Windows Player SDK
Programmer Manual

(For Windows 7/XP/2000/2003/Vista 32 bit)

Version 6.2.XX

2011-10

Index

Chapter 1 Product Description

The Player SDK (hereby referred to as “The SDK” or “The player SDK”) is the secondary developing kit for our DVR, DVS and IP devices, etc. The SDK supports video/ audio decoding from all the devices listed below:

DS-90xx series and DS-76xx series hybrid DVR;

DS-91xx series, DS-81xx/71xx/72xx series, DS-80xx/70xx series, DS-73xx series, ATM DVRs and mobile DVRs;

DS-95xx/96xx series and DS-76xx series NVR;

DS-60xx series, DS-61xx series, DS-63xx series, DS-64xx series, DS-65xx series and DS-66xx series DVS, Decoder and Encoder;

DS-40xx/41xx/42xx series compression card;

IP devices: IP module, IP camera and IP Speed Dome, etc

Main functions of the Player SDK:

Real time stream live-view, recording file playback with player control functions such as pause, step forward, step backward, and the SDK can also get stream information such as file index, decoding frame info, resolution, frame rate. The SDK also supports BMP and JPG snapshot.

Chapter 2 Version Update

Version Description

The naming rules of Player SDK version is described as following.

V *Main version*. *Sub Version*. *Fix Version*. *Reserved Version*

- ***Main version update****:* large-scale modification, re-construction or optimization
- ***Sub version update:*** Additional functions/features added
- ***Fix version update:*** partial changes or bug-fixing
- ***Reserved version:*** reserved

Special Notice: If your CPU supports Hyper-Threading Technology, please use V3.4 or above Player SDK version.

Version 6.2.0.2

Update:

1. Fix display bugs

Addition:

2. Support video stream from DS-65xx and DS-66xx series DVS
3. Support video stream from DS-91xx-ST series DVR

Version 6.1.1.21

Update:

4. Fix the application exception problem when closing service of certain types of display adapter under Win7 OS;
5. Fix the error of single-frame functions invalid under playback pause mode;
6. Fix the playback speed abnormal error under throw B frame mode

Addition:

7. Support multiple watermark encryption types;
8. Support new version watermark

Version 6.1.1.17

Special Notice for Version 6.1.1.17

This new version SDK is thread-safe, and users do not need to consider about multi-thread safety issues anymore during developing. However, some of the API calling method in the previous version may cause application hanging problems, i.e. call API directly in callback function.

Update:

9. Fix the snapshot function under throw B frame mode;
10. Support longer file path;
11. Fix the playback frame reference error under step forward/backward mode;
12. Decrease the time delay of live view;

13. Fix the time stamp of audio decoding playback

Addition:

1. Support pure audio stream/ audio file playback
2. Support new version watermark;

Version 6.1.1.12

Update:

1. Extend the application of API PlayM4_AdjustWaveAudio
2. Fix the return value for PlayM4_GetSrcRemainData
3. Fix the bug of no display after some screen-saver activated
4. Support up to 32 display device for multi-display requirements
5. Fix the memory leak problem of 2CIF Jpeg

Addition:

1. Add API PlayM4_SkipErrorData and PlayM4_CheckDiscontinuousFrameNum
2. Add protect mechanism to prevent message/callback registration at the same time

Version 6.1.1.8

Update:

1. Correct the GetNextKeyFramePos failure problem;
2. Optimize the display effect of the video from 8100 series DVR and 4200 card;
3. Correct the video decoding callback data error after calling relative APIs to set the display position;
4. Does not support pure audio stream playback.

Version 6.1.1.7

Update:

1. Correct the getting total video file duration error for DS-9000 series DVR;

Addition:

1. Support AMR audio;
2. Support PCM-L16 audio.

Version 6.1.1.6

Update:

1. Optimize the 4CIF decoding efficiency;
2. Modified the audio/video decoding callback, and make it consistent with V4.9

Addition:

1. Support video/audio from DS-65xx series;

Version 6.1.1.5

Update:

1. Correct the return value error of GetSourceBufferRemain;
2. Fix the step forward failure.

Version 6.1.1.3

Update:

1. Correct the no display problem after locking screen;

Addition:

1. Support MJPEG video playback;
2. Support G726 audio playback.

V 6.1.1.0

Addition:

1. Self-Adaptive to multi-screen display
2. Support decoding of KY2009 video format.
3. Functions of PlayM4_GetFileSpecialAttr (), PlayM4_GetOriginalFrameCallBack () are no longer supported, these functions can be realized in other separated libraries.

V6.0.0.1

Addition:

1. Support DS-9000 Series device
2. Support up to 500 channel decoding instead of the previous 100 channel decoding

V5.0

Addition:

1. All the decoding modules are separated independently in dll format, and load dynamically during the decoding process if needed.

V4.9

Addition:

1. Add 2 APIs PlayM4_GetPort() and PlayM4_FreePort(), which are used to get the free player port number and release the port number already occupied.
2. Add callback function PlayM4_SetFileEndCallback () to get file decode end information.

V4.8 (Build 2007-08-13)

Addition:

1. Provide API PlayM4_GetBMP () and PlayM4_GetJPEG () to fix the bug of can not capture images on pause mode, and these 2 APIs can be called whenever during playback.
2. Provide API on frame loss occasion to judge whether jump to the next I frame when frame number is not continuous, and the default mode is to jump to the next frame.
3. Provide API PlayM4_SetDecCallBackMend (), which enables user-define parameters.

V4.7 (Build 2006-07-11)

Addition:

1. Provide API PlayM4_SetJpegQuality () to set the quality of the jpeg file.

2. Provide API PlayM4_ConvertToJpegFile () to convert the yuv image to a jpeg file.
3. Provide API PlayM4_SetDeflash () to set deflash of key frame.
4. Provide API PlayM4_InputFileHead to input file header before the file is opened;

V4.6 (Build 2005-08-08)
Update:
1. Optimize the decode of player SDK, decrease the CPU usage it takes.

V4.5 (Build 2005-03-03)
Addition:
1. Provide API PlayM4_GetOriginalFrameCallBack () and PlayM4_GetFileSpecialAttr() to combine files.
Update:
1. Update the decoding lib to match version3.2 HCI board encoding. Be compatible with old version.
Notice: Old version player SDK can not playback the encoded data of HCI board using version3.2 above system SDK.

V4.3 (Build 2004-09-10)
Addition:
1. Add an info function to inform the user if the image format is changed when in encoding.
Update:
1. The original callback function to inform the user if the image format is changed when in encoding is sheltering.
2. The defaulted adjusting video parameter interface will not be opened, efficiently save the resources of CPU.

V4.3 (Build 2004-09-01)
Addition:
1. Add a callback function to inform the user if the image format is changed during encoding, and users can change the size of the image interface.
2. Add APIs to get/adjust video parameters, and therefore users can change video parameters like the brightness, contrast, saturation, and hue during playback, and get a better playback performance.
3. Add a function that is to transform the decoded YUV data directly to AVI format files in the DEMO of the player. Please pay attention: at present we can only process the video data, and the transformed AVI files will occupy large rooms of the disk, one second data will need 3.6M. The files needed to be transformed cannot exceed 500 seconds, and it need to install version DivX5.2 Activex to playback the transformed files.

V4.3 (Build 2004-06-26)
Update:
1. Modify the problem possibly happened when using the network client-end to show the picture

2. Modify the BUG possibly happened when using the length to index the files in the last version.

V4.2 (Build 0616)

Update:

1. Support the dynamic change of the image format, for example: from 4CIF to CIF, the player will self recognized, need not to reboot the player.
2. Modify PlayM4_SetVerifyCallBack function, which can check whether the frame or data is lost in the file.

V4.0 (Build 0420)

Update:

1. Support 4CIF picture format decode.

V3.6 (Build 1230)

Update:

1. Support picture format changed on stream. Example, from CIF to QCIF, needn't restart.

Addition:

1. Adjust wave data.
2. Verify the data.
3. Refer to the wave data through a callback function.

V3.4 (Build 0626)

Update:

1. Support more than 16 ports, the port is now from 0 to 99;

Addition:

1. Set/Get the timer that the player sdk using.
2. Clear the buffer/Get the buffer remain value.

V3.2 (Build 0430)

Addition:

1. Split audio and video stream to input
2. When using off-screen, you can get the surface DC. And then you cound draw on the surface. If you use overlay surface, you needn't this DC, becase you can draw on the window directly.
3. Get and set file index information.

V3.0 (Build 0328)

Addition:

1. Support date stream generated from DS-400XH series card
2. Can set up to 4 display regions and support part-region display (can realize part-region enlargement) (68-70)
3. Can set stream type of decode CALLBACK (67)

Modification:

1. Correct BUG that is originally called I frame back only (please refer to PlayM4_SetDecCallBack).

V2.5 (Build 1118)

Modification:

1. Fix the bug of PlayM4_OpenStream occasionally fails with error value PLAYM4_SYS_NOT_SUPPORT.

V2.5 (Build 1115)

Addition:

1. Set display type

Modification:

1. Modify some bugs

V2.4 (Build 0911)

Addition: (42~63)

1. Some functions to get more information;
2. Some functions to control source buffer in stream mode;
3. Some functions to control render buffer ;
4. Some functions to locate the file. And play step back ;
5. A function to throw B frames ;
6. Some functions to support Multiple-Monitor Systems ;

Modification:

1. The efficiency of the player is improved greatly, and at the same time it can play 9 files (Pentinum4 1.5GHZ), and it can play more if the video is simple. Notice: The display hardware must supports arbitrary shrinking and stretching of a surface along the x-axis (horizontally) and the y-axis (vertically) for blit operations.
2. Don’t exit the decode thread when the file which is playing come to the end.
3. The current time and frame number witch is got through the functions don’t reference to the state of decoding, but reference to the state of playing.
4. Extend the range of playing speed.
5. Correct the BMP snapshot error when the video adapter doesn’t support shrinking and stretching

V2.2 (Build 0703)

Addition :

1. Getting error codes	PlayM4_GetLastError ;
2. Refreshing display windows	PlayM4_RefreshPlay ;
3. Getting the size of image	PlayM4_GetPictureSize ;
4. Using OVERLAY surface to display	PlayM4_SetOverlayMode ;
5. Setting image quality	PlayM4_SetPicQuality ;
6. Opening sound in share mode	PlayM4_PlaySoundShare ;
7. Closing sound in share mode	PlayM4_StopSoundShare ;

Modification:

1. Can play step when it is pausing.
2. Support the operations such as pause, fast, , slow and step in STREAM_FILE mode.
3. Support shrinking and stretching with software method when the display hardware doesn't support it.
4. Correct the playback speed when the video stream is not normal (25 frames per second in PAL).

V2.0 (Build 0607)

Modification:

1. Improved the image quality.
2. Improve the playing performance.

(Build 0605)

Addition:

1. Capture picture.
2. Support the user to display audio and video themselves
3. Can set stream mode (the real time mode and the file mode).
4. Get the current time of playing (millisecond).
5. Locate file by time.
6. Getting the total frames of the file and the number of frame which is currently decoding.
7. Getting the current display rate.
8. Get the current version information.
9. Support QCIF format.

V1.11:

1. Correct the problem that the player causes the computer freeze.

V1.1:

Addition:

1. Support multi channel playback (The performance is related to the PC hardware configuration. Now it can playback 4 channels at the same time in Pentinum4 1.5GHz).
2. Support stream input mode;

Notice: CPU must be Intel Pentinum3 or above; The users do not need to initialize and release DiretDraw surface;

V1.0

The player only support one channel playback, and the input parameter of nPort must be 0, but we will support multi-channel Playback in future. Because the video display requires some capabilities of the video adapter, therefore it is suggested to follow these operation steps if there is problems during display: 1) Set the color quality as 32bit; 2) Replace video adapter.

Chapter 3 Error Code Definition

ID	Code	Description
PLAYM4_NOERROR	0	No error
PLAYM4_PARA_OVER	1	Illegal input parameter
PLAYM4_ORDER_ERROR	2	Calling reference error
PLAYM4_TIMER_ERROR	3	Set timer failure
PLAYM4_DEC_VIDEO_ERROR	4	Video decoding failure
PLAYM4_DEC_AUDIO_ERROR	5	Audio decoding failure
PLAYM4_ALLOC_MEMORY_ERROR	6	Memory allocation failure
PLAYM4_OPEN_FILE_ERROR	7	File operation failure
PLAYM4_CREATE_OBJ_ERROR	8	Create thread failure
PLAYM4_CREATE_DDRAW_ERROR	9	Create directDraw failure
PLAYM4_CREATE_OFFSCREEN_ERROR	10	Create offscreen failure
PLAYM4_BUF_OVER	11	Buffer overflow, input stream failure
PLAYM4_CREATE_SOUND_ERROR	12	Create sound device failure
PLAYM4_SET_VOLUME_ERROR	13	Set volume failure
PLAYM4_SUPPORT_FILE_ONLY	14	This API can only be called in file decoding mode
PLAYM4_SUPPORT_STREAM_ONLY	15	This API can only be called in stream decoding mode
PLAYM4_SYS_NOT_SUPPORT	16	System not support, the SDK can only work with CPU above Pentium 3
PLAYM4_FILEHEADER_UNKNOWN	17	Missing file header
PLAYM4_VERSION_INCORRECT	18	Version mismatch between encoder and decoder
PALYM4_INIT_DECODER_ERROR	19	Initialize decoder failure
PLAYM4_CHECK_FILE_ERROR	20	File too short or unrecognizable stream
PLAYM4_INIT_TIMER_ERROR	21	Initialize timer failure
PLAYM4_BLT_ERROR	22	BLT failure
PLAYM4_UPDATE_ERROR	23	Update overlay surface failure
PLAYM4_OPEN_FILE_ERROR_MULTI	24	Open video & audio stream failure
PLAYM4_OPEN_FILE_ERROR_VIDEO	25	Open video stream failure
PLAYM4_JPEG_COMPRESS_ERROR	26	JPEG compression failure
PLAYM4_EXTRACT_NOT_SUPPORT	27	File type not supported
PLAYM4_EXTRACT_DATA_ERROR	28	Data error
PLAYM4_SECRET_KEY_ERROR	29	Secret key error
PLAYM4_DECODE_KEYFRAME_ERROR	30	Key frame decoding failure

Chapter 4 Description about Display

The display part of the player mainly adopts DirectDraw technology. There are 2 ways to achieve video display: 1. Create off screen image with BLT. 2. Create OVERLAY surface. The characters of these two methods are: 1) For the Off screen display mode, the merits are: each channel of the Multi-channel playback can be relatively independent and not influenced by each other. Shortcomings: the display performance is greatly affected by the display adapter. If the display adapter does not support zoom operation, the player will zoom via software if needed (while the display window and the original image size are different), which will cause high CPU usage. Therefore we provide an interface PlayM4_ GetCaps to test the display adapter's capability. Users can test with their display adapters to see if it supports BLT zooming, etc. Form 1 is a display adapter that we test by ourselves; 2) For the OVERLAY display mode, the merits are: Most display adapters support OVERLAY mode, and the OVERLAY mode supports hardware zooming. If the display adapter does not support off screen with zooming, then we can use overlay mode. Shortcoming: There can be only one OVERLAY surface for each display adapter, and thus only 1 channel display can use OVERLAY mode. If there is other program that occupies the OVERLAY surface, the player cannot use OVERLAY then; and if the OVERLAY surface is occupied by the player, the other programs cannot display in overlay mode, either.

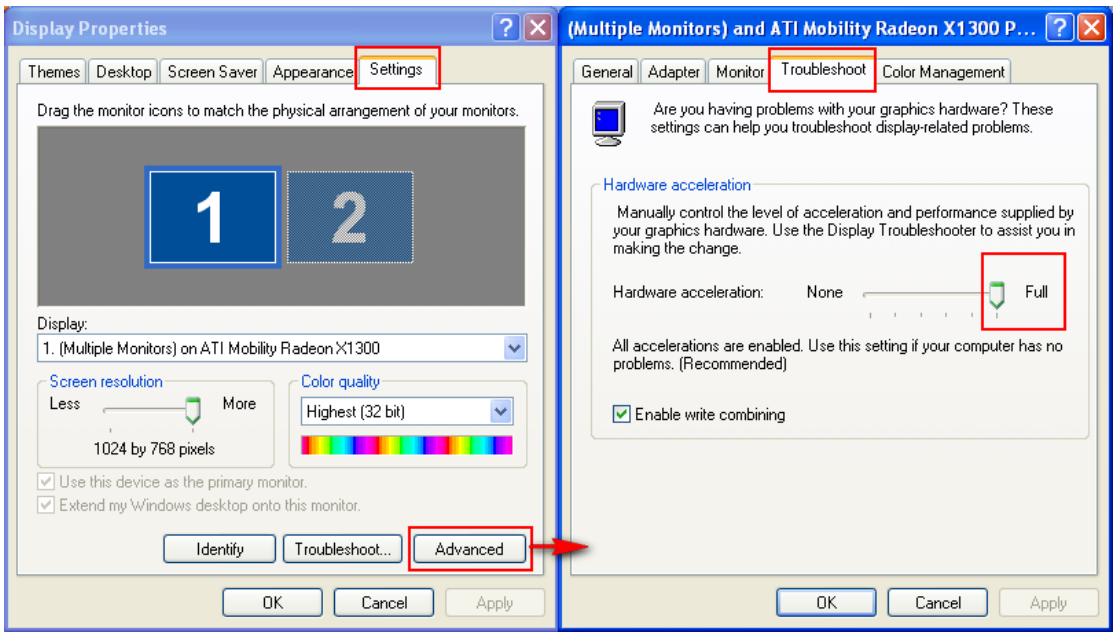
For the Windows 2003 OS users, please kindly notice:

The Server Operating System such as Windows 2003 is not developed for multimedia display functions, and thus some of the video handling functions (i.e. hardware acceleration, directX) are disabled by default and need to be enabled manually.

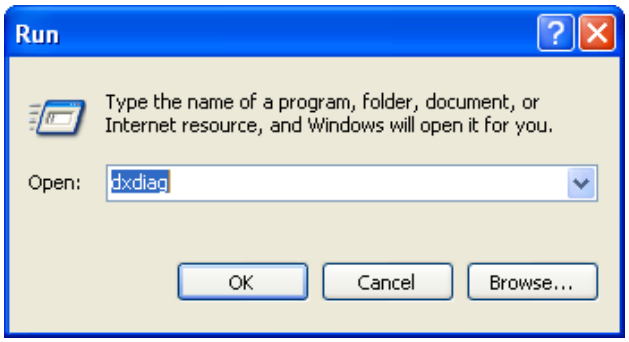
As most of the display functions in the SDK is depended on the BLT function of the display adapter (image zooming via hardware, when the original image size is not consistent with the display image size, we'll need this feature on the display adapter), if the display adapter (or its driver) does not support this feature, the SDK will switch to software zooming mode, and then the CPU usage will increase accordingly.

For Windows 2003 OS, please kindly follow the below operation steps:

1. Right click on Windows Desktop-> Select "Properties" ->Settings-> Advanced-> Troubleshoot->and drag "Hardware Acceleration" to "Full".



2. Click “Start”-> “Run” of Windows task bar, input “dxdiag” and enter DirectX diagnosing program.



3. Enable “DirectDraw” and “Direct3D” options in the “Display” column.

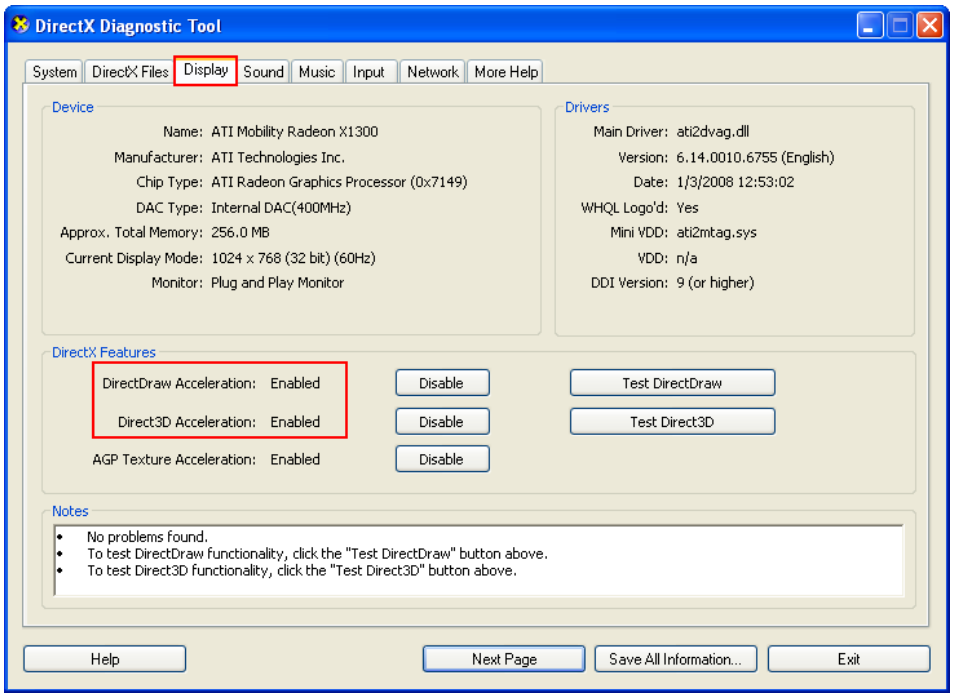


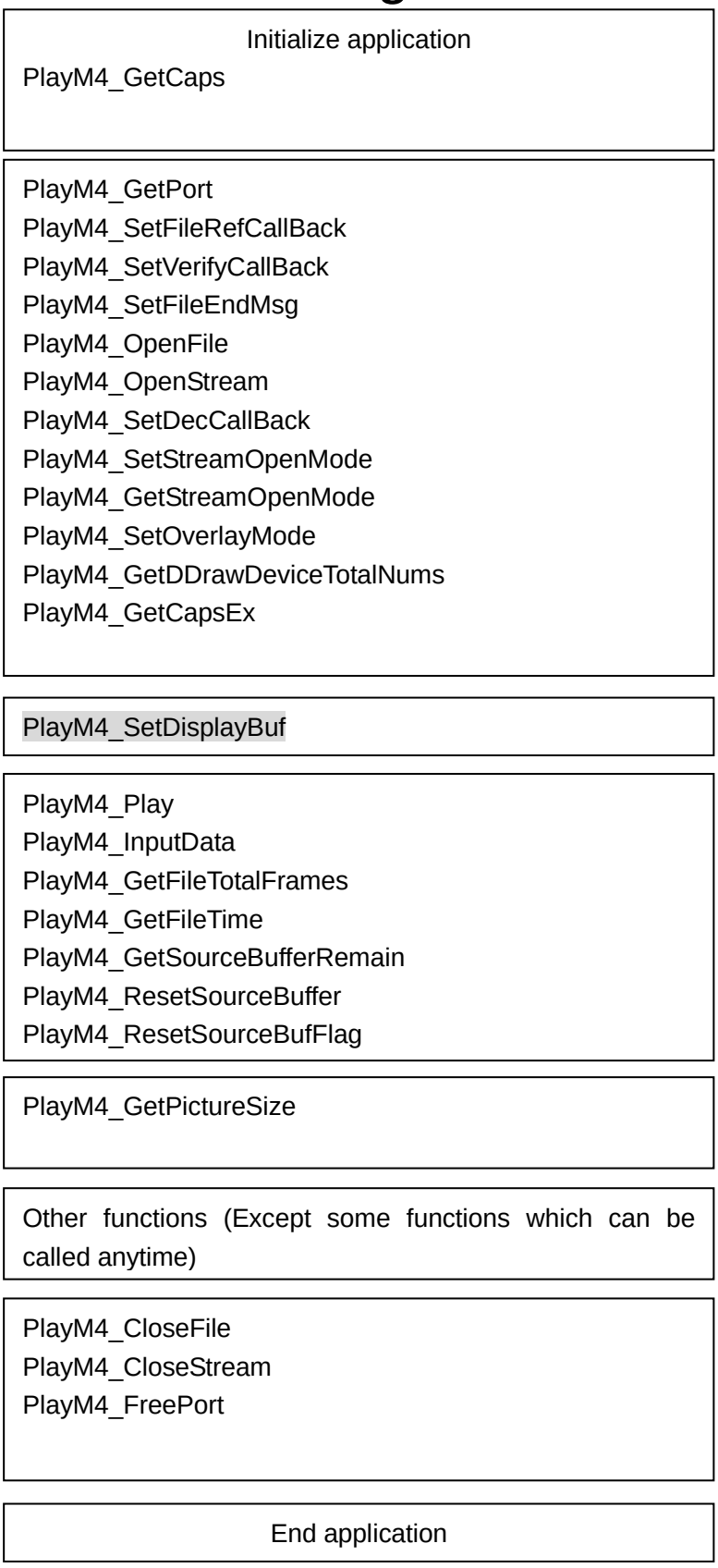
Table 1: These are some video adapter models which have been already tested (Windows2000)

Display Adapter Model	**Memory (M)**	**Support Color Convertion**	**Support shrink**	**Support Stretch**
ATI Rage128	32	YES	YES	YES
ATI Radeon LE	32	YES	YES	YES
ATI Radeon 7200	64	YES	YES	YES
nVidia TNT2 Model64	16 & 32	YES	YES	YES
nVidia TNT2 Pro	32	YES	YES	YES
Geforce2 Mx, Mx200, Mx400	32	YES	YES	YES
Geforce4 Mx420, Mx440	32	YES	YES	YES
**********				
Sis630	16	NO	NO	NO
Sis305	32	YES	NO	NO

Notice:

For the nVidia display adapters, users may need to update the driver to the latest version, otherwise some function may not be supported. If you find other display adapters cannot support some display functions, it is suggested to update the driver and test again.

Chapter 5 API Calling Reference



Chapter 6 API Description

System Operation and Error Code Query

6.1. Get SDK Version and Build Number PlayM4_GetSdkVersion

DWORD PlayM4_GetSdkVersion();

Description:

Get SDK version and build number.

Parameters:

--

Return:

The higher 16 digits is the current build number, digits 9~16 is the main version number and digit 1~8 is the sub version number.

i.e. return 0x06040105 stands version1.5, Build 0604.

Notice: For the debug version SDKs, there will only be difference in build number.

6.2. Get Error Code PlayM4_GetLastError

DWORD PlayM4_GetLastError (LONG nPort) ;

Description:

Get the error code.

Return:

The value specified the error code. For the error description, please refer to the table in Chapter 3.

6.3. Get the system display capabilities PlayM4_GetCaps

int PlayM4_GetCaps ();

Description:

Get the system capabilities information for the player.

Return:

Bit1-8 respectively means the following info (True means support)

SUPPORT_DDRAW

Support DIRECTDRAW. If not, player can't work.

SUPPORT_BLT

Support BLT operation. If not, player can't work.

SUPPORT_BLTFOURCC

Display hardware is capable of color-space conversions during blit operations.

SUPPORT_BLTSHRINKX

Supports arbitrary shrinking of a surface along the x-axis (horizontally). This flag is valid only for blit operations.

SUPPORT_BLTSHRINKY

Supports arbitrary shrinking of a surface along the y-axis (vertically). This flag is valid only for blit operations.

SUPPORT_BLTSTRETCHX

Supports arbitrary stretching of a surface along the x-axis (horizontally). This flag is valid only for blit operations.

SUPPORT_BLTSTRETCHY

Supports arbitrary stretching of a surface along the y-axis (vertically). This flag is valid only for blit operations.

SUPPORT_SSE

Supports SSE instruction set. If supports, It will get high performance.

SUPPORT_MMX

Supports MMX instruction set.

Notice:

If all the above functions are supported by the display adapter, the CPU usage will be lowered. i.e. If the decoded video is 352*288 (PAL) , and the display adapter does not support BLT, then it is suggested to set the display window size also as 352*288.

6.4. Initial DirectDraw Surface PlayM4_InitDDraw

BOOL PlayM4_InitDDraw (HWND hWnd);

Description:

Initialize DirectDraw surface. Please kindly notice that for VB & DELPHI developing, please disable the WS_CLIPCHILDREN window style of the dialog box, otherwise the display image will be covered by the active controls on the dialog box.

Parameters:

hWnd

[in] The window handle of the application mainframe.

Return:

If the function succeeds, the return value will be nonzero.

If the function fails, the return value will be 0. To get extended error information, please call API PlayM4_GetLastError ().

Notice:

For the SDK version above V1.1, users do not need to call this function.

6.5. Release DirectDraw Surface PlayM4_RealeseDDraw

BOOL PlayM4_ReleaseDDraw ();

Description:

Release DirectDraw surface.

Return:

If the function succeeds, the return value is TRUE

If the function fails, the return value is FALSE.

Notice:

For the SDK version above V1.1, users do not need to call this function.

6.6. Set Timer Type PlayM4_SetTimerType

BOOL _stdcall PlayM4_SetTimerType (LONG nPort, DWORD nTimerType, DWORD nReserved);

Description:

Set the player SDK timer.

Parameters:

nTimerType

[in] TIMER_1 or TIMER_2, please refer to the MACRO definition below.

nReserved

[in] reserved.

Macro Definition:

TIMER_1: Multi-media timer, support up to 16 for each process. **It is suggested to use this TIMER_1 for file playback.**

TIMER_2: No numeric limitation, but this timer is with lower precision, and is not recommended for high speed playback. **It is suggested to use this TIMER_2 for live preview.**

Channel 0-15 will use TIMER_1 while others will use TIMER_2 by default.

Return:

If the function succeeds, the return value is TRUE

If the function fails, the return value is FALSE.

This API need to be called before open file or open stream.

Notice: must be called before open operation.

6.7. Get Timer Type PlayM4_GetTimerType

```
BOOL _stdcall PlayM4_GetTimerType (LONG nPort, DWORD *pTimerType, DWORD *pReserved);
```

Description:

Get the player SDK timer

Parameters:

pTimerType

[out] TIMER_1orTIMER_2;

pReserved

[out] reserved.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.8. Get Valid Port Number PlayM4_GetPort

```
BOOL _stdcall PlayM4_GetPort (LONG* nPort);
```

Description:

Get the unused port number.

Parameters:

nPort

[in] Long pointer to get the port number

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.9. Release Player Port PlayM4_FreePort

BOOL _stdcall PlayM4_FreePort (LONG nPort);

Description:

Release the port number which has been occupied

Parameters:

nPort

[in]Port number of the player

It is suggested to set the nPort as -1 after port releasing.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

File Operation

6.10. Open File PlayM4_OpenFile

BOOL PlayM4_OpenFile (LONG nPort, LPSTR sFileName);

Description:

Open the file for playback.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

sFileName

[in] The file name. The file size is from 4k to 4G bytes.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.11. Close File PlayM4_CloseFile

BOOL PlayM4_CloseFile (LONG nPort);

Description:

Close the file that has been opened.

Parameters:

nPort

[in] The port number of the player, it ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Stream Operation

6.12. Set Stream Input Mode PlayM4_SetStreamOpenMode

BOOL PlayM4_SetStreamOpenMode (LONG nPort, DWORD nMode);

Description:

Set stream input mode.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nMode

[in] The stream mode: STREAME_REALTIME mode (default) or STREAME_FILE mode. STREAME_REALTIME mode will ensure real-time performance in priority and prevent data blocking problem with strict data checking mechanism while STREAME_FILE doesn't.

Notice: If the return value of PlayM4_InputData () is FALSE, users need to wait for a moment and then input again.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is zero.

Notice:

This function needs to be called before playback starts. SDK version above 2.2 supports pause, fast/slow forward and step forward operation.

6.13. Get Stream Input Mode PlayM4_GetStreamOpenMode

LONG PlayM4_GetStreamOpenMode (LONG nPort);

Description:

Get stream input mode.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

STREAME_REALTIME or STREAME_FILE

6.14. Open Stream PlayM4_OpenStream

BOOL PlayM4_OpenStream (LONG nPort, PBYTE pFileHeadBuf, DWORD nSize, DWORD nBufPoolSize);

Description:

Open the stream for playback (similar with open file).

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pFileHeadBuf

[in] The file header which is got from the recording callback APIs of network client SDK or card SDK

nSize

[in] The data length of the file header.

nBufPoolSize

[in] Specify the size of the source buffer. It ranges from SOURCE_BUF_MIN to SOURCE_BUF_MAX. Decoding failure may generate if this nBufPoolSize is too small. It is suggested no less than 200*1024 for the devices whose encoding resolution is under 4CIF, and no less than 600*1024 for the HD devices.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is zero.

6.15. Close Stream PlayM4_CloseStream

BOOL PlayM4_CloseStream (LONG nPort);

Description:

Close the stream which has been opened.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is zero.

6.16. Input Stream Data PlayM4_InputData

BOOL PlayM4_InputData (LONG nPort, PBYTE pBuf, DWORD nSize);

Description:

Input stream data.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pBuf

[in] buffer address

nSize

[in] buffer size

Return:

If the data input succeeds, the return value is nonzero.

If the data input fails, the return value is zero.

6.17. Open Video/Audio Stream Separately PlayM4_OpenStreamEx

BOOL _stdcall PlayM4_OpenStreamEx (LONG nPort, PBYTE pFileHeadBuf, DWORD nSize, DWORD nBufPoolSize);

Description:

Open stream. And the video and audio stream input separately.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pFileHeadBuf

[in]The file header which is got from card.

nSize

[in] The size of the file header.

nBufPoolSize

[in]Specifies the size of the source buffer. The range ranges from SOURCE_BUF_MIN to SOURCE_BUF_MAX ;

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is zero.

6.18. Close Stream (Video/Audio Separately) PlayM4_CloseStreamEx

```
BOOL _stdcall PlayM4_CloseStreamEx (LONG nPort);
```

Description:

Close the stream.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is zero.

6.19. Input Video Data PlayM4_InputVideoData

```
BOOL _stdcall PlayM4_InputVideoData (LONG nPort, PBYTE pBuf, DWORD nSize);
```

Description:

Input video stream data got from card

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1. **pBuf** [in] Pointer to the buffer containing the data to be written to the source buffer.

nSize

[in] Number of bytes to write to the source buffer.

Return:

If the video input succeeds, the return value is nonzero.

If the video input fails, the return value is zero.

6.20. Input Audio Data PlayM4_InputAudioData

```
BOOL _stdcall PlayM4_InputAudioData (LONG nPort, PBYTE pBuf, DWORD nSize);
```

Description:

Input audio stream data got from card

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pBuf

[in] Pointer to the buffer containing the data to be written to the source buffer.

nSize

[in] Number of bytes to write to the source buffer.

Return:

If the audio input succeeds, the return value is nonzero.

If the audio input fails, the return value is zero.

Playback Control

6.21. Start Playback PlayM4_Play

BOOL PlayM4_Play (LONG nPort, HWND hWnd);

Description:

Start playback. The display image size will be automatically adjusted according to the hWnd window size. If the user wants full screen display, please magnify the hWnd window to full screen size. If the has already been in playback status, this API will adjust the current playing speed to normal 1X speed.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

hWnd

[in] The handle of the display window.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.22. Stop Playback PlayM4_Stop

BOOL PlayM4_Stop (LONG nPort);

Description:

Stop playback.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.23. Pause Playback PlayM4_Pause

BOOL PlayM4_Pause (LONG nPort, DWORD nPause);

Description:

Pause/Resume playback.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nPause

[in] If it is TRUE, then pause the playing file, else then resume the playing file.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.To get extended error information, call PlayM4_GetLastError.

6.24. Fast Forward PlayM4_Fast

BOOL PlayM4_Fast (LONG nPort);

Description:

Fast forward. The playback speed will be doubled after calling this function , and this function can be called up to 4 times continuously. If the users want to restore to 1X speed mode, please call PlayM4_Play () and it will playback in normal speed from the current position.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.25. Slow Forward PlayM4_Slow

BOOL PlayM4_Slow (LONG nPort);

Description:

Slow forward. The playback speed will be lowered to half of the current speed after calling this function, and this function can be called up to 4 times continuously. If the users want to restore to 1X speed mode, please call PlayM4_Play () and it will playback in normal speed from the current position.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.26. Step Forward PlayM4_OneByOne

BOOL PlayM4_OneByOne (LONG nPort);

Description: Step Forward. Call PlayM4_Play () to restore normal playback mode.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.27. Step Backward PlayM4_OneByOneBack

BOOL PlayM4_OneByOneBack (LONG nPort);

Description:

Single frame replay. Reverse 1 frame after each transfer. This function must be transferred after file index is built.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.28. Play Sound PlayM4_PlaySound

BOOL PlayM4_PlaySound (LONG nPort);

Description:

Open the audio. Only one channel audio playback will be enabled, and the other channels' audio will be disabled automatically.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

The audio display is disabled by default.

6.29. Stop Sound PlayM4_StopSound

BOOL PlayM4_StopSound ();

Description:

Stop the audio playback.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.30. Play Sound in Share Mode PlayM4_PlaySoundShare

BOOL PlayM4_PlaySoundShare (LONG nPort);

Description:

Plays a sound specified by the channel with share mode. It doesn't stop audio from other channels.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

It doesn't support play multiple audio channels in Windows 98 or earlier OS version.

6.31. Stop Sound in Share Mode PlayM4_StopSoundShare

BOOL PlayM4_StopSoundShare (LONG nPort);

Description:

Stop a sound specified by the channel. Notice : The player must use the same mode to play or stop the sound.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.32. Set Volume PlayM4_SetVolume

BOOL PlayM4_SetVolume (LONG nPort, WORD nVolume);

Description:

Set audio volume of the PC's audio adapter. It may effect other applications related with the system's display adapter.

Parameters:

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nVolume

[in] New volume requested for this sound, range 0-0XFFFF.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

Users can call this API before playback, the return value will be FALSE then, yet the volume setting will be saved and set as the initial volume when audio playback starts.

6.33. Get Volume PlayM4_GetVolume

WORD PlayM4_GetVolume (LONG nPort);

Description:

Get the volume level of the PC's audio adapter. It may effect other applications related with the system's display adapter.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

Volume.

6.34. Adjust Audio Wave PlayM4_AdjustWaveAudio

BOOL _stdcall PlayM4_AdjustWaveAudio (LONG nPort, LONG nCoefficient);

Description:

Adjust the wave data. This API can be called to adjust the volume of current channel only, while API PlayM4_SetVolume effects on the whole system.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nCoefficient

[in] The coefficient. The range from MIN_WAVE_COEF to MAX_WAVE_COEF, 0 means no adjust.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This function is not supported by V6.1.1.8. It will lower the audio quality, and is not suggested to use unless the user wants to adjust each channel's volume.

6.35. Set Picture Quality PlayM4_SetPicQuality

BOOL PlayM4_SetPicQuality (LONG nPort, BOOL bHighQuality);

Description:

Specifies the image quality. High image quality will lead to higher CPU usage, and thus for mutli-channel display, it is suggested to set low quality, and switch to high quality when a certain channel is enlarged during display.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bHighQuality

[in] 1 for high quality and 0 for low quality (default).

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.36. Get Picture Quality PlayM4_GetPictureQuality

BOOL PlayM4_GetPictureQuality (LONG nPort, BOOL *bHighQuality);

Description:

Get the quality of the image.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bHighQuality

[out] Pointer to the variable that receives the value of the quality.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.37. Set Display Video Parameters PlayM4_SetColor

BOOL _stdcall PlayM4_SetColor (LONG nPort, DWORD nRegionNum, int nBrightness, int nContrast, int nSaturation, int nHue);

Description:

Set video parameters of the pictures, can be in effect right away;

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nRegionNum:

Display region, please refer to PlayM4_SetDisplayRegion;if there is only one show region (the usual condition) it should be set as “0”.

nBrightness:

The brightness, the defaulted is 64, the range is from 0 to 128;

nContrast:

The contrast, the defaulted is 64; the range is from 0 to 128;

nSaturation:

The saturation, the defaulted is 64; the range is from 0 to 128;

nHue:

The hue, the defaulted is 64; the range is from 0 to 128;

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

If all the parameters are all default, it will not adjust the color.

6.38. Get Display Video Parameters PlayM4_GetColor

BOOL _stdcall PlayM4_GetColor (LONG nPort, DWORD nRegionNum, int *pBrightness, int *pContrast, int *pSaturation, int *pHue);

Description:

Get the corresponding color value, parameters are as above.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to

PLAYM4_MAX_SUPPORTS-1.

nRegionNum:

Display region, please refer to PlayM4_SetDisplayRegion;if there is only one show region (the usual condition) it should be set as “0”.

nBrightness:

The brightness, the defaulted is 64, the range is from 0 to 128;

nContrast:

The contrast, the defaulted is 64; the range is from 0 to 128;

nSaturation:

The saturation, the defaulted is 64; the range is from 0 to 128;

nHue:

The hue, the defaulted is 64; the range is from 0 to 128;

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.39. Set Playback Position (Percentage) PlayM4_SetPlayPos

BOOL PlayM4_SetPlayPos (LONG nPort, float fRelativePos);

Description:

Locate the relative playback position of the file (percentage). To get precise locating performance, users need to create file index before executing this operation, otherwise it can only achieve rough locating.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

fRelativePos

[in] The percent of the file relative position. It is from 0% to 100%.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.40. Get Playback Position (Percentage) PlayM4_GetPlayPos

float PlayM4_GetPlayPos (LONG nPort);

Description:

Get the relative playback position of file (percentage).

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return Value:

The percentage of the file’s relative playback position, ranges from 0% to 100%.

6.41. Set Playback Position (Millisecond) PlayM4_SetPlayedTimeEx

BOOL PlayM4_SetPlayedTimeEx (LONG nPort, DWORD nTime);

Description:

Set the new position in the file in milliseconds for playback. To get precise locating performance, users need to create file index before executing this operation, otherwise it can only achieve rough locating.

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The position in the file in milliseconds from the beginning

6.42. Get Playback Position (Millisecond) PlayM4_GetPlayedTimeEx

DWORD PlayM4_GetPlayedTimeEx (LONG nPort);

Description:

Get the current playback position of the file in milliseconds.

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The current playback position of the file in milliseconds

6.43. Set Playback Position (Frame No.) PlayM4_SetCurrentFrameNum

BOOL PlayM4_SetCurrentFrameNum (LONG nPort, DWORD nFrameNum);

Description:

Specifies the current position in the file (unit: frames) from the beginning. To get precise locating performance, users need to create file index before executing this operation, otherwise it can only achieve rough locating.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nFrameNum

[in] The current frame number.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.44. Get Playback Position (Frame No.) PlayM4_GetCurrentFrameNum

DWORD PlayM4_GetCurrentFrameNum (LONG nPort);

Description:

Get the current playback position in the file in frame number from the beginning

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The current playback position in the file in frame number from the beginning.

6.45. Image De-flashing PlayM4_SetDeflash

BOOL _stdcall PlayM4_SetDeflash (LONG nPort, BOOL bDeflash)

Description:

set deflash of key frame.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bDeflash

[in] TRUE enables the deflash of key frame , FALSE disable the deflash.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Get Playback or Decoding Information

6.46. Get the File's Time Duration PlayM4_GetFileTime

DWORD PlayM4_GetFileTime (LONG nPort);

Description:

Get total file duration in seconds.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The total file length in seconds.

6.47. Get the File's Frame Number PlayM4_GetFileTotalFrames

DWORD PlayM4_GetFileTotalFrames (LONG nPort);

Description:

Get the total frame number of the file.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The total frame number of the file.

6.48. Get Current Frame Rate PlayM4_GetCurrentFrameRate

DWORD PlayM4_GetCurrentFrameRate (LONG nPort);

Description:

Get the current frame rate.

Parameters:

nPort

[in] The port number of the player. It is from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The current frame rate

6.49. Get Current Played Time PlayM4_GetPlayedTime

DWORD PlayM4_GetPlayedTime (LONG nPort);

Description:

Get current file displayed time in seconds

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The current file position in seconds, counted from the beginning.

6.50. Get Current Played Frames PlayM4_GetPlayedFrames

DWORD PlayM4_GetPlayedFrames (LONG nPort);

Description:

Get the decoded frame number.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The decoded frame number from the beginning.

6.51. Get Original Image Size PlayM4_GetPictureSize

BOOL PlayM4_GetPictureSize(LONG nPort,LONG *pWidth,LONG *pHeight)

Description:

Get the original image size of the video.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

LONG *pWidth

[out] original image width, i.e. for CIF/PAL video, the image width is 352

LONG *pHeight

[out] original image height, i.e. for CIF/PAL video, the image height is 288

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.52. Get File Header Length PlayM4_GetFileHeadLength

DWORD PlayM4_GetFileHeadLength ();

Description:

Get the length of the file header or info header for data exchange purpose.

Parameters:

--

Return:

The size of the file header.

Sample:

```
CFile m_TestFile;
void Start ()
{
    //Get file header length
    DWORD nLength= PlayM4_GetFileHeadLength ();
    PBYTE pFileHead=new BYTE[nLength];
    //Open File
```

```
    m_TestFile.Open ("test.mp4 ", CFile: : modeRead, NULL);
    m_TestFile.Read (pFileHead, nLength);
    //Set Stream Mode
    PlayM4_SetStreamOpenMode (0, STREAME_FILE);
    //Open Stream
    if (!PlayM4_OpenStream (0, pFileHead, nLength, 1024*100))
    {
        m_strPlayFileName="";
        MessageBox ("Open File Failure");
    }
    //Playback
    m_bPlaying = PlayM4_Play ( 0, m_hWnd);
    delete []pFileHead;
}
////////////////////////////////////////////////////////////////////////////////
void InputData ()
{
    BYTE pBuf[4096];
    m_TestFile.Read (pBuf, sizeof (pBuf));
    while (!PlayM4_InputData (0, pBuf, sizeof (pBuf)))
    {
        if (!m_bPlaying)
            break;//If the playback has already been stopped, exit
        TRACE ("SLEEEP \n");
        Sleep (5);
    }
}
```

Decoding Operation & Control

6.53. Set Decoder Callback Stream TypePlayM4_SetDecCBStream

BOOL _stdcall PlayM4_SetDecCBStream (LONG nPort, DWORD nStream);

Description:

Set call-back stream type

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nStream

[in]1. video stream; 2.audio stream; 3.video & audio stream

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.54. Decoder Callback PlayM4_SetDecCallBack

BOOL PlayM4_SetDecCallBack (LONG nPort, void (CALLBACK* DecCBFun) (long nPort, char * pBuf, long nSize, FRAME_INFO * pFrameInfo, long nReserved1, long nReserved2));

Description:

Register a callback function to replace the decoder's display, and control the display by user. It should be called before **PlayM4_Play** and will be invalid automatically after **PlayM4_Stop.** Please notice that the decoded part is with no speed control function, and the decoder will decode the next part of data when user returns from the call back function. To use this function, user must have knowledge about video & audio display. Please refer to DirectX development package about the relative knowledge.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

DecCBFun

[in] Address of a function to be called, can't be NULL.

Description of the callback function :

nPort

The channel of player ;

pBuf

Pointer to the buffer that receives the data (audio or video) get from the player.

nSize

Specifies the number of bytes to be get from the player ;

pFrameInfo

Points to a FRAME_INFO structure to receive the information of the image or sound ;

nReserved1, nReserved2

Reserved ;

Description of the structure :

```
typedef struct{
  long nWidth;      //width of image in pixels. If it is audio data the width is 0.
  long nHeight;     // height of image in pixels , if it is audio data, the height is 0 ;
  long nStamp;      //time stamp in milliseconds.
  long nType;       //Received data type. It is must be T_AUDIO16, T_RGB32
                      orT_YV12.
 long nFrameRate;   //Suggest the frame rate to display.
}FRAME_INFO;
```

Macro Definition:

T_AUDIO16	Audio format (PCM). Sample rate is 16 khz, Mono, 16 bits per sample.
T_RGB32	Picture format, RGB32 fromat, 4 bytes per pexels. the format is like as bit values of a bitmap. (Reserved)
T_UYVY	Picture format , uyuv format . “U0-Y0-V0-Y1-U2-Y2-V2-Y3….”, and the first pixel is on top- left. (Reserved)
T_YV12	Picture format, yv12 format. “Y0-Y1-……”, ”V0-V1….”, ”U0-U1-…..”.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice: Currently only T_YV12 format raw video data is supported.

6.55. Decoder Callback with User Data PlayM4_SetDecCallBackMend

```
BOOL _stdcall PlayM4_SetDecCallBackMend (LONG nPort, void (CALLBACK*
    DecCBFun) (long nPort, char * pBuf, long nSize, FRAME_INFO * pFrameInfo, long
    nUser, long nReserved2),
long nUser);
```

Description:

Register a callback function to replace the displayed part. It is controlled by user. It is called back before **PlayM4_Play** and will be invalid automatically when **PlayM4_Stop**. Also it needs to be reset before recalling back **PlayM4_Play**. Please be aware that the decoded part does not control speed, and as user returns from call back function, the decoder will decode the next part of data. To use this function, user must understand video display and audio play. Please refer to directx development package about the relative knowledge.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

DecCBFun

[in] Address of a function to be called, which can’t be NULL.

Description of the callback function:

nPort:

The channel of player

pBuf:

Pointer to the buffer that receives the data (audio or video) get from the player

nSize:

Specifies the number of bytes to be get from the player

pFrameInfo:

Points to a FRAME_INFO structure to receive the information of the image or sound

nUser:

User-defined parameter

nReserved2:

Reserved

Structure description:

```
typedef struct{
  long nWidth;       //width of image in pixels. If it is audio data the width is 0.
  long nHeight;      // height of image in pixels , if it is audio data, the height is 0 ;
  long nStamp;       //time stamp in milliseconds.
  long nType;        //Received data type. It is must be T_AUDIO16, T_RGB32
                       orT_YV12. Please refer to relative macro description for
                       detail information.
 long nFrameRate;    //Suggest the frame rate to display.
}FRAME_INFO; Please refer to PlayM4_SetDecCallBack () for detail information.
```

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.56. Set Audio Callback PlayM4_SetAudioCallBack

```
BOOL _stdcall PlayM4_SetAudioCallBack (LONG nPort, void (_stdcall * funAudio) (long nPort, char * pAudioBuf, long nSize, long nStamp, long nType, long nUser), long nUser);
```

Description:

Register a callback function to refer to the wave format data that have been decoded out.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

funAudio

the callback function pointer.

nUser

user data.

Description of the callback function:

void _stdcall Audio (long nPort, char * pAudioBuf, long nSize, long nStamp, long nType, long nUser)

nPort

port

pAudioBuf

wave data.

nSize wave

data size.

nStamp

time stamp (ms)

nType

audio type, now only T_AUDIO16: 8KHZ, mono, 16bit per sample.

nUser

user data.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This is a reserved function not supported by SDK V6.1.1.8.

6.57. Set File End Message PlayM4_SetFileEndMsg

BOOL PlayM4_SetFileEndMsg (LONG nPort, HWND hWnd, UINT nMsg);

Description:

Register a windows message which will be post when the file reaches its end. For player SDK version above V2.4, when the file is end, the decoding thread will not stop by itself, and the users will need to call **PlayM4_Stop (nPort) after** received this message to stop the playback**.**

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

HWnd

[in] The handle to the window to receive this message.

nMsg

[in] The message is defined by an application. When received this message, it means that the file which is playing is end.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

PlayM4_SetFileEndMsg and PlayM4_SetFileEndCallback cannot be called at the same time.

6.58. Set File End Callback PlayM4_SetFileEndCallback

BOOL_stdcall PlayM4_SetFileEndCallback (LONG nPort, void (CALLBACK *FileEndCallback) (long nPort, void *pUser), void *pUser);

Description:

Set callback for decoding file end. This API should be called before

PlayM4_OpenSteam () or PlayM4_OpenFile ()

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

FileEndCallback

[in] Callback function described below

pUser

[in]Parameter of callback function

Callback function description: void (CALLBACK*FileEndCallback) (long nPort, void *pUser)

Callback Parameters:

nPort:

Port number

pUser:

User defined parameters, as the last parameter pUser of PlayM4_SetFileEndCallback ().

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

For callback functions, as VB does not support multi-thread, thus problems may occur when calling API defined in VB under VC environment. Please refer to: **Microsoft Knowledge Base Article - Q198607 "PRB: Access Violation in VB Run-Time Using AddressOf "** for detail information.

Notice:

PlayM4_SetFileEndMsg and PlayM4_SetFileEndCallback cannot be called at the same time.

6.59. Set Encoding Resolution Change Message PlayM4_SetEncChangeMsg

BOOL _stdcall PlayM4_SetEncChangeMsg (LONG nPort, HWND hWnd, UINT nMsg)

Description:

The message should be sent when the encoding format is change during setting the decode.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

hWnd

[in] handle of the message sending window

nMsg

[in] user will receive this message in hWnd when the playing files come to an end. The parameter 'WParam' in this message is the returned nPort value.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

PlayM4_SetEncChangeMsg and PlayM4_SetEncTypeChangeCallBackcannot be called at the same time.

6.60. Set Encoding Resolution Change Callback PlayM4_SetEncTypeChangeCallBack

BOOL _stdcall PlayM4_SetEncTypeChangeCallBack (LONG nPort, void (CALLBACK *funEncChange) (long nPort, long nUser), long nUser);

Description:

The callback function to inform the change when the picture format is changed in encoding

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

funEncChange

Callback function;

nUser

User defined data

Callback Function Descripton:

void (CALLBACK *funEncChange) (long nPort, long nUser)

nPort

The port number of the player.

nUser

User defined data

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

Please call it before open files.

6.61. Set Throw B Frame Number PlayM4_ThrowBFrameNum

BOOL PlayM4_ThrowBFrameNum (LONG nPort, DWORD nNum);

Description:

Set frames of non-decoding B frame. Non-decoding B frame can reduce CPU usage rate. If there is no B frames in the stream, this function will not it doesn't work by setting this function. It can be used on condition of fast playing and supporting multi channels so that CPU is used too often.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to

PLAYM4_MAX_SUPPORTS-1.

nNum

[in] The number of B-frame that is not decoded. It ranges from 0 to 2.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.62. Check Discontinuous Frame Number PlayM4_CheckDiscontinuousFrameNum

BOOL _stdcall PlayM4_CheckDiscontinuousFrameNum (LONG nPort, BOOL bCheck)

Description:

Whether jump to the next I frame when frame index is not continuous.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bCheck

[in]Whether jump to the next I frame if frame index is not continuous.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This function is reserved and not supported in SDK V6.1.1.8.

6.63. Set Secret Key PlayM4_SetSecretKey

BOOL __stdcall PlayM4_SetSecretKey (LONG nPort, LONG lKeyType, char *pSecretKey, LONG lKeyLen);

Description:

If the user has set secret key during encoding, then the user need to call this function before decoding. This function should be called before PlayM4_OpenSteam or PlayM4_OpenFile.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

lKeyType

[in] secret key type

pSecretKey

[in] secret key string

lKeyLen

[in] secret key length in bits

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Display Operation

6.64. Set Overlay Mode PlayM4_SetOverlayMode

BOOL PlayM4_SetOverlayMode (LONG nPort, BOOL bOverlay, COLORREF colorKey) ;

Description:

Specifies the overlay surface.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bOverlay

[in] Specifies the display surface. If it is TRUE, the player tries to create overlay surface. If it is FALSE, the player creates off-screen surface.

colorKey

[in] If bOverlay is TRUE, It is valid. And the value specifies the color that is covered on the primary surface. We can see the displaying only through this color.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.65. Get Overlay Mode PlayM4_GetOverlayMode

LONG PlayM4_GetOverlayMode (LONG nPort);

Description:

Check the player uses OVERLAY or not.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function returns 0, then the display surface is an off-screen surface; otherwise the display surface is an overlay surface.

6.66. Get Overlay Color Key PlayM4_GetColorKey

COLORREF PlayM4_GetColorKey (LONG nPort);

Description:

Get the color key of the OVERLAY surface.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The color key value.

6.67. Set Display Region PlayM4_SetDisplayRegion

BOOL _stdcall PlayM4_SetDisplayRegion (LONG nPort, DWORD nRegionNum, RECT *pSrcRect, HWND hDestWnd, BOOL bEnable);

Description:

Set or increase display regions. Can realize part-region display enlargement

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nRegionNum

[in]Display region number 0~ (MAX_DISPLAY_WND-1). If nRegionNum is 0, it means to set for main display windows (windows that are set in PlayM4_Play) and that will neglect location of hDestWnd and bEnable.

pSrcRect

[in]Set in the region where original image will be displayed, for instance: if the resolution of original image is 352*288, the set range of pSrcRect is (0, 0, 352, 288). If pSrcRect=NULL, the whole image will be displayed.

hDestWnd

[in]Set display window. If the window of this region has been set (opend), then neglect the parameter.

bEnable

[in]open (set) or close display region.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.68. Refresh Display PlayM4_RefreshPlay

BOOL PlayM4_RefreshPlay (LONG nPort) ;

Description:

Refresh display window. If the user has refreshed the display window under pause status, the displayed image will be vanished. And then the users can call this API to retrieve the image again. This API is only valid under pause and step forward status, otherwise it will return directly.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.69. Refresh Display in Multiple Regions PlayM4_RefreshPlayEx

BOOL _stdcall PlayM4_RefreshPlayEx (LONG nPort, DWORD nRegionNum);

Description:

Refresh display, It is an extended parameter set for PlayM4_SetDisplayRegion.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nRegionNum

[in]display region serial No..

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.70. Set Display Type PlayM4_SetDisplayType

BOOL PlayM4_SetDisplayType (LONG nPort, LONG nType);

Description:

Set display type. When displaying small image, it can reduce workload of display adapter if using DISPLAY_QUARTER. But image quality is lower. Please choose to use DISPLAY_NORMA when displaying normal or large image.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nType

[in]DISPLAY_NORMAL or DISPLAY_QUARTER

Description of macro definition:

DISPLAY_NORMAL Transmit normal resolution data to display adapter

DISPLAY_QUARTER Transmit 1/4 resolution data to display adapter

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.71. Get Display Type PlayM4_GetDisplayType

long PlayM4_GetDisplayType (LONG nPort);

Description:

Receive display type presently set

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

DISPLAY_NORMAL or **DISPLAY_QUARTER**

Buffer Operation

6.72. Get Source Buffer Remain PlayM4_GetSourceBufferRemain

DWORD PlayM4_GetSourceBufferRemain (LONG nPort);

Description:

Get the size of the data remained in the source buffer.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The size of the data remained.

6.73. Set Source Buffer Callback PlayM4_SetSourceBufCallBack

BOOL PlayM4_SetSourceBufCallBack (LONG nPort, DWORD nThreShold, void (CALLBACK * SourceBufCallBack) (long nPort, DWORD nBufSize, DWORD dwUser, void*pResvered), DWORD dwUser, void *pReserved);

Description:

Register a callback function when source buffer data length decreases and get lower than the threshold. Users need to call PlayM4_ResetSourceBufFlag every time after this callback is triggered to make the triggering mechanism valid again.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nThreShold

[in] the threshold.

SourceBufCallBack

[in] Address of a callback function.

dwUser

[in] user data.

pResvered

[in] reserved .

Description of the callback function:

void CALLBACK SourceBufCallBack (long nPort, DWORD nBufSize, DWORD dwUser, void*pContext)

Parameters:

nPort

The port number of the player

nBufSize

Number of bytes remained of the source buffer.

dwUser

user data .

pResvered

reserved.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.74. Reset Source Buffer Flag PlayM4_ResetSourceBufFlag

BOOL PlayM4_ResetSourceBufFlag (LONG nPort);

Description:

Reset the Callback flag of Source Buffer Callback;

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.75. Set Max Buffer Frame Number PlayM4_SetDisplayBuf

BOOL PlayM4_SetDisplayBuf (LONG nPort, DWORD nNum);

Description:

Set buffer size for playback (buffer of decoded images). The buffer size is directly related with the fluency and time-delay during playback. Larger buffer size will usually cause higher fluency and longer time delay with a certain range. Therefore it is suggested to increase buffer size (if the system memory size is large enough). The default buffer size would be 15 (frames), which is about 0.6 sec under 25fps configuration for file mode; and 10 (frames) for stream mode. Users can decrease this value to get shorter delay.

This function should be called between PlayM4_OpenStream and PlayM4_Play.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nNum

[in] The number of images to be buffered. It ranges from MIN_DIS_FRAMES to MAX_DIS_FRAMES.

Macro Definition:

MIN_DIS_FRAMES

The minimum number of image frames to be buffered.

MAX_DIS_FRAMES

The maximum number of image frames to be buffered.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.76. Get Buffer Frame Number PlayM4_GetDisplayBuf

DWORD PlayM4_GetDisplayBuf (LONG nPort);

Description:

Get the maximum number of images to be buffered.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

The maximum number of image frames to be buffered.

6.77. Reset Source Buffer PlayM4_ResetSourceBuffer

BOOL PlayM4_ResetSourceBuffer (LONG nPort);

Description:

Clear the source buffer.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.78. Reset Specified Buffer PlayM4_ResetBuffer

BOOL _stdcall PlayM4_ResetBuffer (LONG nPort, DWORD nBufType);

Description:

Clear the specified buffer.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nBufType

[in] buffer type.See the macro.

Macro Definition:

BUF_VIDEO_SRC	The souce video buffer, only used at stream mode.
BUF_AUDIO_SRC	The souce audio buffer, only used at stream mode.
BUF_VIDEO_RENDER	The video render buffer.
BUF_AUDIO_RENDER	The audio render buffer.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.79. Get Buffer Remain PlayM4_GetBufferValue

DWORD _stdcall PlayM4_GetBufferValue (LONG nPort, DWORD nBufType);

Description:

Get the information of the specified buffer.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nBufType

[in] buffer type.See the macro.

Return:

If nBufType is BUF_VIDEO_SRC or BUF_AUDIO_SRC return the size of the remain data in the souce buffer, else, return the number of frames in the render buffer. One audio frame includes 40ms audio data.

File Reference

6.80. Set File Reference Callback PlayM4_SetFileRefCallBack

BOOL PlayM4_SetFileRefCallBack (LONG nPort,
    void (_stdcall *pFileRefDone) (DWORD nPort, DWORD nUser), DWORD nUser);

Description:

Register a callback function when the file index for the key frames is created. If the callback does not function, it indicates errors in the indexing file.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

SetFileRefCallBack

[in] Address of a function to be called.

nUser

[in] user data.

Description of the callback function:

void FileRefDone (DWORD nPort, DWORD nUser)

Callback Parameters:

nPort

The port number of the player.

nUser

user data.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.81. Get Key Frame Position PlayM4_GetKeyFramePos

BOOL PlayM4_GetKeyFramePos (LONG nPort, DWORD nValue, DWORD nType, PFRAME_POS pFramePos);

Description:

Get the nearest key frame position before the designating position. Decoding process must start from a key frame, and any data before the first key frame will be neglected. Therefore if the users need to achieve video clip function, the clip file should start with a key frame (it doesn't matter if it ends with key frame or not, and the maximum frame loss number at the file end position will not exceed 3 frames).

PlayM4_GetKeyFramePos and PlayM4_GetNextKeyFramePosition can be called to achieve video clip function if the file index has been successfully created. The precision of clip process is related with key frame interval.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nValue

[in] The current position , it is the time in milliseconds or the number of frame

specified by nType .

nType

[in] The value type. It is BY_FRAMENUM or BY_FRAMETIME.

pFramePos

[out] Points to a FRAME_POS structure to receive the information of a key frame.

Description of the structure:

```
typedef struct{
    long nFilePos;          //The position of the file.
    long nFrameNum;         // The number of a frame.
    long nFrameTime;        //The time stamp of a frame (ms).
}FRAME_POS, *PFRAME_POS;
```

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.82. Get the Next Key Frame Position PlayM4_GetNextKeyFramePos

```
BOOL PlayM4_GetNextKeyFramePos (LONG nPort, DWORD nValue, DWORD nType, PFRAME_POS pFramePos);
```

Description:

Get the position of a key frame after the position specified by **nValue**.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nValue

[in] The current position , it is the time in milliseconds or the number of frame specified by nType .

nType

[in] The value type. It is BY_FRAMENUM or BY_FRAMETIME.

pFramePos

[out] Points to a FRAME_POS structure to receive the information of a key frame.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.83. Get File Reference PlayM4_GetRefValue

```
BOOL _stdcall PlayM4_GetRefValue (LONG nPort, BYTE *pBuffer, DWORD *pSize);
```

Description:

Get the file reference information. User must call this after the file reference has been created.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pBuffer

[in] The buffer to save the index information.

pSize

[in/out] Input the pBuffer size. Output the index information size.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.84. Set Reference Value PlayM4_SetRefValue

```
BOOL _stdcall PlayM4_SetRefValue (LONG nPort, BYTE *pBuffer, DWORD nSize);
```

Description:

Set the file index information. If the file reference already exists, then the user can input it directly via this function and does not need to call PlayM4_SetFileRefCallBack.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pBuffer

[in]The file index information.

nSize

[in]The size of the index information

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

The length and content of the file reference need to be correct. This function should be called after PlayM4_OpenFile.

Multi-screen Playback

Notice1: The Following APIs in this section are specially added for multi display adapters support. Only operation systems sith Windows98, Windows2000 and Windows2000 supports multi display adapters and needed installing DirectX6.0 or more advanced edition. If user needn't environment of supporting multi display adapters, these interfaces are dismissal. With regards to multi display adapters, please consult correlative file "Multiple-Monitor Systems " of Microsoft sdk.

Notice2: SDK version above V6.1.1.0 is self-adaptive to multi-screen display, and users do not need to call the APIs in this section manually.

6.85. Enum the Display Devices in the System PlayM4_InitDDrawDevice

BOOL PlayM4_InitDDrawDevice();

Description:

Enumerate display devices of system

Parameters:

--

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.86. Release Display Device Resource PlayM4_ReleaseDDrawDevice

void PlayM4_ReleaseDDrawDevice ();

Description:

Release resource distributed in the course of enumerating display devices

6.87. Get Display Adapter Number PlayM4_GetDDrawDeviceTotalNums

DWORD PlayM4_GetDDrawDeviceTotalNums ();

Description:

Get the total number of display device that are attached to the desktop.

Return:

If return 0, it indicates that there is only main displayed device in system. If 0, it indicates that there are many video adapters in system but only one is tied to windows desktop. If return other value, it indicates the number of video adapter tied to desktop of system. In the system with many video adapter, user can designate any video adapter as main display device via setting display attribution.

6.88. Assign Display Adapter for the Monitor PlayM4_SetDDrawDevice

BOOL PlayM4_SetDDrawDevice (LONG nPort, DWORD nDeviceNum);

Description:

Assign display adapter for the monitor. Notice: The display region should be set consistently.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nDeviceNum

[in] The number of the display device. It ranges from 0 to the return value of PlayM4_GetDDrawDeviceTotalNums. If 0, it means the primary display device.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.89. Assign Display Adapter for Multiple Monitors PlayM4_SetDDrawDevice_EX

BOOL _stdcall PlayM4_SetDDrawDeviceEx (LONG nPort, DWORD nRegionNum, DWORD nDeviceNum);

Description:

Set display adapter used in play window, as 62. It is an added parameter set for PlayM4_SetDisplayRegion.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

nRegionNum

[in]display region serial No.

[in]The video adapter No.

nDeviceNum

[in] The number of the display device. It ranges from 0 to the return value of PlayM4_GetDDrawDeviceTotalNums. If 0, it means the primary display device.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This is a reserved function not supported by SDK V6.1.1.8.

6.90. Get the Display Adapter and Monitor Info PlayM4_GetDDrawDeviceInfo

BOOL PlayM4_GetDDrawDeviceInfo (DWORD nDeviceNum, LPSTR lpDriverDescription, DWORD nDespLen, LPSTR lpDriverName , DWORD nNameLen, HMONITOR *hhMonitor);

Description:

Get the information of the display device and monitor specified by nDeviceNum.

Parameters:

nDeviceNum

[in]The number of the display device. If it is zero, It means the primary display

device.

nDespLen

[in]The number of bytes of the lpDriverDescription that has been allocated.

nNameLen

[in] The number of bytes of the lpDriverName that has been allocated.

lpDriverDescription

[out] Address of a string that contains the driver description.

lpDriverName

[out] Address of a string that contains the driver name.

hhMonitor

[out] Handle of the monitor associated with the enumerated DirectDraw object. This parameter is NULL when the enumerated DirectDraw object is for the primary device, a nondisplay device (such as a 3-D accelerator with no 2-D capabilities), or devices not attached to the desktop. For more information, see the function of GetMonitorInfo (Windows API). Notice: the type of HMONITOR is defined in the header file “windef.h” (_WIN32_WINNT >= 0x0500). Please download and install the new Platform sdk. (http: //www.microsoft.com/msdownload/platformsdk/sdkupdate/)

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.91. Get System Info of Display Devices PlayM4_GetCapsEx

```
int   PlayM4_GetCapsEx (DWORD nDDrawDeviceNum);
```

Description:

Get some capabilities of your system.

Parameters:

nDeviceNum

[in]The number of the display device. If it is zero, It means the primary display device.

Return:

SUPPORT_DDRAW

Support DIRECTDRAW. If not, player can’t work.

SUPPORT_BLT

Support BLT operation. If not, player can’t work.

SUPPORT_BLTFOURCC

Display hardware is capable of color-space conversions during blit operations.

SUPPORT_BLTSHRINKX

Supports arbitrary shrinking along the x-axis (horizontally), this flag is valid only for BLT operations.

SUPPORT_BLTSHRINKY

Supports arbitrary shrinking along the y-axis (vertically), this flag is valid only for BLT operations.

SUPPORT_BLTSTRETCHX

Supports arbitrary stretching of a surface along the x-axis (horizontally), this flag is

valid only for blit operations.

SUPPORT_BLTSTRETCHY

Supports arbitrary stretching of a surface along the y-axis (vertically), this flag is valid only for blit operations.

SUPPORT_SSE

Supports SSE instruction set, if supports, It will get high performance.

SUPPORT_MMX

Supports MMX instruction set.

Snapshot

6.92. Snapshot Callback PlayM4_SetDisplayCallBack

BOOL PlayM4_SetDisplayCallBack (LONG nPort, void (CALLBACK* DisplayCBFun) (long nPort, char * pBuf, long nSize, long nWidth, long nHeight, long nStamp, long nType, long nReserved));

Description:

Register a callback function to capture pictures. The callback is triggered in clock thread, and any time-consuming operation will interfere with the clock pulse and cause display problems. Therefore it is suggested to return ASAP. If the users need to stop the callback, please set the DisplayCDFun as NULL.

This callback can be called at any time, and then it will always be valid until the application exits.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

DisplayCBFun

[in] Address of a function to be called, can be NULL ;

Description of the callback function:

nPort

The port number of the player.

pBuf

Pointer to the buffer that receives the data of picture get from the player.

nSize

Specifies the number of bytes to be get from the player.

nWidth

width of the picture in pixels.

nHeight

height the picture in pixels.

nStamp

time stamp in milliseconds

nType

Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro definition for detail information.

nReserved

Reserved;

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This callback function needs to be returned as fast as possible. If the user wants to stop the callback, please set the DisplayCBFun as NULL. Once the callback is set, it will

be valid until the program exits, and can be called at any time.

6.93. Convert Video Image to BMP File PLayM4_ConvertToBmpFile

BOOL PLayM4_ConvertToBmpFile (char * pBuf, long nSize, long nWidth, long nHeight, long nType, char *sFileName);

Description:

Converts the YUV picture to a bmp file.

Parameters:

pBuf

Pointer to the buffer that receives the data of picture get from the player.

nSize

Specifies the number of bytes to be get from the player.

nWidth

width of the picture in pixels.

nHeight

height the picture in pixels.

nType

Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro definition for detail information.

sFileName

The bmp file name to be saved.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

The converting process takes CPU resource, and is not suggested to call unless necessary.

6.94. Convert Video Image to JPEG File PlayM4_ConvertToJpegFile

BOOL _stdcall PlayM4_ConvertToJpegFile (char *pBuf, long nSize, long nWidth, long nHeight, long nType, char *sFileName)

Description:

Convert the yuv image to a jpeg file.

Parameters:

pBuf

Pointer to the buffer that receives the data of picture get from the player.

nSize

Specifies the number of bytes to be get from the player.

nWidth

width of the picture in pixels.

nHeight

height the picture in pixels.

nType

Received data type: T_YV12, T_RGB32, T_UYVY, please refer to the macro

definition for detail information.

sFileName

The jpeg file name to be saved.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

The converting process takes CPU resource, and is not suggested to call unless necessary.

6.95. BMP Snapshot PlayM4_GetBMP

BOOL _stdcall PlayM4_GetBMP (LONG nPort, PBYTE pBitmap, DWORD nBufSize, DWORD*pBmpSize);

Description:

Capture BMP image

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pBitmap

[in] Address assigned by users for storing BMP data, no less then bmp file size: sizeof (BITMAPFILEHEADER) + sizeof (BITMAPINFOHEADER) + w * h * 4, in which w and h stand for the width and height of the image.

nBufSize

[in] assigned buffer size

pBmpSize

[out] Actual BMP size captured.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.96. JPEG Snapshot PlayM4_GetJPEG

BOOL _stdcall PlayM4_GetJPEG (LONG nPort, PBYTE pJpeg, DWORD nBufSize, DWORD*pJpegSize);

Description:

Capture JPEG image

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pJpeg

[in] Address assigned by users for storing JPEG data, no less then jpeg file size: suggested w * h * 3/2, in which w and h stand for the width and height of the image.

nBufSize

[in] assigned buffer size

pBmpSize

[out]Actual Jpeg image size captured.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.97. Set JPEG Snapshot Quality PlayM4_SetJpegQuality

```
BOOL _stdcall PlayM4_SetJpegQuality (long nQuality);
```

Description:

Set the quality of the jpeg snapshots.

Parameters:

nQuality

[in] the quality of jpeg file, the greater its value is , the better quality of the jpeg file is . It ranges from 0 to 100 (80 by default).

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice*:*

The recommended value is between 75 and 90 .

Others

6.98. Set Draw Function Callback PlayM4_RigisterDrawFun

BOOL _stdcall PlayM4_RigisterDrawFun (LONG nPort, void (CALLBACK* DrawFun) (long nPort, HDC hDc, LONG nUser), LONG nUser);

Description:

Regist a callback fuction to get the off-screen surface device context.If you want to draw text or others on the display window, you must regist it. The call back funtion will be called before bltting each time. The DC is like as the client window DC.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

DrawFun

[in] The callback function pointer.

nUser

[in] User data.

Description of the callback function:

void CALLBACK DrawFun (long nPort, HDC hDc, LONG nUser);

Callback Parameters:

nPort

[out] The port number of the player.

hDc

[out] The off-screen surface DC.

nUser

[out] The user data.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.99. Set Data Verify Callback PlayM4_SetVerifyCallBack

BOOL _stdcall PlayM4_SetVerifyCallBack (LONG nPort, DWORD nBeginTime, DWORD nEndTime, void (_stdcall* funVerify) (long nPort, FRAME_POS * pFilePos, DWORD bIsVideo, DWORD nUser), DWORD nUser);

Description:

Register a callback function to verify the data. It checks the water marker. If the data have been changed, it will call the function that have been registered. Notice: the function must be called before PlayM4_OpenFile, and you must call the function PlayM4_SetFileRefCallBack together (see 53).

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to

PLAYM4_MAX_SUPPORTS-1.

nBeginTime

verify start time (ms);

nEndTime

verify stop time (ms);

funVerify

the callback function pointer;

nUser

user data.

Description of the callback function :

void _stdcall Verify (long nPort, FRAME_POS * pFilePos, DWORD bIsVideo, DWORD nUser);

nPort

port

pFilePos

file position.

bIsVideo

1-video data, 0-audio data

nUser

user data.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This function is not supported in V6.1.1.8 SDK.

6.100. Get Original Frame Callback PlayM4_GetOriginalFrameCallBack

BOOL _stdcall PlayM4_GetOriginalFrameCallBack (LONG nPort, BOOL bIsChange, BOOL bNormalSpeed, long nStartFrameNum, long nStartStamp, long nFileHeader, void (CALLBACK *funGetOrignalFrame) (long nPort, FRAME_TYPE *frameType, long nUser), long nUser)

Description:

Create callback function to get the original frame data. You can change the time stamp and no. of each frame. The API is used after the file is opened. Used to combine two files.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bIsChange

[in] Change the parameters of each frame or not

bNormalSpeed

[in] Get the original frame at normal speed or not

nStartFrameNum

[in] If the user wants to change the original frame number, this is the start frame number of new file.

nStartStamp

[in] If the user wants to change the original frame time stamp, this is the start time stamp of the new file.

nFileHeader

[in] The version information of the file header, if the version is not matched, it will return failure.

Description of callback function:

void (CALLBACK *funGetOrignalFrame) (long nPort, FRAME_TYPE *frameType, long nUser)

nPort:
Channel number;
frameType:
data frame information

```
typedef struct{
    char *pDataBuf;          //the start address of data frame
    long nSize;              //frame size
    long nFrameNum;          //frame number
    BOOL bIsAudio;           //is audio frame or not
long nReserved;
}FRAME_TYPE;
```

nUser:
user data.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This function is not supported in V6.1.1.8 SDK.

6.101. Get File Attributes PlayM4_GetFileSpecialAttr

BOOL _stdcall PlayM4_GetFileSpecialAttr (LONG nPort, DWORD *pTimeStamp, DWORD *pFileNum , DWORD *nFileHeader)

Description:

Get the file last frame number and time stamp. Used after the file is opened and combine with front file.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

pTimeStamp: [out] the file end time stamp;

pFileNum: [out] the file end frame number;

nfileHeader: [out] the file header information.

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

Notice:

This function is not supported in V6.1.1.8 SDK

6.102. Jump to Next Key Frame on Error PlayM4_PlaySkipErrorData

BOOL PlayM4_PlaySkipErrorData(LONG nport, BOOL bSkip)

Description:

Users can enable this function to avoid blurring or other display problems caused by error in data.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bSkip: [in] TRUE-Skip to the next key frame if there is error in video data; FALSE-continue with the current decoding(seek for the next frame for decoding)

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

6.103. Check Frame's Continuity PlayM4_CheckDiscontinueFrameNum

BOOL PlayM4_CheckDiscontinueFrameNum(LONG nport, BOOL bCheck)

Description:

If the frame number is not continuous, the decoder will automatically jump to the next key frame when this function is enabled.

Parameters:

nPort

[in] The port number of the player. It ranges from 0 to PLAYM4_MAX_SUPPORTS-1.

bCheck: [in] TRUE-enable frame continuity checking; FALSE- disable frame continuity checking

Return:

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.