PV-260 Serial board Encode System Programmer Manual （FOR Linux）

（2005-1-13 Release version 2.2）

PV-260 video and audio compression 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.

The SDK of PV-260 series card is made up of encode system SDK, network SDK and player SDK. This manual especially describes encode system SDK, as to the other SDK you can refer to the relevant documents. Encode system SDK is the local record 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 (H.264 Demo Version4.0) 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 PV-260 series compression card is independent from compression. It can be done without compression. It is valuable that the frame rate can be transformed. When moving, record as high frame rate (25 F/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.

Compared with PV-240、245 & 250 generated in the early period, the main characters of PV-260 (4.0 version) are as following:

1. Compared with PV-240, 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, vocality is smoother.
5. Will support high resolution ratio 4CIF (704*576) video compression function.

6. New add screen MASK function, support max 32 regions
7. 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
8. The preview manner is changed. Realized overlay preview using SDL library. Multiple channel preview will consume more CPU load.

SDK interface is completely the same as those of PV-240、245 & 250 serial boards and more functions are added. Developed internet applications can be migrated very fast.

Note: **.** Reboot your computer after the driver installed.

. Demo program included in our SDK must run on XWINDOW.

1. Definition and description for error code using in this SDK:

Error code	Explanation
ErrorCodeDSPUninit	DSP not initialed
ErrorCodeDSPNotReady	DSPnot ready
ErrorCodeDSPLoadFail	DSP load fail
ErrorCodeEncodeChannelOverflow	Encode channel number overflow
ErrorCodeDecodeChannelOverflow	Decode channel number overflow
ErrorCodeBoardOverflow	Board number overflow
ErrorCodeDSPHexBlockLenOverflow	DSP program length overflow
ErrorCodeDSPProgramCheckoutFail	DSP program checkout fail
ErrorCodeDSPInitRecheckFail	Check DSP program fail
ErrorCodeDSPBusy	DSP busy
ErrorCodeNoCardInstalled	No cards installed
ErrorCodeMemLocateFail	Allocate memory failed
ErrorCodeDuplicateSN	Duplicate serial number
ErrorCodeDSPCmdInvalid	Invalid DSP command
ErrorCodeChannelOutofRange	Invalid channel number
ErrorCodeInvalidEncodeChannel	Invalid encode channel number
ErrorCodeInvalidArgument	Invalid argument
ErrorCodeNotSupport	This function not supported
ErrorCodeCreateYUVOverlayFail	Create YUV surface failed

2. Definitions for data types:

2.1 video preview output formats:

vdfRGB16	16 bits RGB video preview format
vdfRGB24	24 bits RGB video preview format
vdfYUV422Planar	YUV422 video preview format

2.2 definitions for frame types

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

2.3 definitions for video standard

StandardNone	No video signal
StandardNTSC	NTSC format
StandardPAL	PAL format

3. Definition for data structure

3.1 definition for extraordinary ability

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

3.2 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;
```

3.3 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;
```

3.4 definition of motion detection data

PV-260 serial boards offer motion intensity information to deal with motion detection. When setting the motion detection areas, use 32x32as one unit, resolution using 4CIF (704x5760) there re 22 blots 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. Definition of functions

4.1 int InitDSPs();

Explanation: This function initializes every board and must be finished before any other operation to be done. If the returned value is 0 means initialization is failed, the reasons perhaps is failed in finding related DSP software module or no board is installed. Its related function is DeInitDSPs() ;

Return value: 0 - initialization DSP failed

>0 – number of successfully initialization DSP

4.2 int DeInitDSPs();

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

Return value: 0 – close DSP success.

4.3 int ChannelOpen(int ChannelNum, STREAM_READ_CALLBACK streamReadCallback);

Parameter : int ChannelNum: // channel no (0-n)

STREAM_READ_CALLBACK StreamReadCallBack: //function pointer of stream data treated callback function(see 5.1 section)

Explanation: Open channels and get operation handles. All operations related with this channel must use this handle;

Return value: >0 - valid handle,

-1 – fopen channel fail

4.4 int ChannelClose(int channelHandle) ;

Parameter: int channelHandle channel handle;

Explanation: close channel and release relative resource;

Return value: 0 – success

-1 - fail

4.5 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 that those initializations of DSP are failed.

4.6 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.

4.7 int StartVideoPreview(int channelHandle, SDL_Surface *display, SDL_Rect dstRect);

Parameter: int channelHandle channel handle;

SDL_Surface *display SDL suface, preview picture will displayed o it.

SDL_Rect dstRect rectangle area in SDL suface;

Explanation: Start video preview. **Adopting the preview mode by SDL overlay.** Displaying multiple preview pictures must disable hardware acceleration, or pictures will twinkle.

putenv("SDL_VIDEO_YUV_HWACCEL=0"). Suggest that CPU is higher than 2GHz with stable system.

Return value: 0 – success;

-1 - fail;

4.8 int StopVideoPreview(int channelHandle);

Parameter: int channelInfo channel handle;

Explanation: Stop video preview;

Return value: 0 – success;

-1 - fail;

4.9 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 - fail;

4.10 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 value (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 - fail;

4.11 void GetSDKVersion (PVERSION_INFO VersionInfo);

Parameter: PVERSION_INFO VersionInfo pointer of VERSION_INFO;

Explanation: get the SDK version used currently. 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;

4.12 int GetCapability (int channelHandle, CHANNEL_CAPABILITY *Capability);

Parameter: int channelHandle channel handle
CHANNEL_CAPABILITY *Capability refer to section 3.1

Explanation: To get the information of special function of the board;

Return value: 0 – success;
-1 - fail;

4.13 int GetLastErrorNum();

Explanation: Get the last error information for SDK;

Return value: return error num (consult the definition for error code using in this SDK)

4.14 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: Set stream type;

Return value: 0 – success;
-1 - fail;

4.15 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;

4.16 int GetFramesStatistics(int channelHandle, PFRAMES_STATISTICS framesStatistics);

Parameter: int channelHandle channel handle
PFRAMES_STATISTICS ramesStatistics statistic information of frame
PFRAMES_STATISTICS frame structure

Explanation: get statistic information of frame;

Return value: 0 – success;
-1 - fail;

4.17 int SetupMotionDetection(int channelHandle, RECT *rectList, int numberOfAreas) ;

Parameter: int channelHandle channel handle
RECT *rectList rectangle array
int numberOfAreas number of rectangle

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 PV-260 is (0,0,703,575).

Return value: 0 – success;
-1 - fail;

4.18 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 ourselves according to the data format given by the SDK refer to section 3.4 and 3.5 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 - fail;

4.19 int GetBoardInfo(int channelHandle, int *BoardType, int *SerialNo);

Parameter:	int channelHandle	channel handle
	Int *BoardType	PV-260 is 2
	char *SerialNo	ID number of card: content is ASCII number of card sequence, SerialNo[0] corresponding to the highest, SerialNo[11] corresponding to the lowest. For instance: "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.
Explanation:	To get hardware information of board	
Return value:	0 – success;	
	-1 - fail;	

4.20 int StopMotionDetection (int channelHandle);

Parameter:	int channelHandle	channel handle
Explanation:	stop motion detection;	
Return value:	0 – success;	
	-1 - fail;	

4.21 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 - fail;	

4.22 int StopVideoCapture(int channelHandle);

Parameter:	int channelHandle	channel handle
Explanation:	stop data intercept;	
Return value:	0 – success;	
	-1 - fail;	

4.23 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)
	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 - fail;

4.24 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 Bframe 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 - fail;

4.25 int SetOsd(int channelHandle, int Enable);

Parameter: int channelHandle channel handle
int Enable enables

Explanation: Set OSD (Overlay String Display) mode. 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 - fail;

4.26 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 - fail;

4.27 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 OSD screen image. User program can call the function LoadYUVFromBmpFile to get YUV data from 24-bits color bmp file (refer to section 4.31). And translucent processing is performed by DSP.

Return value: 0 – success;
-1 - fail;

4.28 int StopLogo(int channelHandle);

Parameter: int channelHandle channel handle

Explanation: To stop OSD display;

Return value: 0 – success;
-1 - fail;

4.29 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 16m, and the max support 128*128 pels.

Return value: 0 – success;
-1 - fail;

4.30 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 PV-260 to capture, the Width is 704, Height is 576 (in PAL) or 480(in NTSC); if it is PV-245 & 250, the Width perhaps is 352 or 176, and Height is 288, 240, 144 or 120. It can be judged according to the size of the buffer.

Return value: 0 – success;
-1 - fail;

4.31 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 PV-260 is the standard 4CIF format (including QCIF encoding), the user program can call SaveYUVToBmpFile to create 24 byte bmp file. While the original image of PV-245 & 250 is CIF image format.

Return value: 0 – success;
-1 - fail;

4.32 int GetVideoSignal(int channelHandle);

Parameter : int channelHandle channel handle

Explanation : To get the information of connect video signal . It can used in alarming for video loss;

Return value : 1 - no video signal
0 - valid video signal
-1 – invalid parameter

4.33 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 (refer to Ondata Ready part of Demo data source), and the result is filled in parameter of iResult. The 2.0 version's motion analysis is based on the motion intensity provided by DSP but not using motion vector any more. The sensibility and reliability is progressed greatly, the user's software can analyze 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 3.4;

Return value: 0 – success;
-1 - fail;

4.34 int AdjustMotionDetectPrecision (int channelHandle, int iGrade, int iFastMotionDetectFps, int iSlowMotionDetectFps);

Parameter: int channelHandle channel handle

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;

Return value: 0 – success;

-1 - fail;

4.35 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 - fail;

4.36 int SetEncoderPictureFormat(int channelHandle, PictureForamt_t PictureFormat) ;

Parameter: int channelHandle channel handle

PictureForamt_t PictureFormat size of coding image (4CIF, 2CIF,CIF, QCIF CIFQ and CIFQCIF)

Explanation: Set the encoding format of the current channel. It must be called after stop recording.

4CIF format is added into the 1.0 version. PV-260 boards support 4CIF encoding format. When some channel is set in the CIFQCIF format, there will be two kinds of data stream: CIF and QCIF sent by DSP after booting the recording. User program should process separately. The original encoding format will not be changed. (The resolution of QQCIF is 96*80.)

User program can also call directly function Set SubEncoderPictureFormat() to set the encoding format of the sub-channel of some channel. And call the function StartSbuVideoCapture()/StopSubVideoCapture() to realize the start and stop of the sub-channel.

Return value: 0 – success;

-1 - fail;

4.37 int SetupBitrateControl(int channelHandle, int MaxBps) ;

Parameter: int channelHandle channel handle;
Int MaxBps the most baud rate (more than 10000)

Explanation: 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 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 - fail;

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

Parameter : int channelHandle channel handle;
int 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

Return value: 0 – success;
-1 - fail;

4.39 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
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’};

4.40 int SetVideoStandard(int channelHandle, VideoStandard_t videoStandard)

Parameter： int channelHandle channel handle;

VideoStandard_t videoStandard video standard

Remark: 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. While it is necessary to call this function if we boot the system without connecting the cameras (after it connects to NTSC format cameras) or we change the different format cameras in the process.

Return value: 0 – success;

-1 - fail;

4.41 int ResetDSP(int dspNumber)

Parameter：int dspNumber index number of DSPs

Remark: To reset some DSP system. Pay attention to be cautious to call this function. After the confirmation that DSP deadlock or malfunction cannot be restored through the software, and reset DSP after closed the relevant resources.

4.42 int GetSoundLevel(int channelHandle)

Parameter：int channelHandle channel handle

Remark: 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;

4.43SetBitrateControlMode(int channelHandle, BitrateControlType_t brc)

Parameter：int channelHandle channel handle

BitrateControlType_t brc bitrate control mode, brVBR and br CBR

Remark: 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 bitrate will be the value set by SetBitrateControl and the picture quality is adjust automatically to maintain constant bitrate.

Return value: 0 – success;
-1 - fail;

4.44 int SetupSubChannel(int channelHandle, int iSubChannel);

Parameter: int channelHandle channel handle
int iSubChannel subchannel

Remark: This function should be cooperated with the mode of CIFQCIF and CIFQQCIF. When the mode of CIFQCIF is selected, the encode of subchannel 0 is CIF and subchannel 1 is QCIF (QQCIF), we can set some parameters of subchannel 0 and subchannel 1 respectively. The parameters like Key Frames Intervals, 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 Quantity Value, bitrate control mode, and value of bitrate. By default, the setting is for subchannel 0.

Return value: 0 – success;
-1 - fail;

4.45int GetSubChannelStreamType(void *DataBuf, int FrameType);

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

Remark: This function should be cooperated with the mode of CIFQCIF and CIFQQCIF. When the mode of CIFQCIF is selected, the encode of subchannel 0 is CIF and subchannel 1 is QCIF (QQCIF). When record is started; DSP will deliver two kinds of data streams, which are CIF and QCIF (QQCIF). We get the type of data stream after calling of this function, which is only used by application.

Return value: 0 - other data
1 - File header of CIF data stream
2 - File header of QCIF (QQCIF) data stream
3 - Video Frame type of CIF data stream
4 – Video Frame type of QCIF (QQCIF) data stream
5 - Audio Frame

New functions for PV-260 card:

4.46int Setup Mask(int channelHandle, RECT *rectList, int iAreas

Parameters: int channelHandle channel handle
RECT*rectList rectangle list
Int iAreas the number of the rectangle

Explanation: The functions to boot screen mask provided by PV-260 serial boards. The max can be set is 32, and the range of the rectangle is (0, 0, 703, 575). The width of the rectangle is the same as 16, and the height is 8.

Return value: 0 – success;
-1 - fail;

4.47int StopMask(int channelHandle)

Parameter: int channelHandel: channel handle

Explanation: The function to stop the screen mask provided by PV-250 & 260 serial boards

Return value: 0 – success;
-1 - fail;

4.48int SetSubEncoderPictureFormat(int channelHandel, PictureFormat_t pictureFormat)

Parameter: int channelHandle channel handle
PictureFormat_t picture Format the size of encoding picture (CIF, QCIF)

Explanation: This function is the encoding format in the sub-channel when setting the double encoding mode. If cooperated with the latter functions, you can realize the boot and stop recording in the sub-channel at you will.

Return value: 0 – success;
-1 - fail;

4.49int StartSubVideoCapture(int channelHandle)

Parameter: int channelHandle channel handle

Explanation: to start the sub-channel video capture of some channel

Return value: 0 – success;

-1 - fail;

4.50int StopSubVideoCapture(int channelHandle)

Parameter: int channelHandle channel handle

Explanation: To stop the sub-channel video capture in some channel

Return value: 0 – success;

-1 - fail;

4.51int SetupDateTime(int channelHandle, SYSTEMTIME *now)

Parameter: int channelHandle channel handle

SYSTEMTIME *now the pointer to SYSTEMTIME

Explanation: Set the time of OSD, it can be used to verify the time on the net.

After calling this function, the function of default local verifying time by SetOsd( ) will be shielded.

4.52 int SetInputVideoPosition(int channelHandle, int x, int y)

Parameter: int channelHandle channel handle

int x the X axis of the coordinate, the defaulted value is 8，

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.

Return value: 0 – success;

-1 - fail;

4.53 int SetEncodePictureFormat(int channelHandle, PictureFormat_t PictureFormat) ;

Parameter: int channelHandle channel handle

PictureFormat_t PictureFormat size of coding image (4CIF, 2CIF, CIF, QCIF CIFQ and CIFQCIF)

Explanation: See 4.36 SetEncoderPictureFormat(), to be compatible with PV-250 card.

5. Other explain

5.1 The callback function StreamReadCallBack must be defined as follows:

```
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
int Length	//frame length
int FrameNum	//frame index

you can deal with stream data in this function ,such as record, motion detect and so on.

In this function you can receive frame type as follows:

PktSysHeader	system header

(After StartVideoCapture(),DSP will sends a PktSysHeader Frame，every record File must start by this frame, or the file may not be replayed.)

PktIFrames	I Frame
PktPFrames	P Frame
PktBBPFrames	BBP Frame
PktAudioFrames	Audio Frame

(Those five 5 types frame above is stream data and can be writing to record file.)

PktMotionDetection	motion detection data

(After StartMotionDetection()，DSP will sends this PktMotionDetection Frame)

PktSFrames

(After CaptureIFrame()，DSP will sends this Iframe.)