# DVP7010B/7020B Functions Library

# Library: DVP7010B.dll

## Data Type Summary

| | |
|---|---|
| Res | The function returns value |

## Method Summary

| **SDK Initialize and close** | |
|---|---|
| AdvDVP_CreateSDKInstance | Creates SDK instance |
| AdvDVP_InitSDK | Initializes the SDK |
| AdvDVP_CloseSDK | Closes up the SDK. |

| **Capture control** | |
|---|---|
| AdvDVP_GetNoOfDevices | Gets the number of video capture devices |
| AdvDVP_Start | Starts video capturing |
| AdvDVP_Stop | Stops video capturing |
| AdvDVP_GetCapState | Gets capture state |
| AdvDVP_SetNewFrameCallback | Sets a callback function for SDK |
| AdvDVP_GetCurFrameBuffer | Gets current frame buffer |

| **Capture setting** | |
|---|---|
| AdvDVP_GetVideoFormat | Gets video input format |
| AdvDVP_SetVideoFormat | Sets video input format |
| AdvDVP_GetFrameRate | Gets frame rate |
| AdvDVP_SetFrameRate | Sets frame rate |
| AdvDVP_GetResolution | Gets video resolution |
| AdvDVP_SetResolution | Sets video resolution |
| AdvDVP_GetVideoInput | Gets video input mux |
| AdvDVP_SetVideoInput | Sets video input mux |

| **Sensor Control** | |
|---|---|
| AdvDVP_GetBrightness | Gets brightness value |
| AdvDVP_SetBrightness | Sets brightness value |
| AdvDVP_GetContrast | Gets contrast value |
| AdvDVP_SetContrast | Sets contrast value |
| AdvDVP_GetHue | Gets hue value |
| AdvDVP_SetHue | Sets hue value |
| AdvDVP_GetSaturation | Gets saturation value |
| AdvDVP_SetSaturation | Sets saturation value |

| **GPIO** | |
|---|---|
| AdvDVP_GPIOGetData | Gets value of specified GPIO pin |
| AdvDVP_GPIOSetData | Sets value of specified GPIO pin |

| **De-interlace Control** | |
|---|---|
| AdvDVP_SetDeinterlace | Enables / Disables de-interlace function |
| AdvDVP_GetDeinterlace | Gets de-interlacing state |

# DVP7010B/7020B Encoding Functions Library

# Library: DVP7010BEnc.dll

# Encoder: rmp4.dll

Before using the DVP7010B/7020B encoding functions library, the “RMP4” codec must be installed to the system. After installing the sample program, the codec will be installed automatically. You can install the codec manually by using the “rmp4.inf” file. Right click on the file, and then click “Install”.

## Data Type Summary

| | |
|---|---|
| EncRes | The function returns value |
| PSTREAMREADBEGIN | The stream Read Begin function pointer |
| PSTREAMREADPROC | The Stream Read Process function pointer |
| PSTREAMREADEND | The Stream Read End function pointer |
| STREAMREAD_STRUCT | The structure stores the Stream Read callback function pointers |

# Method Summary

| **SDK Initialize and close** | |
|---|---|
| AdvDVP_CreateEncSDKInstance | Creates encoding SDK instance |
| AdvDVP_InitSDK | Initializes the SDK |
| AdvDVP_CloseSDK | Closes up the SDK |
| AdvDVP_InitEncoder | Opens and initializes video encoder |
| AdvDVP_CloseEncoder | Closes and releases video encoder |

| **Encode control** | |
|---|---|
| AdvDVP_StartVideoEncode | Starts video encoding |
| AdvDVP_VideoEncode | Encodes one video frame |
| AdvDVP_StopVideoEncode | Stops video encoding |
| AdvDVP_GetState | Gets encoder state |
| AdvDVP_CreateAVIFile | Creates an AVI file |
| AdvDVP_WriteAVIFile | Writes video data to the AVI file |
| AdvDVP_CloseAVIFile | Closes AVI file |
| AdvDVP_SetStreamReadCB | Sets the stream read callback functions for SDK |

| **Encode setting** | |
|---|---|
| AdvDVP_GetVideoQuant | Gets video encoding quant |
| AdvDVP_SetVideoQuant | Sets video encoding quant |
| AdvDVP_GetVideoFrameRate | Gets video encoding frame rate |
| AdvDVP_SetVideoFrameRate | Sets video encoding frame rate |
| AdvDVP_GetVideoResolution | Gets video encoding resolution |
| AdvDVP_SetVideoResolution | Sets video encoding resolution |
| AdvDVP_GetVideoKeyInterval | Gets video encoding key interval |
| AdvDVP_SetVideoKeyInterval | Sets video encoding key interval |

# DVP7010B/7020B Player Functions Library

# Library: DVP7010BPlayer.dll

# Decoder: rmp4.dll

Before using the DVP7010B/7020B player functions library, the “RMP4” codec must be installed to the system. After installing the sample program, the codec will be installed automatically. You can install the codec manually by using the “rmp4.inf” file. Right click on the file, and then click “Install”.

## Data Type Summary

| | |
|---|---|
| PlayerRes | The function returns value |

## Method Summary

| **Playback SDK initialize** | |
|---|---|
| AdvDVP_CreatePlayerSDKInstance | Creates Player SDK instance |

| **Playback control** | |
|---|---|
| AdvDVP_OpenFile | Opens file and initializes player |
| AdvDVP_CloseFile | Closes file that has been opened |
| AdvDVP_Play | Plays file that has been opened |
| AdvDVP_Pause | Pauses or continues |
| AdvDVP_Stop | Stops playing file |
| AdvDVP_Fast | Plays file with faster speed |
| AdvDVP_Slow | Plays file with slower speed |
| AdvDVP_PlayStep | Plays by single frame |
| AdvDVP_GetStatus | Gets playback state |
| AdvDVP_GetCurImage | Gets frame that is rendered |

| | |
|---|---|
| AdvDVP_RegNotifyMsg | Registers message sent to player when event occurs |
| AdvDVP_CheckFileEnd | Checks if file playing is finished |

| **Playback setting** | |
|---|---|
| AdvDVP_GetVideoResolution | Gets video resolution of file |
| AdvDVP_GetFileTime | Gets total file time |
| AdvDVP_GetPlayedTime | Gets current file time |
| AdvDVP_SetPlayPosition | Locates position of file |
| AdvDVP_GetFileTotalFrames | Gets total frame number of file |
| AdvDVP_GetPlayedFrames | Gets current frame number of file |
| AdvDVP_GetPlayRate | Gets current playing rate |

# DVP7010B/7020B Functions Reference

## Data Type

## Res

**Syntax**

```
typedef enum tagRes
{
    SUCCEEDED           = 1,
    FAILED              = 0,
    SDKINITFAILED       = -1,
    PARAMERROR          = -2,
    NODEVICES           = -3,
    NOSAMPLE            = -4,
    DEVICENUMERROR      = -5,
    INPUTERROR          = -6,
} Res;
```

**Description**

The function returns value.

# Method

## AdvDVP_CreateSDKInstance

**Syntax**

int AdvDVP_CreateSDKInstance(void **pp)

**Parameters**

[ OUT ] pp: A pointer to the SDK instance.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| PARAMERROR: | Parameter error. |

**Description**

This function creates SDK instance. You can delete the pointer which is carried by the parameter named “pp” to free this SDK instance before you free the DLL handle.

# AdvDVP_InitSDK

**Syntax**

int AdvDVP_InitSDK()

**Parameters**

None

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| NODEVICES: | No devices found. |

**Description**

This function initializes all video capture devices in the system. After initializing each device, the capture status would be set as “STOPPED”.

**See Also**

AdvDVP_GetNoOfDevices
AdvDVP_GetCapState
AdvDVP_CloseSDK

# AdvDVP_CloseSDK

**Syntax**

int AdvDVP_CloseSDK(void)

**Parameters**

None

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| SDKINITFAILED: | SDK not initialized. |

**Description**

This function cleans all instances of capture devices and closes up the SDK.

**See Also**

AdvDVP_InitSDK

# AdvDVP_GetNumberOfDevices

**Syntax**

int AdvDVP_GetNoOfDevices(int *pNoOfDevs)

**Parameters**

[ OUT ] pNoOfDevs: A pointer to get the number of video capture devices.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |

**Description**

This function gets the number of video capture devices in the system.

# AdvDVP_Start

**Syntax**

int AdvDVP_Start(int nDevNum, int SwitchingChans, HWND Main, HWND hwndPreview)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

[ IN ] SwitchingChans: Single video input or switching between video muxes.
0: single channel.
1: two channels (mux0, mux1).
2: three channels (mux0, mux1, mux2).
3: four channels (mux0, mux1, mux2, mux3).

[ IN ] Main: A main window handle. The handle value can be NULL.

[ IN ] hwndPreview: A windows handle for display area. This parameter is only valid when the "SwitchingChans)" is zero. When the value of this parameter is NULL, the video will not be rendered.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |

**Description**

This function starts video capturing on a specified capture port. The capture state would be set as "RUNNING" after a successful start. If the channels share frames (i.e. SwitchingChans>0), the video input mux will be set to 0.

**See Also**

AdvDVP_Stop
AdvDVP_GetCapState

# AdvDVP_Stop

**Syntax**

int AdvDVP_Stop(int nDevNum)

**Parameters**

[ IN ] nDevNum:  Specifies the device number(0~3).

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |

**Description**

This function stops video capturing on a specified capture port. The capture state would be set as “STOPPED” after a successful stop.

**See Also**

AdvDVP_Start

AdvDVP_GetCapState

# AdvDVP_GetCapState

**Syntax**

int AdvDVP_GetCapState(int nDevNum)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

**Return Value**

DEVICENUMERROR: Invalid device number.
SDKINITFAILED: SDK not initialized.

**Description**

This function gets capture state of a specified capture port.

```
typedef enum {
    STOPPED          = 1,
    RUNNING          = 2,
    UNINITIALIZED    = -1,
    UNKNOWNSTATE     = -2
} CapState;
```

**See Also**

AdvDVP_InitSDK
AdvDVP_Start
AdvDVP_Stop

# AdvDVP_GetCurFrameBuffer

**Syntax**

int AdvDVP_GetCurFrameBuffer(int nDevNum, long* bufSize, BYTE* buf, int VMux)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN, OUT ] bufSize: | Frame buffer size. |
| [ IN ] buf: | Frame buffer. |
| [ IN ] VMux: | Video mux. |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid parameter. |
| NOSAMPLE: | No buffer sample. |

**Description**

This function gets current frame buffer of a specified capture port. Call this function after the AdvDVP_Start, and it should sleep for a little time, such as 500 milliseconds.

The default video format is YUYV. To calculate the size of a frame, use the following formula:

Buffer Size = Image Height * Image Width * 2;

**See Also**

AdvDVP_Start

# AdvDVP_SetNewFrameCallback

**Syntax**

int AdvDVP_SetNewFrameCallback(int nDevNum, int callback)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

[ IN ] callback: Callback function.

Callback fumction type:

typedef int (*CAPCALLBACK)( int nID, int nDevNum, int VMux, int bufsize, BYTE* buf);

[ IN ] nID: Single video input ID or the video mux ID. The value of IDs is showed as the following:

```
#define ID_NEW_FRAME 37810
#define ID_MUX0_NEW_FRAME 37800
#define ID_MUX1_NEW_FRAME 37801
#define ID_MUX2_NEW_FRAME 37802
#define ID_MUX3_NEW_FRAME 37803
```

[ IN ] nDevNum: Specifies the device number(0~3).

[ IN ] VMux: Specifies the video mux number(0~3).

[ IN ] bufsize: An integer pointer of the frame buffer size.

[ IN ] buf: A BYTE pointer of the frame buffer.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |

**Description**

This function sets a callback function for SDK. When new frame arrives, messages and frame information will be sent to callback function.

**See Also**

# AdvDVP_GetVideoFormat

**Syntax**

int AdvDVP_GetVideoFormat(int nDevNum, AnalogVideoFormat* vFormat)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).
[ OUT ] Vformat: A pointer to get video format.

```
typedef enum tagAnalogVideoFormat
{
    Video_None        = 0x00000000,
    Video_NTSC_M      = 0x00000001,
    Video_NTSC_M_J    = 0x00000002,
    Video_PAL_B       = 0x00000010,
    Video_PAL_M       = 0x00000200,
    Video_PAL_N       = 0x00000400,
    Video_SECAM_B     = 0x00001000
} AnalogVideoFormat;
```

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid parameter. |

**Description**

This function gets video input format of a specified capture port.

**See Also**

AdvDVP_SetVideoFormat

# AdvDVP_SetVideoFormat

**Syntax**

int AdvDVP_SetVideoFormat(int nDevNum, AnalogVideoFormat* vFormat)

**Parameters**

[ IN ] nDevNum: Specifies the port device number(0~3).
[ IN ] Vformat: Video format:

```
typedef enum tagAnalogVideoFormat
{
    Video_None        = 0x00000000,
    Video_NTSC_M      = 0x00000001,
    Video_NTSC_M_J    = 0x00000002,
    Video_PAL_B       = 0x00000010,
    Video_PAL_M       = 0x00000200,
    Video_PAL_N       = 0x00000400,
    Video_SECAM_B     = 0x00001000
} AnalogVideoFormat;
```

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |

**Description**

This function sets video input format of a specified capture port. This function should be called before “AdvDVP_Start”.

**See Also**

AdvDVP_GetVideoFormat

# AdvDVP_GetFrameRate

**Syntax**

int AdvDVP_GetFrameRate(int nDevNum, int *FrameRate)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

[ OUT ] FrameRate: A pointer to get video frame rate.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid parameter. |

**Description**

This function gets frame rate of a specified capture port.

**See Also**

AdvDVP_SetFrameRate

# AdvDVP_SetFrameRate

**Syntax**

int AdvDVP_SetFrameRate(int nDevNum , int SwitchingChans, int FrameRate)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] SwitchingChans: | Single video input or switching between video muxes(0~3).<br>0: single channel.<br>1: two channels (mux0, mux1).<br>2: three channels (mux0, mux1, mux2).<br>3: four channels (mux0, mux1, mux2, mux3). |
| [ IN ] FrameRate: | A value to set frame rate. (0<FrameRate<=30, Default value is 30) |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid parameter. |

**Description**

This function sets frame rate of a specified capture port. This function should be called before “AdvDVP_Start”. If the channels share frames (i.e. SwitchingChans>0), the frame rate must be set to 30. Otherwise, the function will return PARAMERROR.

**See Also**

AdvDVP_GetFrameRate

# AdvDVP_GetResolution

**Syntax**

int AdvDVP_GetResolution(int nDevNum, VideoSize *Size)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).
[ OUT ] Size: A pointer to get video resolution.

```
typedef enum
{
    FULLPAL=0,          // (PAL: 768x576)
    SIZED1,             // (NTSC: 720x480, PAL: 720x576)
    SIZEVGA,            // (640x480)
    SIZEQVGA,           // (320x240)
    SIZESUBQVGA         // (160x120)
} VideoSize;
```

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid parameter. |

**Description**

This function gets video resolution of a specified capture port.

**See Also**

AdvDVP_SetResolution

# AdvDVP_SetResolution

**Syntax**

int AdvDVP_SetResolution(int nDevNum, VideoSize Size)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).
[ IN ] Size: A value to set video resolution.

```
typedef enum
{
    FULLPAL=0,          // (PAL: 768x576)
    SIZED1,             // (NTSC: 720x480, PAL: 720x576)
    SIZEVGA,            // (640x480)
    SIZEQVGA,           // (320x240)
    SIZESUBQVGA         // (160x120)
} VideoSize;
```

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |

**Description**

This function sets video resolution of a specified capture port. This function should be called before “AdvDVP_Start”.

**See Also**

AdvDVP_GetResolution

# AdvDVP_GetVideoInput

**Syntax**

int AdvDVP_GetVideoInput(int nDevNum, int* pInput)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).
[ OUT ] pInput: A pointer to get video input mux.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | The “pInput” parameter is invalid. |

**Description**

This function gets video input mux of a specified capture port.
It returns “FAILED” when argument “SwitchingChans” of AdvDVP_Start was set to nonzero. And, the video input mux will be set to 0 automatically when argument “SwitchingChans” of AdvDVP_Start was set to nonzero.

**See Also**

AdvDVP_Start
AdvDVP_SetVideoInput

# AdvDVP_SetVideoInput

**Syntax**

int AdvDVP_SetVideoInput(int nDevNum, int nInput)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).
[ IN ] nInput: A value to set video input mux(0~3).

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid video input. |

**Description**

This function sets video input mux of a specified capture port.
It returns “FAILED” when argument “SwitchingChans” of AdvDVP_Start was set to nonzero. And, the video input mux will be set to 0 automatically when argument “SwitchingChans” of AdvDVP_Start was set to nonzero.

**See Also**

AdvDVP_Start
AdvDVP_GetVideoInput

# AdvDVP_GetBrightness

**Syntax**

AdvDVP_GetBrightness(int nDevNum, int nInput, long *lpValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels. |
| [ OUT ] lpValue: | A long pointer to get brightness value. |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | The “lpValue” parameter is invalid. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function gets brightness value of a specified capture port.

**See Also**

AdvDVP_SetBrightness

# AdvDVP_SetBrightness

**Syntax**

int AdvDVP_SetBrightness(int nDevNum , int nInput, long lValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when no switching channels. |
| [ IN ] lValue: | A value to set brightness(0~100). |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid brightness value. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function sets brightness value of a specified capture port.

**See Also**

AdvDVP_GetBrightness

# AdvDVP_GetContrast

**Syntax**

int AdvDVP_GetContrast(int nDevNum, int nInput, long *lpValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when no switching channels. |
| [ OUT ] lpValue: | A long pointer to get contrast value. |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | The “lpValue” parameter is invalid. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function gets contrast value of a specified capture port.

**See Also**

AdvDVP_SetContrast

# AdvDVP_SetContrast

**Syntax**

int AdvDVP_SetContrast(int nDevNum, int nInput, long lValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels. |
| [ IN ] lValue: | A value to set contrast(0~100). |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device. |
| PARAMERROR: | Invalid contrast value. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function sets contrast value of a specified capture port.

**See Also**

AdvDVP_GetContrast

# AdvDVP_GetHue

**Syntax**

int AdvDVP_GetHue(int nDevNum, int nInput, long *lpValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels. |
| [ OUT ] lpValue: | A long pointer to get hue value. |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | The “lpValue” parameter is invalid. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function gets hue value of a specified capture port.

**See Also**

AdvDVP_SetHue

# AdvDVP_SetHue

**Syntax**

int AdvDVP_SetHue(int nDevNum, int nInput, long lValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels. |
| [ IN ] lValue: | A value to set hue(0~100). |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid hue value. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function sets hue value of a specified capture port.

**See Also**

AdvDVP_GetHue

# AdvDVP_GetSaturation

**Syntax**

int AdvDVP_GetSaturation(int nDevNum, int nInput, long *lpValue)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

[ IN ] nInput: Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels.

[ OUT ] lpValue: A long pointer to get saturation value.

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | The “lpValue” parameter is invalid. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function gets saturation value of a specified capture port.

**See Also**

AdvDVP_SetSaturation

# AdvDVP_SetSaturation

**Syntax**

int AdvDVP_SetSaturation(int nDevNum , int nInput, long lValue)

**Parameters**

| | |
|---|---|
| [ IN ] nDevNum: | Specifies the device number(0~3). |
| [ IN ] nInput: | Specifies the video input mux(-1~3). This value must be set to -1 when there are no switching channels. |
| [ IN ] lValue: | A value to set saturation(0~100). |

**Return Value**

| | |
|---|---|
| SUCCEEDED: | Function succeeded. |
| FAILED: | Function failed. |
| SDKINITFAILED: | SDK not initialized. |
| DEVICENUMERROR: | Invalid device number. |
| PARAMERROR: | Invalid saturation value. |
| INPUTERROR: | Invalid video input mux. |

**Description**

This function sets saturation value of a specified capture port.

**See Also**

AdvDVP_GetSaturation

# AdvDVP_GPIOGetData

**Syntax**

int AdvDVP_GPIOGetData(int nDevNum, int nDINum, BOOL* pValue)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

[ IN ] nDINum: Specifies the digital input number(0~3).

[ OUT ] pValue: A pointer to get the value of the specified digital input.

**Return Value**

SUCCEEDED: Function succeeded.

FAILED: Function failed.

SDKINITFAILED: SDK not initialized.

PARAMERROR: The “pValue” parameter is invalid.

DEVICENUMERROR: Invalid device number.

**Description**

This function gets the value of the specified digital input.

**See Also**

AdvDVP_GPIOSetData

# AdvDVP_GPIOSetData

**Syntax**

int AdvDVP_GPIOSetData(int nDevNum, int nDONum, BOOL bValue)

**Parameters**

[ IN ] nDevNum: Specifies the device number(0~3).

[ IN ] nDONum: Specifies the digital output number(0~3).

[ IN ] bValue: A value to set the value of the specified digital output.

**Return Value**

SUCCEEDED: Function succeeded.

FAILED: Function failed.

SDKINITFAILED: SDK not initialized.

PARAMERROR: Invalid parameter.

DEVICENUMERROR: Invalid device number.

**Description**

This function sets the value of the specified digital output.

**See Also**

AdvDVP_GPIOGetData

# AdvDVP_SetDeinterlace

**Syntax**

void AdvDVP_SetDeinterlace(BOOL bEnable)

**Parameters**

[ IN ] bEnable: TRUE for Enable; False for Disable.

**Return Value**

Void.

**Description**

This function enables or disables the de-interlace function.

**See Also**

AdvDVP_GetDeinterlace

# AdvDVP_GetDeinterlace

**Syntax**

BOOL AdvDVP_GetDeinterlace ()

**Parameters**

Void.

**Return Value**

| | |
|---|---|
| TRUE: | De-interlace is enabled. |
| FALSE: | De-interlace is disabled. |

**Description**

This function retrieves de-interlace state.

**See Also**

AdvDVP_SetDeinterlace

## Data Type

## EncRes

**Syntax**

```
typedef enum tagRes
{
    ENC_SUCCEEDED       = 1,
    ENC_FAILED          = 0,
    ENC_SDKINITFAILED   = -1,
    ENC_ENCINITFAILED   = -2,
    ENC_PARAMERROR      = -3,
    ENC_ENCNUMERROR     = -4,
    ENC_BUFFERFULL      = -5
} EncRes;
```

**Description**

The function returns value.

## PSTREAMREADBEGIN

### Syntax

void (*PSTREAMREADBEGIN)(int nEncNum)

### Parameters

[ IN ] nEncNum: Specifies the encoder number.

### Return Value

None

### Description

The pointer to the Stream Read Begin callback function will be called when the video stream read process begins.

### See Also

STREAMREAD_STRUCT

## PSTREAMREADPROC

### Syntax

void (*PSTREAMREADPROC)(int nEncNum, LPVOID pStreamBuf, long lBufSize, DWORD dwCompFlags)

### Parameters

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number. |
| [ IN ] pStreamBuf: | A pointer to the data buffer that stores an encoded video frame. |
| [ IN ] lBufSize: | Specifies the size of the encoded video frame. |
| [ IN ] dwCompFlags | Specifies if this encoded video frame is I-frame. The AVIIF_KEYFRAME value means the frame is I-frame. |

```
#define AVIIF_KEYFRAME  0x00000010L
```

### Return Value

None

### Description

The pointer to the Stream Read Process callback function will be called after every video frame is encoded. The user can use this function to get every encoded video frame.

### See Also

STREAMREAD_STRUCT

## PSTREAMREADEND

### Syntax

void (*PSTREAMREADEND)(int nEncNum)

### Parameters

[ IN ] nEncNum: Specifies the encoder number.

### Return Value

None

### Description

The pointer to the Stream Read End callback function will be called when the video stream read process is finished.

### See Also

STREAMREAD_STRUCT

## STREAMREAD_STRUCT structure

### Syntax

```
typedef struct
{
    void (*PSTREAMREADBEGIN)(int nEncNum);
    void (*PSTREAMREADPROC)(int nEncNum, LPVOID
    pStreamBuf, long lBufSize, DWORD dwCompFlags);
    void (*PSTREAMREADEND)(int nEncNum);
}STREAMREAD_STRUCT;
```

### Parameters:

| | |
|---|---|
| PSTREAMREADBEGIN: | The pointer to the Stream Read Begin callback function will be called when begins the video stream read process. |
| PSTREAMREADPROC: | The pointer to the Stream Read Process callback function will be called after every video frame is encoded. |
| PSTREAMREADEND: | The pointer to the Stream Read End callback function will be called when the video stream read process is finished. |

### Description

This structure stores the Stream Read callback function pointers.

### See Also

PSTREAMREADBEGIN
PSTREAMREADPROC
PSTREAMREADEND
AdvDVP_SetStreamReadCB

# Method

## AdvDVP_CreateEncSDKInstance

**Syntax**

int AdvDVP_CreateEncSDKInstance (void **pp)

**Parameters**

| | |
|---|---|
| [ OUT ] pp: | A pointer to the encoding SDK instance. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_PARAMERROR: | Parameter error. |

**Description**

This function creates the encoding SDK instance. You can delete the pointer which is carried by the parameter named “pp” to free this SDK instance before you free the DLL handle.

# AdvDVP_InitSDK

**Syntax**

int AdvDVP_InitSDK(void)

**Parameters**

None

**Return Value**

ENC_SUCCEEDED: Function succeeded.

**Description**

This function initializes all parameters of the SDK in the system.

**See Also**

AdvDVP_CloseSDK

# AdvDVP_CloseSDK

**Syntax**

int AdvDVP_CloseSDK(void)

**Parameters**

None

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |

**Description**

This function cleans all parameters of the SDK and closes up the SDK.

**See Also**

AdvDVP_InitSDK

# AdvDVP_InitEncoder

**Syntax**

int AdvDVP_InitEncoder(int nEncNum, int nEncBufSize)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ IN ] nEncBufSize: | Specifies the encoding buffer size. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |

**Description**

This function opens and initializes the specified video encoder. After initializing the encoder, the encoding state would be set as “ENC_STOPPED”.

**See Also**

AdvDVP_CloseEncoder

AdvDVP_GetState

# AdvDVP_CloseEncoder

**Syntax**

int AdvDVP_CloseEncoder(int nEncNum)

**Parameters**

[ IN ] nEncNum: Specifies the encoder number (0~15).

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function closes and releases the specified video encoder. After successfully calling this function, the encoding state would be set as “ENC_UNINITIALIZED”.

**See Also**

AdvDVP_InitEncoder
AdvDVP_GetState

# AdvDVP_StartVideoEncode

**Syntax**

int AdvDVP_StartVideoEncode(int nEncNum)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function notifies the specified video encoder to prepare to encode the video. The encode state would be set as “ENC_RUNNING” after a successful beginning.

**See Also**

AdvDVP_VideoEncode
AdvDVP_StopVideoEncode
AdvDVP_GetState

# AdvDVP_VideoEncode

**Syntax**

int AdvDVP_VideoEncode(int nEncNum, LPVOID lpInBuf,
int InBufSize, BOOL bKeyFrame)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ IN ] lpInBuf: | Specifies the input buffer that stores the source video frame. |
| [ IN ] InBufSize: | Specifies the size of the input buffer. |
| [ IN ] bKeyFrame: | Specifies if the video frame is encoded as a I-frame. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |
| ENC_PARAMERROR: | Parameter error. |
| ENC_BUFFERFULL: | Encoding buffer is full, the video frame can not be written to the buffer. |

**Description**

This function writes the video frame to the encoding buffer to encode it by the specified encoder.

**See Also**

AdvDVP_StartVideoEncode
AdvDVP_StopVideoEncode

# AdvDVP_StopVideoEncode

**Syntax**

int AdvDVP_StopVideoEncode(int nEncNum)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function notifies the specified video encoder to stop encoding and releases all relational resources. The encoding state would be set as “ENC_STOPPED” after a successful stop.

**See Also**

AdvDVP_StartVideoEncode
AdvDVP_VideoEncode
AdvDVP_GetState

# AdvDVP_GetState

**Syntax**

int AdvDVP_GetState(int nEncNum)

**Parameters**

[ IN ] nEncNum: Specifies the encoder number (0~15).

**Return Value**

ENC_ENCNUMERROR: Invalid encoder number.

**Description**

This function gets encoding state of a specified video encoder.

```
typedef enum
{
    ENC_STOPPED          = 1,
    ENC_RUNNING          = 2,
    ENC_UNINITIALIZED    = -1,
} EncoderState;
```

**See Also**

AdvDVP_InitEncoder
AdvDVP_CloseEncoder
AdvDVP_StartVideoEncode
AdvDVP_StopVideoEncode

# AdvDVP_SetStreamReadCB

**Syntax**

void AdvDVP_SetStreamReadCB(STREAMREAD_STRUCT *pStreamRead)

**Parameters**

[ IN ] pStreamRead: A pointer to STREAMREAD_STRUCT structure recording the pointers to the StreamRead callback functions.

**Return Value**

None

**Description**

This function registers the Stream Read callback functions to the SDK.

**See Also**

STREAMREAD_STRUCT structure

# AdvDVP_GetVideoQuant

**Syntax**

int AdvDVP_GetVideoQuant(int nEncNum, int *nQuant)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ OUT ] nQuant: | A pointer to get the video quant. The range is 1~31. The default video quality is 4. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |
| ENC_PARAMERROR: | The nQuant parameter is invalid. |

**Description**

This function gets video quant of the specified video encoder. The lower video quant can get the compressed video with higher quality and bit rate, vice versa.

**See Also**

AdvDVP_SetVideoQuant

# AdvDVP_SetVideoQuant

**Syntax**

int AdvDVP_SetVideoQuant(int nEncNum, int nQuant)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ IN ] nQuant: | A value to set the video quant. The range is 1~31. The default video quality is 4. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function sets video quant of the specified video encoder. The lower video quant can get the compressed video with higher quality and bit rate, vice versa.

**See Also**

AdvDVP_GetVideoQuant

# AdvDVP_GetVideoFrameRate

**Syntax**

int AdvDVP_GetVideoFrameRate(int nEncNum, int *nFrameRate)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ OUT ] nFrameRate: | A pointer to get the video frame rate. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |
| ENC_PARAMERROR: | The nFrameRate parameter is invalid. |

**Description**

This function gets video frame rate of the specified video encoder.

**See Also**

AdvDVP_SetVideoFrameRate

# AdvDVP_SetVideoFrameRate

**Syntax**

int AdvDVP_SetVideoFrameRate(int nEncNum, int nFrameRate)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ IN ] nFrameRate: | A value to set the video frame rate. The range is 1~30. The default video frame rate is 30. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function sets video frame rate of the specified video encoder. The set value of this function will come into effect only before AdvDVP_StartVideoEncoder is called.

Calling this function after AdvDVP_StartVideoEncoder will not take effect, but you can use AdvDVP_GetVideoFrameRate to get the latest value you have set.

**See Also**

AdvDVP_GetVideoFrameRate

# AdvDVP_GetVideoResolution

**Syntax**

int AdvDVP_GetVideoResolution(int nEncNum, int *nWidth, int *nHeight)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ OUT ] nWidth: | A pointer to get the width of the video. |
| [ OUT ] nHeight: | A pointer to get the height of the video. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |
| ENC_PARAMERROR: | Invalid pointer parameter. |

**Description**

This function gets video resolution of the specified video encoder.

**See Also**

AdvDVP_SetVideoResolution

# AdvDVP_SetVideoResolution

**Syntax**

int AdvDVP_SetVideoResolution(int nEncNum, int nWidth, int nHeight)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ IN ] nWidth: | A value to set the width of the video. The default width is 320. |
| [ IN ] nHeight | A value to set the height of the video. The default height is 240. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function sets video resolution of the specified video encoder. The set value of this function will come into effect only before AdvDVP_StartVideoEncoder is called.

Calling this function after AdvDVP_StartVideoEncoder will not take effect, but you can use AdvDVP_GetVideoResolution to get the latest value you have set.

**See Also**

AdvDVP_GetVideoResolution

# AdvDVP_GetVideoKeyInterval

**Syntax**

int AdvDVP_GetVideoKeyInterval(int nEncNum,
int *nKeyInterval)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ OUT ] nKeyInterval: | A pointer to get the interval of the video key frame. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |
| ENC_PARAMERROR: | The nKeyInterval parameter is invalid. |

**Description**

This function gets the interval of the video key frame of the specified video encoder.

**See Also**

AdvDVP_SetVideoKeyInterval

# AdvDVP_SetVideoKeyInterval

**Syntax**

int AdvDVP_SetVideoKeyInterval(int nEncNum, int nKeyInterval)

**Parameters**

| | |
|---|---|
| [ IN ] nEncNum: | Specifies the encoder number (0~15). |
| [ IN ] nKeyInterval: | A value to set the interval of the video key frame. The range is 1~300. The default interval of video key frame is 100. If the value is less than 1, it will use 1. If the value is greater than 300, it will use 300. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_ENCNUMERROR: | Invalid encoder number. |
| ENC_ENCINITFAILED: | Encoder is not initialized successfully. |

**Description**

This function sets the interval of the video key frame of the specified video encoder.

**See Also**

AdvDVP_GetVideoKeyInterval

# AdvDVP_CreateAVIFile

**Syntax**

HANDLE AdvDVP_CreateAVIFile(LPCSTR lpcsFileName, int nWidth, int nHeight, int nFrameRate)

**Parameters**

| | |
|---|---|
| [ IN ] lpcsFileName: | Specifies the file name of the AVI file. |
| [ IN ] nWidth: | The width of the image. |
| [ IN ] nHeight: | The height of the image. |
| [ IN ] nFrameRate | Specifies the frame rate of the video. |

**Return Value**

If the function succeeds, the file handle is returned. Otherwise, the function returns NULL.

**Description**

This function creates the AVI file to save the encoded video stream.

**See Also**

AdvDVP_WriteAVIFile
AdvDVP_CloseAVIFile

# AdvDVP_WriteAVIFile

**Syntax**

int AdvDVP_WriteAVIFile(HANDLE hAVIFile, LPVOID lpStreamBuf, long lBufSize, DWORD dwCompFlags)

**Parameters**

| | |
|---|---|
| [ IN ] hAVIFile: | Specifies the AVI file handle. |
| [ IN ] lpStreamBuf: | A pointer to the video stream data buffer written into the file. |
| [ IN ] lBufSize: | Specifies the size of the video stream data buffer. |
| [ IN ] dwCompFlags: | Flag associated with this data. The AVIIF_KEYFRAME flag is defined to indicate that this data does not rely on preceding data in the file. |

#define AVIIF_KEYFRAME 0x00000010L

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |
| ENC_PARAMERROR: | The lpStreamBuf is NULL. |

**Description**

This function writes the video stream data into the specified AVI file.

**See Also**

AdvDVP_CreateAVIFile
AdvDVP_CloseAVIFile

# AdvDVP_CloseAVIFile

**Syntax**

int AdvDVP_CloseAVIFile(HANDLE hAVIFile)

**Parameters**

| | |
|---|---|
| [ IN ] hAVIFile: | Specifies the AVI file handle. |

**Return Value**

| | |
|---|---|
| ENC_SUCCEEDED: | Function succeeded. |
| ENC_FAILED: | Function failed. |
| ENC_SDKINITFAILED: | SDK is not initialized successfully. |

**Description**

This function closes the specified AVI file.

**See Also**

AdvDVP_CreateAVIFile
AdvDVP_WriteAVIFile

# DVP7010B/7020B Player Functions Reference

## Data Type

## PlayerRes

**Syntax**

```
typedef enum tagRes
{
    PLAYER_SUCCEEDED            = 1,
    PLAYER_FAILED               = 0,
    PLAYER_SDKINITFAILED        = -1,
    PLAYER_PARAMERROR           = -2,
} PlayerRes;
```

**Description**

The function returns value.

# Method

## AdvDVP_CreatePlayerSDKInstance

**Syntax**

int AdvDVP_CreatePlayerSDKInstance(void **pp)

**Parameters**

[ OUT ] pp: A pointer to the player SDK instance.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |
| PLAYER_PARAMERROR: | Parameter error. |

**Description**

This function creates playback SDK instance. You can delete the pointer which is carried by the parameter named “pp” to free this SDK instance before you free the DLL handle.

# AdvDVP_OpenFile

**Syntax**

int AdvDVP_OpenFile(LPCSTR lpcsFileName)

**Parameters**

| | |
|---|---|
| [ IN ] lpcsFileName: | Specifies the file name of the source video file. |

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function opens the source video file and initializes the video player. The playback status would be set as “PLAYER_STOPPED” after successfully calling this function.

**See Also**

AdvDVP_CloseFile
AdvDVP_GetStatus

# AdvDVP_CloseFile

**Syntax**

int AdvDVP_CloseFile()

**Parameters**

None.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function closes the source video file and free resources allocated for video player. The playback status would be set as “PLAYER_NOTOPENED” after successfully calling this function.

**See Also**

AdvDVP_OpenFile
AdvDVP_GetStatus

# AdvDVP_Play

**Syntax**

int AdvDVP_Play(HWND hwndApp, BOOL bAutoResizeWnd)

**Parameters**

| | |
|---|---|
| [ IN ] hwndApp: | A windows handle for display area. |
| [ IN ] bAutoResizeWnd: | Specifies if the display area is resized automatically according to the video resolution. |

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function plays the file that has been opened. The playback status would be set as “PLAYER_PLAYING” after successfully calling this function.

**See Also**

AdvDVP_Pause
AdvDVP_Stop
AdvDVP_GetStatus

# AdvDVP_Pause

**Syntax**

int AdvDVP_Pause()

**Parameters**

None.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function pauses or continues the file that has been opened. The playback status would be set as “PLAYER_PAUSED” after successfully calling this function.

**See Also**

AdvDVP_Play
AdvDVP_Stop
AdvDVP_GetStatus

# AdvDVP_Stop

**Syntax**

int AdvDVP_Stop()

**Parameters**

None.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function stops the file that is playing. The playback status would be set as “PLAYER_STOPPED” after successfully calling this function.

**See Also**

AdvDVP_Play
AdvDVP_Pause
AdvDVP_GetStatus

# AdvDVP_Fast

**Syntax**

int AdvDVP_Fast()

**Parameters**

None.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function improves the current play speed by one time, 4 times at most. The playback status would be set as “PLAYER_PLAYING” after successfully calling this function.

**See Also**

AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Slow
AdvDVP_GetStatus

# AdvDVP_Slow

**Syntax**

int AdvDVP_Slow()

**Parameters**

None.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function slows the current play speed by one time, 4 times at most. The playback status would be set as “PLAYER_PLAYING” after successfully calling this function.

**See Also**

AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Fast
AdvDVP_GetStatus

# AdvDVP_PlayStep

**Syntax**

int AdvDVP_PlayStep()

**Parameters**

None.

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function makes the video to step forward one frame. The playback status would be set as “PLAYER_PAUSED” after successfully calling this function. This function should sleep for a little time, for example, 100 milliseconds before calling AdvDVP_GetStatus.

**See Also**

AdvDVP_Pause
AdvDVP_Stop
AdvDVP_GetStatus

# AdvDVP_GetStatus

**Syntax**

int AdvDVP_GetStatus ()

**Parameters**

None

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function gets playback status.

```
typedef enum tagPlayerStatus{
    PLAYER_NOTOPENED = 0,
    PLAYER_OPENED    = 1,
    PLAYER_PLAYING   = 2,
    PLAYER_STOPPED   = 3,
    PLAYER_PAUSED    = 4
} PlayerStatus;
```

**See Also**

AdvDVP_OpenFile
AdvDVP_CloseFile
AdvDVP_Play
AdvDVP_Pause
AdvDVP_Stop
AdvDVP_Fast
AdvDVP_Slow
AdvDVP_PlayStep

# AdvDVP_GetCurImage

**Syntax**

int AdvDVP_GetCurImage(LPBYTE *lpImage,
long *pBufSize)

**Parameters**

| | |
|---|---|
| [ OUT ] lpImage: | A pointer to a image buffer. |
| [ IN, OUT ] pBufSize: | A long pointer to receive the returned image buffer size. |

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |
| PLAYER_PARAMERROR: | Parameter error. |

**Description**

This function gets current played image.

**See Also**

# AdvDVP_RegNotifyMsg

**Syntax**

int AdvDVP_RegNotifyMsg(HWND hWnd, UINT nMsg)

**Parameters**

| | |
|---|---|
| [ IN ] hWnd: | Specifies the handle of the window receiving this message. |
| [ IN ] nMsg: | Specifies the user-defined message. When this message is received, it means some event of the playback occurs, such as the file playing is finished. |

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function registers a user-defined message. When an event of the playback occurs, this message will be sent to the specified window.

This function must be called after “AdvDVP_OpenFile” function.

**See Also**

AdvDVP_CheckFileEnd

# AdvDVP_CheckFileEnd

**Syntax**

BOOL AdvDVP_CheckFileEnd ()

**Parameters**

None

**Return Value**

If the event that the file playing end is detected, this function returns TRUE. Otherwise, the function returns FALSE.

**Description**

This function checks if the file playing is finished.

**See Also**

AdvDVP_RegNotifyMsg

# AdvDVP_GetVideoResolution

**Syntax**

int AdvDVP_GetVideoResolution(int *nWidth, int *nHeight)

**Parameters**

| | |
|---|---|
| [ OUT ] nWidth: | An integer pointer to get the width of the video. |
| [ OUT ] nHeight: | An integer pointer to get the height of the video. |

**Return Value**

| | |
|---|---|
| PLAYER_SUCCEEDED: | Function succeeded. |
| PLAYER_FAILED: | Function failed. |

**Description**

This function gets the width and height of the video.

**See Also**

# AdvDVP_GetPlayRate

**Syntax**

double AdvDVP_GetPlayRate()

**Parameters**

None

**Return Value**

If the function succeeded, the playback ratio is returned. Otherwise, the function returns 0.

**Description**

This function retrieves the playback rate.

**See Also**

AdvDVP_Play
AdvDVP_Fast
AdvDVP_Slow

# AdvDVP_GetFileTime

**Syntax**

double AdvDVP_GetFileTime()

**Parameters**

None

**Return Value**

If the function succeeded, the total file time is returned. Otherwise, the function returns 0.

**Description**

This function retrieves total file time in seconds.

**See Also**

AdvDVP_GetPlayedTime
AdvDVP_SetPlayPosition

# AdvDVP_GetPlayedTime

**Syntax**

double AdvDVP_GetPlayedTime()

**Parameters**

None

**Return Value**

If the function succeeded, the current file time is returned. Otherwise, the function returns 0.

**Description**

This function retrieves current file time in seconds.

**See Also**

AdvDVP_GetFileTime
AdvDVP_SetPlayPosition

# AdvDVP_SetPlayPosition

**Syntax**

int AdvDVP_SetPlayPosition (double dTime)

**Parameters**

[ IN ] dTime: Specifies the file time in seconds.

**Return Value**

PLAYER_SUCCEEDED: Function succeeded.
PLAYER_FAILED: Function failed.

**Description**

This function adjusts the file time to the specified file time. If you want to call AdvDVP_CheckFileEnd after this function, this function should sleep at least 500 milliseconds before AdvDVP_CheckFileEnd.

**See Also**

AdvDVP_GetFileTime
AdvDVP_GetPlayedTime

# AdvDVP_GetFileTotalFrames

**Syntax**

LONGLONG AdvDVP_GetFileTotalFrames()

**Parameters**

None

**Return Value**

If the function succeeded, the total number of the frames in the file is returned. Otherwise, the function returns 0.

**Description**

This function retrieves total number of the frames in the file.

**See Also**

AdvDVP_GetPlayedFrames

# AdvDVP_GetPlayedFrames

**Syntax**

LONGLONG AdvDVP_GetPlayedFrames()

**Parameters**

None

**Return Value**

If the function succeeded, the current frame number of the file is returned. Otherwise, the function returns 0.

**Description**

This function retrieves current frame number of the file.

**See Also**

AdvDVP_GetFileTotalFrames