

Return value:

0—success; -1 – failed;

4.2.54. int MotionAnalyzer(int channelHandle, char *MotionData, int iThreshold， int *iResult);

Parameter:

int channelHandle channel handle;

char *MotionData pointer of motion vector;

int iThreshold bound of area used in judging movement (0-100);

int *iResult It is the result of motion detection according to the bound of area , and it is a array which size is set with parameter numberOfArea in the function SetupMotionDetection . If the value of some areas is greater than 0 then it is means that there is movement in this area.

Explanation:

Analyze motion detection. Motion detection is performed by DSP. The Frame of pktMotionData given out by DSP is the motion information has been analyzed. The movement of areas is performed by Host computer, and the data source is given by frame of PktMotionData and the result is filled in parameter of iResult. The application can analyze it by itself through the information of motion intensity provided by code stream or the bound analyze calling this function. The data structure of motion intensity is explained in 2.25

Return value:

0—success; -1 – failed;

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

Parameter:

int hChannelHandle channel handle;

int iGrade sensitiveness grade of motion analysis (0-6)

int iFastMotionDetectFps frame interval of high speed motion detection (0-12) , the value 0 is means there is no need of high motion detection and usually it is 2

int iSlowMotionDetectFps frame interval of low speed motion detection (>13) , the value of 0 is means there is no need of low speed motion detection

UINT delay The delay after the last motion detect

RECT *rectList rectangle array

Int iAreas number of rectangle

MOTION_DETECTION_CALLBACK MotionDetectionCallback
call back function when detect some motion
int reserved reserved

MotionDetectionCallback (ULONG channelNumber,
Int bMotionDetected,
Void *context)

Parameter:

ULONG channelNumber the number for the channel；
int bMotionDetected the flag that be sign when motion. If there have motion in areas set, the bMotionDetected will be1，else is 0.
void *context

Explanation:

this is the expand for the motion detection.

Return value:

1— success; -1 – failed;

Notes: The adaptive function described below is invalid in DS-42xx card.

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

4.2.56. int SetAudioPreview(int channelHandle, int bEnable);

Parameter :

int channelHandle channel handle
int bEnable enable

Explanation:

To set audio preview. There is only 1 channel of all the audio inputs to the cards selected outputting to sound board.

Return value:

0—success; -1 – failed;

4.2.57. int GetSoundLevel(int channelHandle)

Parameter:

int channelHandle channel handle

Explanation:

To get current audio input level of the current channel. Attention should be paid that the return value will no be zero even if no audio input is connected due to the background noise.

Return value:

>0 – sound level; -1 - fail;

4.2.58. int RegisterImageStreamCallback(IMAGE_STREAM_CALLBACK, void *context)

Parameter:

IMAGE_STREAM_CALLBACK callback function

callback function: (*IMAGE_STREAM_CALLBACK)
(UINT channelNumber, void *context)

UINT channelNumber	channel no.
void *context	the context provided when call the callback function

Explanation:

Register the user to get the original picture stream function, and user can get the real-time video data on YUV420 format.

Return value:

0 - success; -1 - failed;

Remarks:

DS-42xxHCI/HFVI Card does not support this function

4.2.59. int SetImageStream(int hChannelHandle, int bStart, unsigned int fps,
unsigned width, unsigned height, unsigned char* imageBuf)

Parameter:

int hChannelHandle		Channel handle
int bStart		1:boot capture, 0: Stop capture
unsigned int	fps	frame rate
unsigned int	width	the width of the picture (should be 1.8,1/4,1/2 original size or twice of the width of CIF)
unsigned int	height	the height of picture(should be1.8,1/4,1/2 original size or twice of the height of CIF)
unsigned char	*imageBuffer	The address to store data after capturing the picture

Explanation:

User can boot or stop getting original picture data stream through this function, and this function depends on the host PC processing speed.

Return value:

0—success; -1 – failed;

Remarks:

DS-42xxHCI/HFVI Card does not support this function

4.2.60. int GetOriginalImage(int channelHandle, unsigned char *ImageBuf, int *Size);

Parameter :

int channelHandle	channel handle
unsigned char*ImageBuf	pointer of original image
int *Size	size of original image (before calling it, it is the size of imagebuf, but after calling it, it is the byte factually used)

Explanation:

To get the original image. The original image of DS40xxHCI is the standard 4CIF format (including QCIF encoding), the user program can call SaveYUVToBmpFile to create 24 byte bmp file.

Return value:

0—success; -1 – failed;

Remarks:

DS-42xxHCI card supports CIF raw image;

DS-42xxHFVI card supports 4CIF raw image;

DS-40xxHSI card supports CIF raw image;

DS-40xxHCI/HCSI card supports 4CIF raw image.

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

Parameter :

char *FileName	file name
unsigned char *yuv	image pointer of YUV format
int Width	width of YUV image
int Height	height of YUV image

Explanation :

To transfer the YUV image to BMP file. If it is DS40xxHCI to capture, the Width is 704, Height is 576 (in PAL) or 480(in NTSC); It can be judged according to the size of the buffer.

Return value:

0—success; -1 – failed;

4.2.62. int GetJpegImage(int hChannelHandle,unsigned char *ImageBuf,
unsigned long *Size,unsigned int nQuality);

Parameter :

int channelHandle	channel handle
unsigned char *ImageBuf	the pointer of image buffer
unsigned long* Size	the pointer of the size IN: the buffer size OUT: the size of the image
unsigned int nQuality	the quality of the JPEG(1—100, 100 is best)

Explanation:

Get JPEG image

Return value:

0—success; -1 – failed;

Remarks:

DS-42xxHCI card supports CIF JPEG image;

DS-42xxHFVI card supports 4CIF JPEG image;

DS-40xxHSI card supports CIF JPEG image;

DS-40xxHCI/HCSI card supports 4CIF JPEG image.

4.2.63. int SetupSubChannel(int channelHandle, int iSubChannel);

Parameter:

int channelHandle	channel handle
int iSubChannel	subchannel

Explanation:

We can set some parameters of subchannel 0 and subchannel 1 respectively. The parameters like OSD, LOGO, STREAMTYPE are the same to the 0 or 1 sub channel. This function should be called to set subchannel 0 and subchannel 1 respectively, when we set these parameters, such as Key Frames Intervals, Quantity Value, bit rate control mode, and value of bit rate. By default, the setting is for Sub channel 0.

Return value:

0—success; -1 – failed;

4.2.64. int GetSubChannelStreamType(void *DataBuf, int FrameType);

Parameter:

void *Databuf	data buffer which will be put in
int FrameType	frame type

Explanation:

It is just compatible older version. Now the Frame type is provide directly.

Return value:

0 - other data
1 - File header of master data stream
2 - File header of sub data stream
3 - Video Frame type of master data stream
4 - VideoFrame type of sub data stream
5 - Audio Frame

4.2.65. int GetFramesStatistics(int channelHandle, PFRAMES_STATISTICS framesStatistics);

Parameter:

int channelHandle	channel handle
PFRAMES_STATISTICS ramesStatistics	statistic information of frame(refer to section 2.2)

Explanation:

Get statistic information of frame;

Return value:

0—success; -1 – failed;

4.2.66. int CaptureIFrame(int channelHandle);

Parameter:

int channelHandle	channel handle

Explanation:

Force the current frame encode as I frame. We can read this I frame from data stream and used in the internet transmission independently.

Return value:

0—success; -1 – failed;

4.2.67. int SetDeInterlace(int hChannelHandle,UINT mode,UINT level)

Parameter:

int channelHandle	channel handle
UINT mode：	0: this channel don't use deinterlace 1: use deinterlace and set the Level, DS-42xx Card do not support this mode 2: SDK will use the default arithmetic, and parameter Level will be invalid.
UINT Level：	0－10，0 is the most weakness, and default is 5. Only valid when mode is 1.

Explanation:

Set the deinterlace and intensity.

Return value:

0—success; -1 – failed;

4.2.68. int ResetDSP(int dspNumber)

Note: It is invalid now.

4.2.69. int SetWatchDog(unsigned int boardNumber,int bEnable)

Parameter:

int boardNumber	the index for the board

Explanation:

Set the watchdog. There have 4 pin in the DS4016HCS. It must connect the reset in PC with it then link to the motherboard.

Return value:

0—success; -1 – failed;

V4.2 New add

4.2.70. int StartVideoPreviewEx(int hChannelHandle, PREVIEWCONFIG* pPreviewConf, UINT useSyncSem, UINT mode)

Parameter:

int channelHandle	channel handle;
PREVIEWCONFIG* pPreviewConf	the struct refer to section 2.5
UINT useSyncSem	1 means use the semaphore to sync the preview. The application init a semaphore，then provide to sdk by pPreviewConf->SyncSem. The sdk will post this sem after a new frame ready. 0 means use the timer to sync the preview
mode	NORMAL_SIZE, D1_SIZE, DCIF_SIZE, QCIF_SIZE, MINI_SIZE

The definition as follows:

```
#define NORMAL_SIZE     0    /* just not change */
#define D1_SIZE         1    /* 704 x 576 */
#define DCIF_SIZE       2    /* 528 x 288 */
```

#define CIF_SIZE	3	/* 352 x 288 */
#define QCIF_SIZE	4	/* 176 x 144 */
#define MINI_SIZE	5	/* 88 x 72 */

The argument mode being NORMAL_SIZE, the function works same as StartVideoPreview which means that the width and height of image in the memory is adjusted by the display area. While being other value, the size of image in the memory keeps unchanged. Note that the number of the D1-sized images in memory should be less than 4. To avoid the PCI bus over pressed, SDK will decrease the quality of image to reduce the transfer data.

V5.1 SDK New add

4.2.71. int __stdcall SetEncoderAudioOutput(UINT nEncodeChannel, int bEnable, UINT nOutputChannel)

Parameters:

UINT nEncodeChannel	Encoding channel index
int bEnable	Enable or not
UINT nOutputChannel	Output channel that can be 0 only.

Return Values:

If the function succeeds, the return value is 0.

If the function fails, the return value is -1.

Remarks:

Only DS-42xxHFVI Card support this function

5. Decoding Card SDK

5.1. API of Encoding Card

Note: If without special description. The return values always is 0 if calling is success and is -1 if calling is failed. You could call GetLastErrorNum to get the failed info.

Decoding card does not support DS-42xx Card

5.1.1. int HW_InitDecDevice(long *pDeviceTotal);

Explanation:

Init the system.

Parameter:

pDeviceTotal the decode channle

5.1.2. int HW_ReleaseDecDevice();

Explanation:

close the system, should be called before exit.

5.1.3. int HW_ChannelOpen(long nChannelNum,int* phChannel);

Explanation:

Open the decode channel and get the channel handler

Parameter:

nChannelNum	channel number, begin from 0
phChannel	channel handler

5.1.4. int HW_ChannelClose(int hChannel);

Explanation:

Close the decode channel;

Parameter:

hChannel	channel handler

5.1.5. int HW_OpenStream(int hChannel,char* pFileHead,int nHeadSize);

Explanation:

Open the interface for the Stream mode（just as open the file）;

Parameter:

hChannel	channel handler
pFileHead	the data for the file head;
nHeadSize	the length for the file head;

5.1.6. int HW_CloseStream(int hChannel);

Explanation:

Close the Stream mode;

Parameter:

hChannel channel handler;

5.1.7. int HW_InputData(int hChannel,char* pBuf,int nSize);

Explanation:

Input the data that must be called after open the stream

Parameter:

hChannel	channel handler;
pBuf	the address for the data buffer
nSize	the size for the data buffer

Return:

nSize if call success, else -1

5.1.8. int HW_OpenFile(int hChannel,char* sFileName);

Explanation:

Open the file that want to be decoded;

Parameter:

hChannel	channel handler;
sFileName	the file name

5.1.9. int HW_CloseFile(int hChannel);

Explanation:

Close the decoding file;

Parameter:

hChannel	channel handler;

5.1.10. int HW_Play(int hChannel);

Explanation:

Start decoding;

Parameter:

hChannel	channel handler;

5.1.11. int HW_Stop(int hChannel);

Explanation:

Stop decoding;

Parameter:

hChannel	channel handler;

5.1.12. int HW_Pause(int hChannel ,ULONG bPause);

Explanation:

Pause decoding;

Parameter:

hChannel	channel handler;
bPause	1--pause，0--continue;

5.1.13. int HW_PlaySound(int hChannel);

Explanation:

Open the audio decoding, it is closed in default

Parameter:

hChannel	channel handler

5.1.14. int HW_StopSound(int hChannel);

Explanation:

Close the audio decoding;

Parameter:

hChannel	channel handler

5.1.15. int HW_SetVolume(int hChannel,ULONG nVlome);

Explanation:

Adjust the volume for the audio;

Parameter:

hChannel	channel handler
nVlome	0~0xffff。

5.1.16. int HW_StartCapFile(int hChannel,char* sFileName);

Explanation:

Capture the decoding data and save as…

Parameter:

hChannel	channel handler
sFileName	file name

5.1.17. int HW_StopCapFile(int hChannel);

Explanation:

Stop capturing

Parameter:

hChannel	channel handler

5.1.18. int HW_GetPictureSize(int hChannel,ULONG* pWidth, ULONG* pHeight);

Explanation:

Get the size of the picture

Parameter:

hChannel	channel handler
*pWidth	width for the picture

*pHeight	height for the picture

5.1.19. int HW_GetYV12Image(int hChannel, char* pBuffer, ULONG nSize);

Explanation:

Capture current image and save to buffer(format is YV12)

Parameter:

hChannel	channel handler
pBuffer	the buffer that to be saved.The size must bigger than （*pWidth）*（*pHeight）*3/2
nSize	the size of the buffer

5.1.20. int HW_ConvertToBmpFile(char * pBuf,ULONG nSize,ULONG nWidth,ULONG nHeight,char *sFileName,ULONG nReserved);

Explanation:

covert the YV12 image to the BMP file

Parameter:

pBuf	the buffer for the image
nSize	the size of the buffer
nWidth	the width of the image
nHeight	the height of the image
sFileName	the file name of the BMP
nReserved	reserved

5.1.21. int HW_GetSpeed(int hChannel,long *pSpeed);

Explanation:

Get the speed for decodeing；

Parameter:

hChannel	channel handler；
*pSpeed	speed: -4 ~ 4

5.1.22. int HW_SetSpeed(int hChannel,long nSpeed);

Explanation:

Set the decoding speed；

-4 is stop，then call with HW_Pause(hChannel,0) could play just one frame

-3 ------ 1/8

-2 ------ 1/4

-1 ------ 1/2

0 ------ normal

1 ------ 2 double fast

2 ------ 4 multiple fast

3 -------8 multiple fast

4 ------- the highest speed

Parameter:

hChannel channel handler;
nSpeed the speed for decoding

5.1.23. int HW_SetPlayPos(int hChannel,ULONG nPos);

Explanation:

Set the position to decode

Parameter:

hChannel channel handler;
nPos the percent of the file length (0 – 100)

5.1.24. int HW_GetPlayPos(int hChannel,ULONG* pPos);

Explanation:

Get the position that now decoding

Parameter:

hChannel channel handler;
*pPos the percent of the file (0 ~ 100)

5.1.25. int HW_SetJumpInterval(int hChannel,ULONG nSecond);

Explanation:

Set the interval to jump

Parameter:

hChannel channel handler;
nSecond the interval as second

5.1.26. int HW_Jump(int hChannel,ULONG nDirection);

Explanation:

Set the direction for jumping;

Parameter:

hChannel channel handler;
nDirection JUMP_FORWARD: forward;
JUMP_BACKWARD: back;

5.1.27. int HW_GetVersion(PHW_VERSION pVersion);

Explanation:

Get the information for the version

Parameter:

pVersion information, describe as follow

```
typedef struct {
        ULONG DspVersion, DspBuildNum;
        ULONG DriverVersion, DriverBuildNum;
        ULONG SDKVersion, SDKBuildNum;
}HW_VERSION, *PHW_VERSION;
```

5.1.28. int HW_GetCurrentFrameRate(int hChannel,ULONG* pFrameRate);

Explanation:

Get the frame rate of the decoding

Parameter:

hChannel	channel handler;
* pFrameRate	frame rate

5.1.29. int HW_GetCurrentFrameNum(int hChannel,ULONG* pFrameNum);

Explanation:

Get the frame index that now decoding

Parameter:

hChannel	channel handler;
*pFrameNum	frame index

5.1.30. int HW_GetFileTotalFrames(int hChannel,ULONG* pTotalFrames);

Explanation:

Get the total frame number of the file

Parameter:

hChannel	channel handler;
* pTotalFrames	total frame number

5.1.31. int HW_GetFileTime(int hChannel, ULONG* pFileTime);

Explanation:

Get the total time of the file

Parameter:

hChannel	channel handler;
* pFileTime	total time（ms）;

5.1.32. int HW_GetCurrentFrameTime(int hChannel,ULONG* pFrameTime);

Explanation:

Get the current time

Parameter:

hChannel;	channel handler
* pFrameTime	times（ms）;

5.1.33. int HW_GetPlayedFrames(int hChannel,ULONG *pDecVFrames);

Explanation:

Get the frame counts that have decoded

Parameter:

hChannel	channel handler;
*pDecVFrames	frame counts

5.1.34. int HW_SetFileEndMsg(int hChannel, sem_t* nMsg);

Explanation:

Register the semaphore that will be post when the file finish decode

Parameter:

hChannel	channel handler
nMsg	semaphore

5.1.35. int HW_SetStreamOpenMode(int hChannel,ULONG nMode);

Explanation:

Set the change mode of the stream decoding

Parameter:

hChannel channel handler

nMode 0 ~ 5 , 0 means don't change just as decode file， 1 ~ 5 will change the smooth and delay. If set 5, the stream will be more smooth but delay more

5.1.36. int HW_GetStreamOpenMode(int hChannel,ULONG *pMode);

Explanation:

Get the current change mode

Parameter:

hChannel	channel handler
*pMode	0 ~ 5 ；

5.1.37. int HW_SetAudioPreview(int hChannel, UINT bEnable);

Explanation:

Set audio preview. The Link just as the HCI There will be only one channel could be open in the time. The Sdk will close other channels. Call after HW_ChannelOpen and HW_PlaySound

Parameter:

bEnable 1—open, 0—close

5.1.38. int HW_StartDecVgaDisplay(int hChannel, PREVIEWCONFIG* pPreviewConf, UINT useSyncSem);

Explanation:

Start video display on the Vga；Just as the function StartVideoPreview() Please refer to the Section4.16

Parameter:

pPreviewConf Please refer to the section 4.16

useSyncSem If set, the sdk will post pPreviewConf->SyncSem when decode one frame

5.1.39. int HW_StopDecChanVgaDisplay(int hChannel);

Explanation:

Stop video display

Parameter:

hChannel channel handler

5.1.40. int SetDisplayStandard (UINT nDisplayChannel,VideoStandard_t VideoStandard)

Parameter:

UINT nDisplayChannel the index for the display channel

VideoStandard_t VideoStandard video standard

Explanation:

Set the video standard for the video out

5.1.41. int SetDisplayRegion (UINT nDisplayChannel,UINT nRegionCount,

REGION_PARAM *pParam,UINT nReserved)

Parameter:

UINT nDisplayChannel the index for the display channel

UINT nRegionCount the count of the areas

REGION_PARAM *pParam parameter of the areas

UINT nReserved reserved

```
typedef struct
{
    UINT left;          the left of the area, align with 16
    UINT top;           the top of the area, align with 8
    UINT width;         the width of the area, align with 16
    UINT height;        the height of the area, align with 8
    UINT r;             the red weight of the background
UINT g                  the green weight of the background
UINT b;                 the blue weight of the background
UINT param;             expand parameter
}REGION_PARAM;
#define MAX_DISPLAY_REGION  16  //The max count of the area that could display
```

Explanation:

There could partition the display to some area.

Return：

ErrorCodeNotSupport：the resource of the DSP is limited. The dsp could not partition more area. That must shrink the area. Every display channel could support one 4CIF size area + 2 QCIF areas

ErrorCodeInvalidDevice：nDisplayChannel is overflow

ErrorCodeInvalidArgument：nRegionCount is overflow，or pParam is error or the area overstep the limit.

5.1.42. int ClearDisplayRegion (UINT nDisplayChannel,UINT nRegionFlag)

Parameter:

UINT nDisplayChannel the index for the display channel

UINT	RegionFlag	the area that want to clead

Explanation:

To clean the display area and display the background color. Bit0－Bit15：just as area1－area16，If bitn is 1，then relevant area be clear

Note: If the current area is just displaying, this api is invalid

5.1.43. int SetDisplayRegionPosition (UINT nDisplayChannel,UINT nRegion, UINT nLeft,UINT nTop)

Parameter:

UINT	nDisplayChannel	the index for the display channel
UINT	nRegion	the area that want to adjust
UINT	nLeft,UINT nTop	the position that want to put

Explanation:

Adjust the position that display

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

Parameter:

UINT	nDisplayChannel	the index for the display channel
UINT	nRegion	the area that want to fill
unsigned char *	pImage	the point that YUV420 image

Explanation:

Fill the area by YUV420 image. The size of the Image must equal to the area.

Note: If the current area is just displaying, this api is invalid

5.1.45. int SetDecoderAudioOutput(UINT nDecodeChannel,UINT bOpen, UINT nOutputChannel)

Parameter:

UINT	nDecodeChannel	the index for the decode channel
UINT	bOpen	1 – open; 0-- close
UINT	nOutputChannel	the index for the display channel

Explanation:

Set the decoding audio output. Put the nDecodeChannel decoding audio to nOutputChannel output. There have 2 audio output channel in the DS-4002MDI.So the nOutputChannel must be 0 or 1，There have 4 audio output channel in the 4004MDI. So the nOutputChannel must be 0，1，2 or 3.The Sdk will close the former audio output before start new data output

5.1.46. int SetDecoderVideoOutput(UINT nDecodeChannel,UINT nPort,UINT bOpen,UINT nDisplayChannel,UINT nDisplayRegion,UINT nReserved)

Parameter:

UINT	nDecodeChannel	the index for the decode channel
UINT	nPort	the output port for the decode

UINT	bOpen	1 – open; 0 -- close
UINT	nDisplayChannel	the index for the display channel
UINT	nDisplayRegion	the area that want to dispaly
UINT	nReserved	reserved

Explanation:

Set the decodeing video to display (just in the current board). Put the decoding video from the nPort in the nDecodeChannel (that is in the current board)to the nDisplayRegion in the nDisplayChannel. There have 2 port in the decode channel, so the nPort must be 0or 1. There have 4 display channel in the DS400xMDI, so the nDisplayChannel must be 0,1,2 or 3

5.1.47. int SetDecoderVideoExtOutput(UINT nDecodeChannel,UINT nPort,UINT bOpen,UINT nDisplayChannel,UINT nDisplayRegion,UINT nReserved)

Parameter:

UINT	nDecodeChannel	the index for the decode channel
UINT	nPort	the output port for the decode
UINT	bOpen	1 – open; 0 -- close
UINT	nDisplayChannel	the index for the display channel
UINT	nDisplayRegion	the area that want to dispaly
UINT	nFrameRate	Frame rate of video output

Explanation:

This is the expand for the SetDecoderVideoOutput. It is Matrix output. It could put the decodingdata to the other boards. The index for the nDisplayChannel is index from the whole MDI board.

5.1.48. int SetEncoderVideoExtOutput(UINT nEncodeChannel,UINT nPort,int bOpen,UINT nDisplayChannel,UINT nDisplayRegion,UINT nReserved);

Parameter:

UINT	nEncodeChannel	the index for the encode channel
UINT	nPort	the output port for the decode
UINT	bOpen	1 – open; 0 -- close
UINT	nDisplayChannel	the index for the display channel
UINT	nDisplayRegion	the area that want to dispaly
UINT	nReserved	reserved

Explanation:

Set the encoding data to display output.（Matrix output）。Put the video data from the nPort in the nEncodeChannel to the nDisplayRegion in the nDisplayChannel. The encode channel only support 2 port output. So the nPort must be 0 or 1.

This API is fit for combination use of DS-40xxHCI and MDI card, or DS-42xxHFVI card. But DS-42xxHFVI card only output single channel and nPort should be 0.

New API in v4.1 SDK

5.1.49. int HW_SetFileRef(int hChannel, UINT bEnable,FILE_REF_DONE_CALLBACK FileRefDoneCallback);

Parameter:

int nChannel	channel handler;
UINT bEnable	enable or disable
FileRefDoneCallback	The callback after the index created

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

UINT nChannel	the channel number
UINT nSize	the size of the Index(invalid now)

Explanation:

Set the index for the decoding file. Please call before HW_OpenFile()

5.1.50. int HW_GetFileAbsoluteTime(int hChannel, SYSTEMTIME *pStartTime, SYSTEMTIME *pEndTime);

Parameter:

int hChannel	channel handler;
SYSTEMTIME* pStartTime	begin time for the recoding
SYSTEMTIME* pEndTime	end time for the recoding

Explanation:

Get the absolute time for the recoding file.

Note: The millisecond in the struct SYSTEMTIME is invalid now and always is 0.

5.1.51. int HW_GetCurrentAbsoluteTime(int hChannel,SYSTEMTIME *pTime);

Parameter:

int hChannel	channel handler;
SYSTEMTIEM* pTime	the current absolute time of the recording

Explanation:

Get the current absolute time of the recording file

Note: The millisecond in the struct SYSTEMTIME is invalid now and always is 0.

5.1.52. int HW_LocateByAbsoluteTime(int hChannel,SYSTEMTIME time);

Parameter:

int hChannel	channel handler;
SYSTEMTIME time	the absolute time that want to locate

Explanation:

Locate the playing by absolute time. It will be valid after set the index and on file mode

Note: the millisecond in the struct SYSTEMTIME is invalid now and always is 0.

And the begin time could get by calling HW_GetFileAbsoluteTime()

5.1.53. int HW_LocateByFrameNumber(int hChannel,UINT frmNum);

Parameter:

int hChannel	channel handler
UINT frmNum	the frame number That want to locate

Explanation:

Locate the playing by the frame number. It will be valid after set the index and on file mode. The total frame numbers could get by calling HW_GetFileTotalFrames().

New API in v4.2 SDK

5.1.54. int HW_InputDataByFrame(int hChannel,char* pBuf,int nSize)

Parameter:

hChannel	channel handler;
pBuf	the address for the data buffer
nSize	the size for the data buffer

Explanation:

The function should be called after open the stream, and processes in more real-time

Return: nSize if call success, else -1

5.1.55. HW_SetDecoderPostProcess(int hChannel, UINT param)

Explanation:

Avoid the flash of the image while decoding.

Parameter:

int hChannel	channel handler	
UINT param	0: end the process	1: start the process

New API in v4.3 SDK

5.1.56. int RegisterDecoderVideoCaptureCallback(DECODER_VIDEO_CAPTURE_CALLBACK DecoderVideoCaptureCallback,void *context)

Explanation:

Invoke this API to register a callback function for getting decoded video data.

Parameters:

DecoderVideoCaptureCallback	Callback function;
void *context	Context when invoke callback function

Callback function description:

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

UINT nChannelNumber	Channel handle
void *DataBuf	Address of buffer
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 grabbed (ms)
SYSTEMTIME *pFrameAbsoluteTime	Absolute time of current frame grabbed
void *context	Context when invoke callback function.

Return value:

Return 0 if successful, return error code if failed.

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

Explanation:

Start or stop grabbing decoded data in yuv420 format.

Parameters:

HANDLE hChannel	Channel handle
BOOL bStart	Start or Stop grabbing.
UINT param	Reserved, value is 0

Return Value:

Return 0 if successful, return error code if failed.