MPEGIOPro SDK User Manual

For Writing Software to
Live Capture & Stream Multi-Channel Multi-Card Hardware-Encoded MPEG2/4/1 Video with OSD

Version 1.0.4

Table of Contents

1. Introduction:

MPEGIOPro SDK (Software Development ToolKit) is for developing application software using **MPEGIOPro** hardware MPEG2/4/1 encoder card.

For details on the **MPEGIOPro PCI card, including its hardware and device driver installation**, please refer to the "**MPEGIOPro User Manual**", here is a summary of the PCI card's main features:

- **Realtime Capture 4-Channel superior-quality video/audio as MPEG2, MPEG4(H.263) or MPEG1 files**
- **Live Stream All Channels' Video/Audio to Any User-Defined TCP/IP Address in Unicast or Multicast**
- **All channels can capture/stream full D1 size (720X576 or 720X480 Pixel) video simultaneously**
- **Support up to 4 Cards in One PC to Simultaneously Record & Stream 16-Channel Full D1-Size MPEG Video**
- **Each Input Channel Can Record Video Only, Stream Video Only, or Record and Stream Simultaneously**
- **All Channels' Recording and Streaming Parameters Can Be Individually Defined**
- **Wide-Range Capture/Streaming Data Rates 128Kbps~15Mbps for MPEG1, MPEG2 & MPEG4 formats**
- **Various Video Capture/Streaming Frame Sizes from 176X120 to 720X576-Pixels**
- **Multiple encoding Frame Rate supported: 23.976fps, 24 fps, 25fps, 29.97fps, 30fps, 50fps, 59.94fps, 60fps**
- **Allow Multiple Aspect Ratio Encoding & Streaming: 4:3, 16:9(Wide-Screen), and Square PEL**
- **Forever Video & Audio Sync during video preview, capture, streaming and in the recorded files**
- **Video Capture Process Can Seamlessly Split Files either manually or at user-defined file size or time**
- **No single frame lost during video splitting: combining all split files will form the exact original video stream**
- **Instantly Start Video Capture at User's Request (manually or programmatically) without lengthy delays**
- **Start Video Capture as Video Signal Appears**
- **Stops Video Capture after Video Signal Disappears for a period of user-defined time**
- **Pause and Resume recording without closing recording file**
- **Realtime Overlay Text/Time onto Recorded MPEG File, Streamed Video and Previewed Video**
- **Overlaid Text Can Have Different Size, Position, Duration, Black/White Colour, User-Definable Frames**
- **Text Overlay Can Contain Any Text, or Current Encoding File Name, Video Bit Rate, MPEG Type, etc**
- **Simultaneous Multi-line Text Overlay: max. 450 characters (15 rows X 30 columns) can be overlaid**
- **Motion Detect is supported by on-board hardware for 9 separate regions on every input channel**
- **Motion Detect returns quantitative Motion Vector Sums indicating the strength of each region's motion activity**
- **Capture still images in various formats: JPEG, BMP, GIF, TIFF, and PNG**
- **Capture/Stream DVD-Compliant or arbitrary MPEG2, MPEG4 or MPEG1 video files**
- **Captured/Streamed DVD-Compliant files can be burnt to DVD movie immediately without re-compression**
- **Capture Audio-Only MPEG files in small sizes without video components**
- **Realtime on-screen preview video and audio always in sync with incoming video signal**
- **PAL and NTSC video format are automatically detected and switched**
- **Composite/BNC Video Input**
- **Stereo Audio Input**
- **Hardware-encoding chip built-in, low system resources needed: Pentium4 3GHz can record 16 Channels**
- **Any-length Timer Recording and Date-Time-Scheduled Recording with user-definable file names**
- **Support live colour change on incoming video**
- **Automatic email-out of warning messages on signal loss, disk full and start/stop conditions**
- **Recorded video file name can contain multiple fields inc. recording date, time, video length, video format, etc.**
- **Loop Disk and Split File Number Reset functions to automatically free disk space on pre-set conditions**
- **Works on Windows XP, Vista or Windows 7 PCs with PCI slot**

2. MPEGIOPro.DLL & MPEGIOPro SDK

The **MPEGIOPro SDK** is based on a dynamic linking library **MPEGIOPro.DLL,** which supports all the **SDK** function calls the application software might use. **MPEGIOPro.DLL** will need to be copied to the target PC running your application software. The **MPEGIOPro.DLL** works under Windows XP, Vista and Windows 7.

3. MPEGIO SDK Function List:

-- const char * **MPEGIOPro_SDKVer** (void);
Function: Return the version of the MPEGIOPro SDK as a string.

-- ULONG **MPEGIOPro_Init**(bool ignoreOSD = false);
Function: Initialize the **MPEGIOPro** hardware
Parameters:
ignoreOSD: If true, then OSD will not be displayed although can still be set up in the SDK, default is false(OSD will be displayed). If this is set to true then the SDK load (start up) will be faster.
Return: the number of total video channels, < 1 means no channel or hardware not ready

-- ULONG **MPEGIOPro_Uninit**(void);
Function: Un-initialize the **MPEGIOPro** hardware
Return: 0 for success, non-zero for failure

-- BOOL **MPEGIOPro_StartDevice**(ULONG nDev, bool ignoreOSD = false);
Function: Start one video channel on the **MPEGIOPro** card
Parameters:
nDev: the channel number (first channel is 0) to be started, must be < the total channel number returned by **MPEGIOPro_Init**
ignoreOSD: If true, then OSD will not be displayed although can still be set up in the SDK, default is false(OSD will be displayed).
Return: TRUE (non-zero) for success, FALSE(zero) for failure

-- BOOL **MPEGIOPro_StopDevice**(ULONG nDev);
Function: Stop one video channel that has been started by **MPEGIOPro_StartDevice**
Parameters:
nDev: the channel number (first channel is 0) to be stopped, must be < the total channel number returned by **MPEGIOPro_Init**
Return: TRUE (non-zero) for success, FALSE(zero) for failure

-- BOOL **MPEGIOPro_IsChannelStarted** (ULONG nDev);
Function: Test if a channel has been started
Parameters:
nDev: the channel number (first channel is 0) to be tested, must be < the total channel number returned by **MPEGIOPro_Init**
Return: TRUE (non-zero) if the channel has been started

-- ULONG **MPEGIOPro_CreateScreen**(void);
Function: Create a DirectX display surface for displaying live video overlay
Return: Zero for success

-- ULONG **MPEGIOPro_ReleaseScreen**(void);
Function: Release theDirectX display surface created by **MPEGIOPro_CreateScreen**
Return: Zero for success

-- ULONG **MPEGIOPro_UpdateScreen**(void);
Function: Reclaim a DirectX display surface that was created by **MPEGIOPro_CreateScreen**

but later lost because of other application software or user intervention

Return: Zero for success

Note: On PCs with some graphics cards such as Nvidia GeForce 7X/8X00, this function needs to be called constantly to get live video overlay on PC's screen: this can be done in a timer of about 30~40 ms interval.

-- ULONG **MPEGIOPro_CleanScreen**(RECT *pRect);

Function: Clean a rectangle area on the display surface created by **MPEGIOPro_CreateScreen**

Parameters:

pRect: the rectangle area to be cleaned. NULL means the entire display surface.

Return: Zero for success

-- ULONG **MPEGIOPro_SetOverlayColorKey** (COLORREF ColorKey);

Function: Set the colour key for the video display: only video displayed on this background brush colour can be seen.

Parameters:

ColorKey: the RGB colour used to create background brush for the video preview window

Return: Zero for success

Note: Use a rarely seen colour as ColorKey, such as pink (RGB(255,0,255))

-- ULONG **MPEGIOPro_SetScreenScale**(ULONG nXScale, ULONG nYScale);

Function: Set video display screen scale: scale 1000:1000 is for displaying the video in the original resolution of 704X576-Pixel for PAL, and 704X480-Pixel for NTSC. To display video in a new resolution **newXPixel** X **newYPixel**, use this formula to calculate the values for nXScale and nYScale:

nXScale = (newXPixel * 1000) / 704

nYScale = (newYPixel * 1000) / 576 for PAL, or

nYScale = (newYPixel * 1000) / 480 for NTSC

Example 1: to display live PAL video in an area of 1024X768 – Pixels, call

MPEGIOPro_SetScreenScale((1024 * 1000)/704, (768 * 1000)/576);

Example 2: to display live NTSC video in an area of 800X600 – Pixels, call

MPEGIOPro_SetScreenScale((800 * 1000)/704, (600 * 1000)/480);

Parameters:

nXScale: new resolution's X scale as calculated above

nYScale: new resolution's Y scale as calculated above

Return: Zero for success

Note: To set scale for video display in an area <= 704X576 (PAL) or 704 X 480(NTSC) pixels, use

nXScale = 1000

nYScale = 1000

-- ULONG **MPEGIOPro_CleanWindow**(HWND hWnd, RECT rect, COLORREF color);

Function: Clean up rectangle area "**rect**" in window **hWnd** using background colour "**color**"

hWnd:handle of the window whose rectangle area will be cleaned

rect: rectangle area in hWnd to be cleaned

color: colour used to clean up the "**rect**" area

Return: Zero for success

Note: **1. color** normally should equal to the **ColorKey** used for MPEGIOPro_SetOverlayColorKey()

2. this function will clean up any content inside rectangle area **"rect"**

3. hWnd should be the window MPEGIOPro uses to display video

-- ULONG **MPEGIOPro_StartPreview**(ULONG nDev, RECT *pWindow);

Function: Start Video Preview for Channel nDev

Parameters:

nDev: the channel number to start preview, must be < the total channels found

pWindow: address of the rectangle to display the video, in screen-co-ordinates

Return: Zero for success

Note:

1. the **right** & **bottom** values of pWindow must be smaller than the current screen width and height

2. after calling this function the live video should appear in the area defined by pWindow

-- ULONG **MPEGIOPro_StopPreview**(ULONG nDev);

Function: Stop Video Preview for Channel nDev

Parameters:

nDev: the channel number to stop preview, must be < the total channels found

Return: Zero for success

-- ULONG **MPEGIOPro_GetVideoParam**(ULONG nDev, ULONG * pStandard, RLS_VIDEOPARAM * pVideoParam);

Function: Retrieve video standard and colour parameters of a video channel.

Parameters:

nDev: the channel number, must be < the total channels found

pStandard: pointer to variable to hold the returned video standard.

Video standard values: 1 for PAL, 2 for NTSC

pVideoParam: pointer to structure holding the returned colour parameters

RLS_VIDEOPARAM structure is defined as:

```
typedef struct _RLS_VIDEOPARAM
{
    int nBrightness;
    int nContrast;
    int nSaturation;
    int nHue;
}RLS_VIDEOPARAM;
```

Return: Zero for success

-- ULONG **MPEGIOPro_SetVideoParam**(ULONG nDev, RLS_VIDEOPARAM * pVideoParam);

Function: Set colour parameters for one video channel.

Parameters:

nDev: the channel number, must be < the total channels found

pVideoParam: pointer to structure holding the returned colour parameters, as defined in **MPEGIOPro_GetVideoParam**

Return: Zero for success

-- ULONG **MPEGIOPro_SetVideoOffset**(ULONG nDev, ULONG nXOffset, ULONG nYOffset);

Function: Set video preview's X and Y offset

Parameters:

nDev: the channel number to set offset, must be < the total channels found

nXOffset: the number of pixels video display will cutoff from the left edge

nYOffset: the number of pixels video display will cutoff from the top

Return: Zero for success

Note:

1. The default offsets are zero for **nXOffset** and **nYOffset**
2. offset set will only affect video preview display, not the recorded video

-- ULONG **MPEGIOPro_CapturePicture**(ULONG nDev, UCHAR * pBuffer, int *pBufSize, int nWidth, int nHeight, ULONG tFormat);

Function: Capture one frame of pixels from video
Parameters:
nDev: the channel number, must be < the total channels found
pBuffer: buffer to hold the captured frame pixel data
pBufSize: pointer to variable holding the size of pBuffer, this function returns the actual size of the captured frame data in this variable
nWidth: width of the picture to capture in pixels, must be within 48 ~ 720
nHeight: height of the picture to capture in pixels, within 32~576 for PAL, 32~480 for NTSC
tFormat: must be 0
Return: Zero for success
Note: the captured picture is in 24-bit bitmap colour format, upper left corner pixel first, running left to right, top-down, max.720X576 pixel for PAL, 720X480 pixel for NTSC.

-- ULONG **MPEGIOPro_StartAudioMonitor** (ULONG nDev);
Function: Start audio for one channel
Parameters:
nDev: the channel number, must be < the total channels found
Return: Zero for success

-- ULONG **MPEGIOPro_StopAudioMonitor** (ULONG nDev);
Function: Stop audio for one channel
Parameters:
nDev: the channel number, must be < the total channels found
Return: Zero for success

-- ULONG **MPEGIOPro_RegisterCallback**(MPEGIOPRO_CALLBACK callbackProc);
Function: Register a function to be called back by **MPEGIOPro.DLL** to pass back encoding data
Parameters:
nDev: the channel number, must be < the total channels found
callbackProc: callback function name, MPEGIOPRO_CALLBACK is defined as
typedef void (CALLBACK *MPEGIOPRO_CALLBACK)
(int nDev, UCHAR * pData, int nSize, ULONG DataType, void *pUserParam);
where **dDev** is the channel number,
pData is the encoded video data
nSize is the length of data held by **pData,** in number of bytes
DataType is the encoded data type: 5 for "Motion Detect", others for MPEG data)
pUserParam points to the user-defined variable passed over to **MPEGIOPro_StartEncode**(see below), normally used to pass channel/application specific data to the callback function
Return: Zero for success
Note: the successfully registered function callbackProc will be called by **MPEGIOPro.dll** if any video

channel has called **MPEGIOPro_StartEncode**(see below) successfully and encoded data on that channel is ready for processing

-- ULONG **MPEGIOPro_StartEncode**(ULONG nDev, void * pUserParam,
HWND hwnd, UINT errMsg);

Function: Start encoding for a video channel (start the on-board video encoding hardware)

Parameters:

nDev: the channel number, must be < the total channels found

pUserParam: user-defined variable passed over to the callback function registered by calling **MPEGIOPro_RegisterCallback**, see description in **MPEGIOPro_RegisterCallback** above

hwnd: window handle to which the SDK will send back encoding error message

errMsg: the message MPEGIOPro SDK will send back to window hwnd when encoding error happens, message "WM_MPEGIOPRO_ENCODE_ERR" can be used here.

Return: Zero for success

Note: If parameter hwnd is not a valid window then no message will be sent by the SDK

-- ULONG **MPEGIOPro_StopEncode**(ULONG nDev);

Function: Stop encoding for a video channel (stop the on-board video encoding hardware)

Parameters:

nDev: the channel number, must be < the total channels found

Return: Zero for success

Note: Once encoding is stopped, the callback function will not be called for that particular channel

-- ULONG **MPEGIOPro_StartMotionDetect** (ULONG nDev);

Function: Start Motion Detect for a video channel

Parameters:

nDev: the channel number, must be < the total channels found

Return: Zero for success

Note: 1. Before calling this function, **MPEGIOPro_SetMotionDetect**() must be called successfully to setup each motion detect region's starting X/Y and Width/Height

2. MPEGIOPRo must be in encoding mode(**MPEGIOPro_StartEncode** is called) before calling this function.

-- ULONG **MPEGIOPro_StopMotionDetect** (ULONG nDev);

Function: Stop Motion Detect for a video channel

Parameters:

nDev: the channel number, must be < the total channels found

Return: Zero for success

Note: If encoding has started(**MPEGIOPro_StartEncode** is called) and **MPEGIOPro_StartMotionDetect** () has been called successfully, this function must be called before stopping encoding(calling **MPEGIOPro_StopEncode**).

-- ULONG **MPEGIOPro_SetEncodeStreamType** (ULONG nDev, ULONG StreamType);

Function: Set MPEG encoding type

Parameters:

nDev: the channel number, must be < the total channels found

StreamType: 1 for MPEG1, 2 for MPEG2, 4 for MPEG4(H.263)

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeAudioBitRate**(ULONG nDev, ULONG nAudioBitRate);

Function: Set MPEG audio encoding bit rate

Parameters:

nDev: the channel number, must be < the total channels found

nAudioBitRate: in bit per sec., valid values are:
32000, 48000, 56000, 64000, 80000, 96000, 112000, 128000,
160000, 192000, 224000, 256000, 320000, 384000

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeAudioSamplingRate**(ULONG nDev,
ULONG nAudioSamplingRate);

Function: Set MPEG audio encoding sampling rate

Parameters:

nDev: the channel number, must be < the total channels found

nAudioSamplingRate: in Hz., valid values are: 32000, 44100, 48000

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeAudioMode**(ULONG nDev, ULONG nAudioMode);

Function: Set MPEG audio encoding mode

Parameters:

nDev: the channel number, must be < the total channels found

nAudioMode: 0- Stereo (default), 1-Joint Stereo, 2-dual channel, 3-single channel

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeAudioLayer**(ULONG nDev, ULONG nAudioLayer);

Function: Set MPEG audio encoding layer

Parameters:

nDev: the channel number, must be < the total channels found

nAudioLayer: 1-MPEG1Layer1, 2-MPEG1Layer2(Default), 3-MPEG1Layer3

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeAudioStreamType**(ULONG nDev,
ULONG nAudioStreamType);

Function: Set MPEG audio encoding stream type

Parameters:

nDev: the channel number, must be < the total channels found

nAudioStreamType: 1-MPEG1(default), 2-MPEG2, 6-AC3

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeVideoBitRate**(ULONG nDev, BOOL bVBR,
ULONG nVideoBitRate, ULONG nVideoMaxBitRate);

Function: Set MPEG video encoding bit rate

Parameters:

nDev: the channel number, must be < the total channels found

bVBR: true for Variable Bit Rate(VBR) Encoding, false for Constant Bit Rate(CBR) encoding

nVideoBitRate: average encoding bit rate, in bit per sec. Unit

nVideoMaxBitRate: maximum encoding bit rate, in bit per sec., only valid if **bVBR** is true

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeVideoSize**(ULONG nDev,
ULONG nWidth, ULONG nHeight);

Function: Set MPEG video encoding frame size

Parameters:

nDev: the channel number, must be < the total channels found

nWidth: encoding frame width, in pixels

nHeight: encoding frame height, in pixels

Return: Zero for success

Note: valid combinations for nWidth X nHeight:

PAL: 176 X 144, 352 X 288, 480 X 576, 704 X 576, 720 X 576

NTSC: 176 X 120, 352 X 240, 480 X 480, 704 X 480, 720 X 480

-- ULONG **MPEGIOPro_SetEncodeVideoStandard**(ULONG nDev, ULONG nStandard);

Function: Set MPEG video encoding frame size

Parameters:

nDev: the channel number, must be < the total channels found

nStandard: 1 for PAL, 2 for NTSC

Return: Zero for success

-- ULONG **MPEGIOPro_SetEncodeFrameMN**(ULONG nDev, ULONG M, ULONG N);

Function: Set MPEG video encoding intervals of the I,B,P frames in each GOP

Parameters:

nDev: the channel number, must be < the total channels found

M: frame distance between reference frames, 0 ~ 3

N: number of frames in GOP, 1 ~ 256

Return: Zero for success

Note:

1. If M is not 0, then N must be a multiple of M. If M = 0, then N must be set to 1.
2. M and N parameters set the group_of_picture (GOP) structure:
   6 <= N < 256 in integer multiples of M for M = 3
   4 <= N < 256 in integer multiples of M for M = 2
   2 <= N < 256 in integer multiples of M for M = 1
   N= 1, M = 0 (I-Frames only)
   For example,
   If N=15 and M=3, the GOP structure is I B B P B B P B B P B B P B B.
   If M = 0, the GOP structure is I (encoder will generate I frames only)
   If M = 1, the GOP structure is IP
   If M = 2, the GOP structure is IBP
   If M = 3, the GOP structure is IBBP
3. Default is N = 15, M = 3

-- ULONG **MPEGIOPro_SetEncodeFrameSkip**(ULONG nDev, ULONG nSkip);

Function: Set number of skipped over frames during encoding

Parameters:

nDev: the channel number, must be < the total channels found

nSkip: frame skip number:

0 means not dropping any frame, making frame rate 25/sec PAL, 30/sec. NTSC
1 means drop 1 frame every 2 frames, making frame rate 25/2 PAL, 30/2 NTSC
2 means drop 2 frames every 3 frames, making frame rate 25/3 PAL, 30/3 NTSC
3 means drop 3 frames every 4 frames, making frame rate 25/4 PAL, 30/4 NTSC

Return: Zero for success

Note: Default is nSkip = 0 (not dropping any frame)

-- ULONG **MPEGIOPro_DefaultEncodeParam**(ULONG nDev);

Function: Make the encoding parameters set by all above **MPEGIOPro_SetXXX**() calls effective

Parameters:

nDev: the channel number, must be < the total channels found

Return: Zero for success

Note:

1. This function must be called after calling all the above functions
from **MPEGIOPro_SetEncodeStreamType**
to **MPEGIOPro_SetEncodeFrameSkip**

2. This function must be called before all following **MPEGIOPro_Set…()** functions :
from **MPEGIOPro_ SetVOBFormat ()**
to **MPEGIOPro_SetAspectRatio()**

-- ULONG **MPEGIOPro_SetEncodeFrameRate**(ULONG nDev, int frcode);

Function: Set video encoding frame rate

Parameters:

nDev: the channel number, must be < the total channels found

frcode: frame rate value:

1: 24000/1001 (23.976) frame per sec.
2: 24 fps, for film
3: 25 fps, for PAL default
4: 29.97 fps, for NTSC default
5: 30 fps, NTSC Drop Frame
6: 50 fps, Double Frame Rate for PAL
7: 59.94 fps, Double Frame Rate for NTSC
8: 60 fps, Double Frame Rate Drop Frame NTSC

Return: Zero for success

Note: Default value for PAL is 3 (25 fps), for NTSC is 4 (29.97 fps).

-- ULONG **MPEGIOPro_SetVOBFormat**(ULONG nDev, BOOL vob);

Function: Set Video Encoding VOB Format

Parameters:

nDev: the channel number, must be < the total channels found

vob: 1 for VOB format(default), 0 for StandardPS format

Return: Zero for success

Note:

1. non-VOB format (vob = 0) will cause MPEGIOPro.dll to crash if signal is lost during encoding

2. This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- ULONG **MPEGIOPro_SetBlackScreen**(ULONG nDev, ULONG black);

Function: Set Video Encoding Black Screen Status

Parameters:

nDev: the channel number, must be < the total channels found
black: non-zero means all encoded frames are black
Return: Zero for success
Note: This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- ULONG **MPEGIOPro_SetBadSync**(ULONG nDev, ULONG badSync);
Function: Set Video Encoding Bad Sync Status
Parameters:
nDev: the channel number, must be < the total channels found
badSync: non-zero means enable bad sync: useful for bad quality input video signal
Return: Zero for success
Note: This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- ULONG **MPEGIOPro_SetClosedGOP**(ULONG nDev, ULONG closedGOP);
Function: Set Video Encoding Closed GOP status
Parameters:
nDev: the channel number, must be < the total channels found
closedGOP: non-zero means closed GOP : prediction is based only on pictures in the present GOP, not on pictures in previous or next GOP.
Return: Zero for success
Note: This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- ULONG **MPEGIOPro_SetHSP**(ULONG nDev, ULONG hsp);
Function: Set new Horizontal Start Position for encoding video frame
Parameters:
nDev: the channel number, must be < the total channels found
hsp: new horizontal start position (in pixels)
Return: Zero for success
Note: This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- ULONG **MPEGIOPro_SetVSP**(ULONG nDev, ULONG vhsp);
Function: Set new Vertical Start Position for encoding video frame
Parameters:
nDev: the channel number, must be < the total channels found
vsp: new vertical start position (in pixels)
Return: Zero for success
Note: This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- ULONG **MPEGIOPro_SetAspectRatio**(ULONG nDev, ULONG newar,
int MPEGType, int VideoType);
Function: Set new encoding video aspect ratio
Parameters:
nDev: the channel number, must be < the total channels found
newar: new aspect ratio:
1 -- Square pels
2 -- 4:3 display
3 -- 16:9 display

4 -- 2.21:1 display

MPEGType: 1 for MPEG1, 2 for MPEG2, 4 for MPEG4

VideoType: 1 for PAL, 2 for NTSC

Return: Zero for success

Note: This function must be called after calling **MPEGIOPro_DefaultEncodeParam**()

-- BOOL **MPEGIOPro_textOSDEnabe**(int nDev, bool enable);

Function: Enable/Disable On Screen Display for a channel

Parameters:

nDev: the channel number, must be < the total channels found

enable: TRUE(default setting) to enable OSD, false to disable OSD

Return: TRUE(non-zero) for success

Note: after displaying many OSDs on a channel, calling this function with **enable=FALSE** will hide all OSD result instantly, while calling this function with **enable=TRUE** will show all OSD result again: useful for toggling a channel's overlay

-- ULONG **MPEGIOPro_OSDTime**(ULONG nDev, BOOL bEnable,
ULONG row, ULONG column);

Function: Display OSD as Date/Time

Parameters:

nDev: the channel number, must be < the total channels found

eEnable: TRUE to enable OSD date/time, false to disable OSD date/time display

row: X-position for display, must be between 0 ~ 14

column: Y-position for display, must be between 0 ~ 29

Return: Zero for success

Note:

1. Each channel can have at most one Date/Time OSD at any particular time. Subsequent calls to this function with a different row/column stop previous calls' display of date/time.

2. Date/Time is always displayed as white colour

-- BOOL **MPEGIOPro_textOSDEraseRow**(int nDev, int row);

Function: Erase OSD display on one row

Parameters:

nDev: the channel number, must be < the total channels found

row: row to erase OSD , must be 0 ~ 14

Return: TRUE(non-zero) for success

Note: to erase arbitrary length of characters on a row, just display equal amount of spaces (blank character, 0x20) by calling function **MPEGIOPro_textOSDDisplay.**

-- BOOL **MPEGIOPro_textOSDDisplay**(int nDev, char *string, int row, int column,
bool doubleWidth, bool doubleHeight, bool blinking, char colour);

Function: Display OSD text

Parameters:

nDev: the channel number, must be < the total channels found

string: null-terminated string of text to display

row: X-position for display, must be between 0 ~ 14

column: Y-position for display, must be between 0 ~ 29

doubleWidth: TRUE to display double width text

doubleHeight:TRUE to display double height text
blinking: TRUE to blink text
colour: 1(or odd number < 8) is white, 0(or even value < 8)
Return: TRUE(non-zero) for success

-- BOOL **MPEGIOPro_textUserFont**(int nDev, char fontIndex, char *fontBitmap);
Function: Set User- defined font for OSD for ASCII code 248 ~ 255
Parameters:
nDev: the channel number, must be < the total channels found
fontIndex: must be between 0 ~ 7, representing ASCII code 248 ~ 255
fontBitmap: must be 36 bytes, each two bytes indicate a row in the font bitmap with the last 4 bits discarded(i.e. only the left-most/highest 12 bits are used to form the bitmap)
Return: TRUE(non-zero) for success
Note: Each font is 18 Row by 12 Pixels, each row is represented by 2 Bytes with the last 4 bits unused

-- BOOL **MPEGIOPro_textSetFrame**(int nDevth, int frameNum, bool enable = false,
char rowStartAddr = 0, char rowEndAddr = 0,
char colStartAddr = 0, char colEndAddr = 0, char colour = 1);
Function: Set frame area: maximum 4 frames can be set up as background frames under OSD text
Parameters:
nDev: the channel number, must be < the total channels found
frameNum: frame number, within 1 ~ 4
enable: TRUE to enable, FALSE to disable this frame
rowStartAddr: frame's start row, 0 ~ 14
rowEndAddr: frame's end row, 0 ~ 14, Must be >= rowStartAddr
colStartAddr: frame's start column, 0 ~ 29
colEndAddr: frame's end column, 0 ~ 29, Must be >= colStartAddr
colour: 1 for white, 0 for black
Return: TRUE(non-zero) for success

-- BOOL **MPEGIOPro_textFrameStartPos**(int nDev, char HORD, char VERTD);
Function: set text characters' starting display position inside the video frame(default is [15, 4] pixels)
Parameters:
nDev: the channel number, must be < the total channels found
HORD: horizontal position, must be within [0, 255], default is 15
VERTD: vertical position, must be within [0, 255], default is 4
Return: TRUE(non-zero) for success

-- BOOL **MPEGIOPro_textVerticalHeight**(int nDev, char height);
Function: set text characters' vertical height (default is 18 pixels)
Parameters:
nDev: the channel number, must be < the total channels found
height: text height, must be between [18, 71]
Return: TRUE(non-zero) for success
Note: default height is 18

-- BOOL **MPEGIOPro_textSetRowSpacing**(int nDev, char spacing);
Function: set pixel distance between rows of text
Parameters:
nDev: the channel number, must be < the total channels found

spacing: in pixels, must be between 0, 31
Return: TRUE(non-zero) for success
Note: default spacing is 0

-- BOOL **MPEGIOPro_textEnableShadow**(int nDev, bool enableShadow, bool enableBorder);
Function: set OSD text shadow and border styles
Parameters:
nDev: the channel number, must be < the total channels found
enableShadow: enable (TRUE) or disable (FALSE) shadow
enableBorder: enable (TRUE) or disable (FALSE) border
Return: TRUE(non-zero) for success
Note: default are not enabled for both

-- ULONG **MPEGIOPro_ResetVideoChip** (int nDev);
Function: Hardware reset the on-board Video Decoder (Analogue-Digital Converter) Chip
Parameters:
nDev: the channel number, must be < the total channels found
Return: Zero for success
Note: This function must be used when video preview is in progress, and is usually used in extreme situations when the MPEGIOPro card is facing difficult incoming signal/noise etc, e.g. when video suddenly becomes very unclear/noisy. Do not use this function when recording/streaming is in progress or breakups will appear in the encoded video stream.

-- ULONG **MPEGIOPro_SetMirror**(ULONG nDev, bool mirror);
Function: Flip the video preview horizontally 180-degree (left-to-right, right-to-left)
Parameters:
nDev: the channel number, must be < the total channels found
mirror: TRUE to flip(mirror the video), FALSE to restore back the normal display
Return: Zero for success
Note: the flipped video is only for live preview, will not affect the recorded/streamed video

-- int **MPEGIOPro_OpenFile**(char *fileName);
Function: Open a disk file for writing
Parameters:
fileName: full file name inc. path
Return: opened file's handle if >= 0, negative value means failure
Note: returned file handle can be passed to **MPEGIOPro_CloseFile**() and **MPEGIOPro_WriteFile**()

-- int **MPEGIOPro_CloseFile**(int handle);
Function: Close a disk file opened by **MPEGIOPro_OpenFile**
Parameters:
handle: file handle returned by a successful **MPEGIOPro_OpenFile** call.
Return: 0 for success, non-zero for failure

-- int **MPEGIOPro_WriteFile**(int handle, , int pStr, int nSize);
Function: Write data to disk file
Parameters:
handle: file handle returned by a successful **MPEGIOPro_OpenFile** call
pStr: integer form of the address of a memory area holding data to be written
nSize: number of bytes in the memory area pointed to by **pStr**

Return: equal to nSize means nSize of bytes have been written successfully, otherwise failure

Note: This function can be used in programming languages where passing string address is difficult, e.g., in callback functions inside VisualBasic, in conjunction with **MPEGIOPro_OpenFile** and **MPEGIOPro_CloseFile** function calls.

-- ULONG **MPEGIOPro_NewInitialTimeCode**(ULONG nDev, int drop_frame_flag, int hour, int min, int sec, int frame);

Function: Set new initial video encoding time code

Parameters:

nDev: the channel number, must be < the total channels found

drop_frame_flag: 1 for drop frame frame-rate, 0 for not drop frame frame-rate (NTSC only)

hour: hour value in the time code

min: minute value in the time code

sec: second value in the time code

frame: frame number in the time code

Return: Zero for success

Note: If called during encoding or streaming time, this function must be followed by calling function **MPEGIOPro_UpdateEncodingParam()** or new time code will not become effective. If this function is never called the initial encoding time code defaults to all zeroes.

-- ULONG **MPEGIOPro_UpdateEncodingParam** (ULONG nDev, UINT yes);

Function: Make some encoding parameter change effective during encoding time

Parameters:

nDev: the channel number, must be < the total channels found

yes: 1 for updating, 0 for not updating

Return: Zero for success.

Note: During encoding time, some functions such as **MPEGIOPro_NewInitialTimeCode** (), must be followed by calling this function to make their parameter change to become effective. However, calling this function during preview time still returns success although that has no effect at all.

-- ULONG **MPEGIOPro_SetMotionDetect**(ULONG nDev, RECT *pRect, int *pThreshold, int nNumOfRect);

Function: Set up Motion Detect area (regions).

Parameters:

nDev: the channel number, must be < the total channels found

pRect: point to an array of RECT structures each has 4 X 32-bit integers(left, top, right, bottom), indicating the **nNumOfRect** motion detect regions' upper left (left, top) and lower right (right, bottom) screen co-ordinates.

pThreshold: point to an array of **nNumOfRect** 32-bit integers indicating the motion detect regions' threshold, each valid threshold is between 1 ~ 65535 (0xFFFF)

nNumOfRect: indicating how many motion detect regions to be set up: must be between 1 and 9. If this is smaller than 9, only the first nNumOfRect members in array pRect and pThreshold hold valid values.

Return: Zero for success.

Note: **1.**MPEGIOPro allows 1 to 9 regions on screen to be setup for motion detection. Each region's upper-left and lower right X/Y co-ordinates must be multiples of 16, and must be within the maximum video frame size (720X576-pixel for PAL, 720X480-pixel for NTSC). The width and height of each region must be at least 16-pixel. Regions can overlay.

2.Each region has a motion detect **Threshold**, as indicated by the members pointed to by

parameter **pThreshold:** each threshold is between 1 ~ 65535, indicating the sensitivity of motion detect: larger value means less sensitive: more activity is needed for MPEGIOPro card to think there is motion, smaller value means more sensitive: less activity will make MPEGIOPro to think there is motion. Zero value is illegal

3.This function must be called successfully before calling **MPEGIOPro_StartMotionDetect().**

-- bool **MPEGIOPro_GetMotionDetectResult**(ULONG nDev, WORD *MDResult);

Function: Receive Motion Detect results from the MPEGIOPro hardware

Parameters:

nDev: the channel number, must be < the total channels found

MDResult: must point to an array of 9 16-bit unsigned integers, receiving the corresponding motion detect region's motion vector: bigger value means more motion is detected in the region. If less than 9 regions were setup by **MPEGIOPro_SetMotionDetect** then extra members in **MDResult** receive meaningless value.

Return: true for success: some regions have motion detected and their motion vectors are non-zero.

Note: After encoding and motion detect both are started (**MPEGIOPro_StartEncode** and **MPEGIOPro_StartMotionDetect** are called), call this function from within the callback function (see **MPEGIOPro_RegisterCallback**) when the callback function's "DataType" vale is 5 (indicating "motion detect"), if this function returns true, then some values in **MDResult** array will have non-zero value indicating motion.

-- ULONG **MPEGIOPro_ResetEncoderIC** (ULONG nDev);

Function: Reset the MPEG Encoding Chip

Parameters:

nDev: the channel number, must be < the total channels found

Return: Zero for success.

Note: Use this function cautiously, use it only when a channel is receiving encoding error, e.g. when the "errMsg**"** is received from the SDK as passed to MPEGIOPro_StartEncode in its last parameter, or when resetting card hardware is necessary. Avoid using this function too frequently since that could cause unnecessary wearing out of the IC.

4. Encoding OSD

MPEGIOPro allows text overlay on all video channels in the following manners:

Individual Video Channels can set their own OSD differently
Colour: black or white
Max. Horizontal Text Columns: 30
Max. Vertical Text Rows: 15
Text Features: Double Width, Double Height, Blinking, Shadow, Border
Built-in Font Characters: ASCII code 0 ~ 247
User-Definable Font Characters: ASCII code 248 ~ 255
Number of Frames Displayable on Screen: 4 (can be used as background for text or stand alone rectangles)
Frame Colours: black or white

During video preview, recording and streaming, all OSD text can be turned on or off instantly by **MPEGIOPro.dll** function calls. Date and Time are provided as special OSD strings.

5. Live Streaming

MPEGIOPro allows un-modified raw MPEG video streams to be sent out in multicast/unicast UDP protocol to any TCP/IP address and port, therefore allowing any third-party MPEG video streaming client, such as Inventa **MPEGIO PCI card, VideoLan vlc.exe** software etc**,** to receive, record and display the streamed out video. No **MPEGIOPro** hardware or software is required to receive the streamed MPEG video.

Application software can collect the video data to stream out from within the callback function(callbackProc) whose pointer was passed to **MPEGIOPro.dll** in the **MPEGIOPro_RegisterCallback** (callbackProc) function call.

Live video streaming is implemented with the following functions:

-- ULONG **MPEGIOPro_NetworkInit**(void);
Function: Initialise TCP/IP network of this PC
Return: 0 means success, otherwise failure in which case all other streaming functions cannot work.
Note: 1. MPEGIOPro_NetworkInit must precede all other streaming functions listed below
2. Each **MPEGIOPro_NetworkInit()** must be matched by a **MPEGIOPro_NetworkEnd()**

-- void **MPEGIOPro_NetworkEnd**(void);
Function: De-Initialise TCP/IP network of this PC
Note: Each **MPEGIOPro_NetworkInit()** must be matched by a **MPEGIOPro_NetworkEnd()**

-- HANDLE **MPEGIOPro_Multicast_SocketCreate**(char *IP, int port);
Function: create a multicast streaming socket
Parameters:
pIP: IP address of multicast in string format (e.g. “239.255.0.0”)
port: port number for the IP address
Return: handle for the socket(non-zero), NULL for failure

-- void **MPEGIOPro_Multicast_Send**(HANDLE hSocket, unsigned char * pData, int len);
Function: Send data to multicast socket
Parameters:
hSocket: valid socket as returned by **MPEGIOPro_Multicast_SocketCreate**
pData: data to send
len: length of data in bytes

-- void **MPEGIOPro_StreamVideoMulticast**(HANDLE socket, int pStr, int nSize);
Function: Send data to multicast socket using integer data address
Parameters:
hSocket: valid socket as returned by **MPEGIOPro_Multicast_SocketCreate**
pStr: address of the data memory to send, in 4-byte integer form
nSize: length of data in bytes
Note: This function can be used in programming languages where passing string address is difficult, e.g., in callback functions inside VisualBasic, to replace **MPEGIOPro_ Multicast _Send**

-- void **MPEGIOPro_Multicast_SocketClose**(HANDLE hSocket);
Function: close a multicast streaming socket
Parameters:
hSocket: valid socket as returned by **MPEGIOPro_Multicast_SocketCreate**

-- HANDLE **MPEGIOPro_Udp_SocketCreate**(int port);
Function: create a unicast streaming socket
Parameters:
port: port number for the IP address
Return: handle for the socket(non-zero), NULL for failure

-- void **MPEGIOPro_Udp_Send**(HANDLE hSocket, const char * pIP,
unsigned char * pData, int len);
Function: Send data to unicast socket
Parameters:
hSocket: valid socket as returned by **MPEGIOPro_Udp_SocketCreate**
pIP: IP address of unicast in string format (e.g. "169.212.123.2")
pData: data to send
len: length of data in bytes

-- void **MPEGIOPro_StreamVideoUnicast**(HANDLE socket, int IP, int pStr, int nSize);
Function: Send data to unicast socket using integer data area address
Parameters:
hSocket: valid socket as returned by **MPEGIOPro_Udp_SocketCreate**
IP: IP address in 4-byte integer form, e.g. "192.168.1.255" is in 0xC0A801FF form
pStr: address of the data memory to send in 4-byte integer form
nSize: length of data in bytes
Note: This function can be used in programming languages where passing string address is difficult, e.g., in callback functions inside VisualBasic, to replace **MPEGIOPro_Udp_Send**

-- void **MPEGIOPro_Udp_SocketClose**(HANDLE hSocket);
Function: close a unicast streaming socket
Parameters:
hSocket: valid socket as returned by **MPEGIOPro_Udp_SocketCreate**

6. Direct Chipset Programming

The following functions program the three main chipsets on board directly, consult the chips' user manuals and **Inventa Australia** before proceeding if you are not sure of the chips' registers and working principles.

ULONG **RlsSendMuxCommand**(ULONG nDev, CMD_BLOCK *cmd_block, UINT reg_size);
Function: Send command to MPEG Encoder Chip VW2005
Parameters:
nDev: the channel number, must be < the total channels found
cmd_block: command parameter structure defined as:

```
typedef struct _CMD_BLOCK
{
    unsigned char       ACK; // Must set to 1
    ULONG               Handle; // Set to 0
    ULONG               RCode; //returning: 0=good; otherwise error code
    ULONG               ParamsLen;    // number of parameters for this cmd
    ULONG               VWCmd;       //VW2005 command code
    ULONG               Params[256];  // VW2005 cmd parameters
} CMD_BLOCK, *PCMD_BLOCK;
```

reg_size: Must set to sizeof(CMD_BLOCK)
Return: 0 for success

Note:

1. Before calling this function, zero the **cmd_block** structure, then filling the values
2. calling **MPEGIOPro_DefaultEncodeParam**(nDev) will reset many values to the encoder, so this function should be called after **MPEGIOPro_DefaultEncodeParam**

ULONG **RLSI2CSendDataArray**(ULONG nDev, UCHAR cAddr, ULONG nLen, UCHAR *pVal);

Function: Send command to OSD Chip MTV118

Parameters:

nDev: the channel number, must be < the total channels found

cAddr: Must be set to 0x7A

nLen: Length of command pointed to by **pVal**, in bytes

pVal: Point to the actual commands

Return: 0 for success

ULONG **RlsGet713xReg**(ULONG nDev, ULONG nReg, UCHAR *pRegVal);

Function: Get Register Value from Video Decoder Chip SAA7130

Parameters:

nDev: the channel number, must be < the total channels found

nReg: Register Address

pRegVal: Pointer of Variable to Hold the Returned Register Value (One Byte Long)

Return: 0 for success

ULONG **RlsSet713xReg**(ULONG nDev, ULONG nReg, UCHAR ucRegVal);

Function: Set Register Value for Video Decoder Chip SAA7130

Parameters:

nDev: the channel number, must be < the total channels found

nReg: Register Address

ucRegVal: Register Value (One Byte Long) to Set

Return: 0 for success

7. SDK Function Calling Sequences

(1). MPEGIOPro_Init()
Must be called first before calling any other functions except **MPEGIOPro_SDKVer**

(2). All Other Functions except MPEGIOPro_SDKVer
Can be called after **MPEGIOPro_Init**()

(3). Various encoding parameters setup functions, such as **MPEGIOPro_SetEncodeVideoBitRate, MPEGIOPro_SetAspectRatio**, etc, must be called before calling **MPEGIO_StartEncode,** to make the encoded video bearing the intended encoding parameters. Once **MPEGIO_StartEncode** has been called, setting up encoding parameters will not be possible until **MPEGIO_StopEncode** is called.

(4). OSD and Colour setup(**MPEGIOPro_SetVideoParam**) functions can be called during the encoding time (after **MPEGIO_StartEncode** is called and before **MPEGIO_StopEncode** is called**)**

(5). MPEGIOPro_Uninit()
Must be called before exiting Application Software

(6). MPEGIOPro_SDKVer
MPEGIOPro_SetMirror
Can be called any time anywhere

(7). Streaming functions must be called between **MPEGIOPro_NetworkInit()** and **MPEGIOPro_NetworkEnd()**

8. SDK Operation Requirement

8.1 To use **MPEGIOPro SDK** functions to write application software, or to execute application software written by using **MPEGIOPro.DLL, MPEGIOPro.DLL-related files**(see 8.2 section), are needed.

To run the linked C++ sample program **MPEGIOPro.exe** on the **MPEGIOPro** Setup CD, **MPEGIOPro** device driver must be installed first (see Section 5. Software Installation in the "MPEGIOPro User Manual"). To run the compiled VisualBasic sample program **MPEGIOProVB.exe** on the **MPEGIOPro** Setup CD, apart from the device drivers, Microsoft .Net Framework 2.0 and above must also be installed.

8.2 To copy **MPEGIOPro.DLL-related files** to a **development PC** that you will use to write and execute application software written by using **MPEGIOPro.dll**:

1. copy all **".lib"** and **".dll"** files from the SDK**lib** folder of the **MPEGIOPro** Setup CD to a folder on the **development PC** that is in the library search path, e.g. the C:\Windows\System32
2. Copy all **".sre"** files from the SDK**lib** folder to the folder on the **development PC** where the compiled and linked .exe file of your application software will stay, e.g. the Debug and Release folders under your Microsoft VisualStudio's Project folder of your application software to be developed
3. Copy all **".h"** files from the SDK**inc** folder of the **MPEGIOPro** Setup CD to a folder on the **development PC** that is in the header file search path, e.g. your project's include folder

8.3 To copy **MPEGIOPro.DLL-related files** to a **target PC** that will run application software written by using **MPEGIOPro.dll**:

1. Copy all **".dll"** files from the SDK**lib** folder of the **MPEGIOPro** Setup CD to a folder on the **target PC** that is in the library search path, e.g. the C:\Windows\System32 or your application software's installation folder
2. Copy all **".sre"** files from the SDK**lib** folder to the folder on the **target PC** where your application software will be installed

8.4 Include Files:
MPEGIOPro.h must be included within C++ source code calling **MPEGIOPro.DLL** functions.

8.5 Firmware Files:
The boot.sre and pscodec.src files are MPEG encoding chipset firmware files, they must reside in the same folder as any **MPEGIOPro** executable programs, inc. the sample **MPEGIOPro.exe** and **MPEGIOProVB.exe** programs mentioned above.

8.6 Windows XP, Windows Vista and Windows 7 share the same MPEGIOPro.DLL Library

9. Sample Source Code

Sample applications with full C++ and VisualBasic source codes and Microsoft VisualStudio Pro 2008 project files are included on the **MPEGIOPro** Setup CD under the SDK**src** folder.

Note:

1. in C++ the XP and above XP source codes are identical except the preprocessor have different values: In XP WINVER=0x0501 is defined, in above XP WINVER=0x0600 and ABOVE_WINXP are defined.
2. both the C++ and VisualBasic .Net projects are for MS VisualStudio Pro 2008 environment