CHAPTER 1

Video Capture Application Reference

This section is an alphabetic reference to the functions and data structures provided by Video for Windows for use by video capture applications. There are separate sections for functions, messages, and data structures. The messages and data structures are defined in MSVIDEO.H.

Extensions are being added to the video capture functions to make it easier for applications to access video capture drivers. If your application needs video capture services, it should use the extensions rather than these functions. If you are developing video capture device drivers, you might use these functions for testing your drivers.

If you need information on the video capture extensions to develop your application, you can request the latest information from the following group:

Microsoft Corporation
Multimedia Systems Group
Product Marketing
One Microsoft Way
Redmond, WA 98052-6399

FAX: (206) 93MSFAX

Video Capture Function Reference

This section contains a listing of the functions used by video capture applications. The function definition is given, followed by a description of each parameter.

Video Capture Function Summary

The following function operates on a single frame:

videoFrame

This function transfers a single frame from or to a video device channel.

The following functions are used to open, close, and communicate with a video capture device:

videoClose

This function closes the specified video device channel.

videoGetErrorText

This function retrieves a description of the error identified by the error number.

videoMessage

This function sends messages to a video device channel.

videoOpen

This function opens a channel on the specified video device.

The following functions control the configuration of a video capture device:

videoConfigure

This function sets or retrieves a configurable driver option.

videoConfigureStorage

This function saves or loads all configurable options for a channel.

videoDialog

This function displays a channel specific dialog box used to set configuration parameters.

videoGetChannelCaps

This function retrieves a description of the capabilities of a channel.

videoGetNumDevs

This function returns the number of MSVIDEO devices installed.

videoUpdate

This function directs a channel to repaint the display.

The following functions control video capture streaming:

videoStreamAddBuffer

This function sends a buffer to a video device channel.

videoStreamFini

This function terminates streaming from the specified device channel.

videoStreamGetError

This function returns the most recent error encountered.

videoStreamGetPosition

This function retrieves the current position of the specified video device channel.

videoStreamInit

This function initializes a video device channel for streaming.

videoStreamPrepareHeader

This function prepares a buffer for video streaming.

videoStreamReset

This function stops streaming on the specified video device channel, returns all video buffers from the driver, and resets the current position to zero.

videoStreamStart

This function starts streaming on the specified video device channel.

videoStreamStop

This function stops streaming on a video channel.

videoStreamUnprepareHeader

This function cleans up the preparation performed by **videoStreamPrepareHeader**.

Video Capture Function Alphabetic Reference

videoClose

Syntax DWORD **videoClose**(*hVideo*)

This function closes the specified video device channel.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel. If the function is successful, the handle will be invalid after this call.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The

following errors are defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

DV_ERR_NONSPECIFIC

The driver failed to close the channel.

Comments If buffers have been sent with **videoStreamAddBuffer** and they haven't been returned to the application, the close operation fails. You can use **videoStreamReset** to mark all pending buffers as done.

See Also videoOpen, videoStreamInit, videoStreamFini, videoStreamReset

---

videoConfigure

Syntax DWORD **videoConfigure**(*hVideo*, *msg*, *dwFlags*, *lpdwReturn*, *lpData1*, *dwSize1*, *lpData2*,

This function sets or retrieves a configurable driver option.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

UINT *msg*

Specifies the option to set or retrieve.

DVM_PALETTE
Indicates a palette is being sent to the driver or retrieved from the driver.

DVM_PALETTERGB555
Indicates an RGB555 palette is being sent to the driver.

DVM_FORMAT
Indicates format information is being sent to the driver or retrieved from the driver.

DWORD *dwFlags*

Specifies flags for configuring or interrogating the device driver. The following flags are defined:

VIDEO_CONFIGURE_SET
Indicates values are being sent to the driver.

VIDEO_CONFIGURE_GET
Indicates values are being obtainded from the driver.

VIDEO_CONFIGURE_QUERY
This flag is used to determine if the driver supports the option specified

by *msg*. This flag should be combined with either the VIDEO_CONFIGURE_SET or VIDEO_CONFIGURE_GET flag. If this flag is set, the *lpData1*, *dwSize1*, *lpData2*, and *dwSize2* parameters are ignored.

VIDEO_CONFIGURE_QUERYSIZE
Returns the size, in bytes, of the configuration option in *lpdwReturn*. This flag is only valid if the VIDEO_CONFIGURE_GET flag is also set.

VIDEO_CONFIGURE_CURRENT
Requests the current value. This flag is only valid if the VIDEO_CONFIGURE_GET flag is also set.

VIDEO_CONFIGURE_NOMINAL
Requests the nominal value. This flag is only valid if the VIDEO_CONFIGURE_GET flag is also set.

VIDEO_CONFIGURE_MIN
Requests the minimum value. This flag is only valid if the VIDEO_CONFIGURE_GET flag is also set.

VIDEO_CONFIGURE_MAX
Get the maximum value. This flag is only valid if the VIDEO_CONFIGURE_GET flag is also set.

LPDWORD lpdwReturn

Points to a DWORD used for returning information from the driver. If the VIDEO_CONFIGURE_QUERYSIZE flag is set, *lpdwReturn* is filled with the size of the configuration option.

LPVOID *lpData1*

Specifies a pointer to message specific data.

DWORD *dwSize1*

Specifies the size of the *lpData1* buffer in bytes.

LPVOID *lpData2*

Specifies a pointer to message specific data.

DWORD *dwSize2*

Size of the *lpData2* buffer in bytes.

Return Value

Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

DV_ERR_NOTSUPPORTED

Function is not supported.

See Also videoOpen, videoMessage

---

videoConfigureStorage

Syntax DWORD **videoConfigureStorage**(*hVideo*, *lpstrIdent*, *dwFlags*)

This function saves or loads all configurable options for a channel. Options can be saved and recalled for each application or each application instance.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPSTR lpstrIdent

Identifies the application or instance. Use an arbitrary string which uniquely identifies your application or instance.

DWORD *dwFlags*

Specifies flags for the storage. The following flags are defined:

VIDEO_CONFIGURE_GET
Requests that the values be loaded.

VIDEO_CONFIGURE_SET
Requests that the values be saved.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

DV_ERR_NOTSUPPORTED

Function is not supported.

Comments The method used by a driver to save configuration options is device dependent.

See Also videoOpen

---

videoDialog

Syntax DWORD **videoDialog**(*hVideo*, *hWndParent*, *dwFlags*)

This function displays a channel-specific dialog box used to set configuration parameters.

Parameters

HVIDEO *hVideo*

Specifies a handle to the video device channel.

HWND hWndParent

Specifies the parent window handle.

DWORD *dwFlags*

Specifies flags for the dialog box. The following flag is defined:

VIDEO_DLG_QUERY

If this flag is set, the driver immediately returns zero if it supplies a dialog box for the channel, or DV_ERR_NOTSUPPORTED if it does not.

Return Value

Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

DV_ERR_NOTSUPPORTED

Function is not supported.

Comments

Typically, each dialog box displayed by this function lets the user select options appropriate for the channel. For example, a VIDEO_IN channel dialog box lets the user select the image dimensions and bit depth.

See Also

videoOpen, videoConfigureStorage

---

videoFrame

Syntax

DWORD **videoFrame**(*hVideo*, *lpVHdr*)

This function transfers a single frame from or to a video device channel.

Parameters

HVIDEO *hVideo*

Specifies a handle to the video device channel. The channel must be of type VIDEO_IN or VIDEO_OUT.

LPVIDEOHDR *lpVHdr*

Specifies a far pointer to an **VIDEOHDR** structure.

Return Value

Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

Comments Use this function with a VIDEO_IN channel to transfer a single image from the frame buffer. Use this function with a VIDEO_OUT channel to transfer a single image to the frame buffer.

To determine the size of the buffer needed for capturing data, use **videoConfigure**. Set the size of the buffer in the **dwBufferLength** field of the **VIDEOHDR** structure. If the buffer size is not large enough, the function can fail.

See Also videoOpen

videoGetChannelCaps

Syntax DWORD **videoGetChannelCaps**(*hVideo*, *lpChannelCaps*, *dwSize*)

This function retrieves a description of the capabilities of a channel.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPCHANNEL_CAPS *lpChannelCaps*

Specifies a far pointer to a **CHANNEL_CAPS** structure.

DWORD *dwSize*

Specifies the size of the **CHANNEL_CAPS** structure.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_UNSUPPORTED

Function is not supported.

Comments The channel capabilities structure returns the capability information. For example, capability information might include whether or not the channel can crop and scale images.

videoGetErrorText

Syntax DWORD **videoGetErrorText**(*hVideo*, *wError*, *lpText*, *wSize*)

This function retrieves a description of the error identified by the error number.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel. This might be NULL if the error is not device specific.

UINT *wError*

Specifies the error number.

LPSTR *lpText*

Specifies a far pointer to a buffer which is filled with a null-terminated string corresponding to the error number.

UINT *wSize*

Specifies the length of the buffer pointed to by *lpText*.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_BADERRNUM

Specified error number is out of range.

Comments If the error description is longer than the buffer, the description is truncated. The returned error string is always null-terminated. If *wSize* is zero, nothing is copied and the function returns zero.

videoGetNumDevs

Syntax DWORD **videoGetNumDevs**()

This function returns the number of MSVIDEO devices installed.

Parameters None

Return Value Returns the number of MSVIDEO devices listed in the [drivers] section of the SYSTEM.INI file.

See Also videoOpen

videoMessage

Syntax DWORD **videoMessage**(*hVideo*, *wMsg*, *dwP1*, *dwP2*)

This function sends a message to a video device channel.

Parameters HVIDEO *hVideo*

Specifies the handle to the device.

UINT *wMsg*

Specifies the message to send.

DWORD *dwP1*

Specifies the first parameter for the message.

DWORD *dwP2*

Specifies the second parameter for the message.

Return Value Returns the message specific value returned from the driver.

Comments This function is used for configuration messages such as **DVM_SRC_RECT** and **DVM_DST_RECT** and device specific messages.

See Also videoConfigure

---

videoOpen

Syntax DWORD **videoOpen**(*lphvideo*, *dwDeviceID*, *dwFlags*)

This function opens a channel on the specified video device.

Parameters LPHVIDEO *lphvideo*

Specifies a far pointer to a **HVIDEO** handle. The video capture driver uses this location to return a handle that uniquely identifies the opened video device channel. Use this handle to identify the device channel when calling other video functions.

DWORD dwDeviceID

Identifies the video device to open. The value of *dwDeviceID* varies from zero to one less than the number of video capture devices installed in the system.

DWORD *dwFlags*

Specifies flags for opening the device. The following flags are defined:

VIDEO_EXTERNALIN
: Specifies the channel is opened for external input. Typically, external input channels capture images into a frame buffer.

VIDEO_EXTERNALOUT
: Specifies the channel is opened for external output. Typically, external output channels display images stored in a frame buffer on an auxiliary monitor or overlay.

VIDEO_IN
: Specifies the channel is opened for video input. Video input channels transfer images from a frame buffer to system memory buffers.

VIDEO_OUT

Specifies the channel is opened for video output. Video output channels transfer images from system memory buffers to a frame buffer.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_BADDEVICEID

Specified device ID is out of range.

DV_ERR_ALLOCATED

Specified resource is already allocated.

DV_ERR_NOMEM

Unable to allocate or lock memory.

Comments At a minimum, all capture drivers support a **VIDEO_EXTERNALIN** and a **VIDEO_IN** channel. Use **videoGetNumDevs** to determine the number of video devices present in the system.

See Also videoClose, videoGetNumDevs

---

videoStreamAddBuffer

Syntax DWORD **videoStreamAddBuffer**(*hVideo*, *lpvideoHdr*, *dwSize*)

This function sends a buffer to a video device channel. After the buffer is filled by the device, the device sends it back to the application.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPVIDEOHDR *lpvideoHdr*

Specifies a far pointer to a **VIDEOHDR** structure that identifies the buffer.

DWORD *dwSize*

Specifies the size of the **VIDEOHDR** structure.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

The device handle specified is invalid.

DV_ERR_UNPREPARED

The *lpvideoHdr* structure hasn't been prepared.

Comments The data buffer must be prepared with **videoStreamPrepareHeader** before it is passed to

videoStreamAddBuffer. The **VIDEOHDR** data structure and the data buffer pointed to by its **lpData** field must be allocated with **GlobalAlloc** using the GMEM_MOVEABLE and GMEM_SHARE flags, and locked with **GlobalLock**.

To determine the size of the buffer needed for capturing data, use **videoConfigure**. Set the size of the buffer in the **dwBufferLength** field of the **VIDEOHDR** structure. If the buffer size is not large enough, the driver might return DV_ERR_NONSPECIFIC.

See Also videoStreamPrepareHeader

---

videoStreamFini

Syntax DWORD **videoStreamFini**(*hVideo*)

This function terminates streaming from the specified device channel.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

The device handle specified is invalid.

DV_ERR_STILLPLAYING

There are still buffers in the queue.

Comments If there are buffers that have been sent with **videoStreamAddBuffer** that haven't been returned to the application, this operation will fail. Use **videoStreamReset** to mark all pending buffers as done.

Each call to **videoStreamInit** must be matched with a call to **videoStreamFini**.

For **VIDEO_EXTERNALIN** channels, this function is used to halt capturing of data to the frame buffer.

For **VIDEO_EXTERNALOUT** channels that support overlay, this function is used to disable the overlay.

See Also videoStreamInit

---

videoStreamGetError

Syntax DWORD **videoStreamGetError**(*hVideo, lpdwErrorID, lpdwErrorValue*)

This function returns the error most recently encountered.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPDWORD lpdwErrorID

Specifies a far pointer to the DWORD to be filled with the error ID.

LPDWORD lpdwErrorValue

Specifies a far pointer to the DWORD to be filled with the number of frames skipped.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_INVALHANDLE

The device handle specified is invalid.

Comments While streaming video data, a capture driver can fill buffers faster than the client application can save the buffers to disk. In this case, the **DV_ERR_NO_BUFFERS** error is returned in *lpdwErrorID* and *lpdwErrorValue* contains a count of the number of frames missed. After receiving this message and returning the error status, the driver should reset its internal error flag to **DV_ERR_OK** and the count of missed frames to zero.

Applications should send this message frequently during capture since some drivers which do not have access to interrupts use this message to trigger buffer processing.

See Also videoOpen

---

videoStreamGetPosition

Syntax DWORD **videoStreamGetPosition**(*hVideo*, *lpInfo*, *dwSize*)

This function retrieves the current position of the specified video device channel.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPMMTIME *lpInfo*

Specifies a far pointer to an **MMTIME** structure.

DWORD *dwSize*

Specifies the size of the **MMTIME** structure.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

Comments Before using **videoStreamGetPosition**, set the **wType** field of the **MMTIME** structure to indicate the time format you desire. After **videoStreamGetPosition** returns, check the **wType** field to determine if the your time format is supported. If not, **wType** will specify an alternate format. Video capture drivers typically provide the milliseconds time format.

The position is set to zero when streaming is started with **videoStreamStart**.

videoStreamInit

Syntax DWORD **videoStreamInit**(*hVideo*, *dwMicroSecPerFrame*, *dwCallback*, *dwCallbackInstance*, *dwFlags*)

This function initializes a video device channel for streaming.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

DWORD dwMicroSecPerFrame

Specifies the number of microseconds between frames.

DWORD dwCallback

Specifies the address of a callback function or a handle to a window called during video streaming. The callback function or window processes messages related to the progress of streaming.

DWORD dwCallbackInstance

Specifies user instance data passed to the callback function. This parameter is not used with window callbacks.

DWORD *dwFlags*

Specifies flags for opening the device channel. The following flags are defined:

CALLBACK_WINDOW
If this flag is specified, *dwCallback* is a window handle.

CALLBACK_FUNCTION
If this flag is specified, *dwCallback* is a callback procedure address.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_BADDEVICEID

The device ID specified in *hVideo* is not valid.

DV_ERR_ALLOCATED

The resource specified is already allocated.

DV_ERR_NOMEM

Unable to allocate or lock memory.

Comments

If a window is chosen to receive callback information, the following messages are sent to the window procedure function to indicate the progress of video input: **MM_DRVM_OPEN** at the time of videoStreamInit, **MM_DRVM_CLOSE** at the time of videoStreamFini, **MM_DRVM_DATA** when a buffer of image data is available, **MM_DRVM_ERROR** when an error occurs.

If a function is chosen to receive callback information, the following messages are sent to the function to indicate the progress of video input: **MM_DRVM_OPEN**, **MM_DRVM_CLOSE**, **MM_DRVM_DATA**, **MM_DRVM_ERROR**. The callback function must reside in a DLL. You do not have to use **MakeProcInstance** to get a procedure-instance address for the callback function.

Callback

void CALLBACK **videoFunc**(*hVideo*, *wMsg*, *dwInstance*, *dwParam1*, *dwParam2*)

videoFunc is a placeholder for the application-supplied function name. The actual name must be exported by including it in an EXPORTS statement in the DLL's module-definition file.

Callback Parameters

HVIDEO *hVideo*

Specifies a handle to the video device channel associated with the callback.

DWORD *wMsg*

Specifies the **MM_DRVM_** message.

DWORD *dwInstance*

Specifies the user instance data specified with **videoOpen**.

DWORD *dwParam1*

Specifies a parameter for the message.

DWORD *dwParam2*

Specifies a parameter for the message.

Callback Comments

Because the callback is accessed at interrupt time, it must reside in a DLL and its code segment must be specified as FIXED in the module-definition file for the DLL. Any data that the callback accesses must be in a FIXED data segment as well. The callback may not make any system calls except for **PostMessage**, **timeGetSystemTime**, **timeGetTime**, **timeSetEvent**, **timeKillEvent**, **midiOutShortMsg**, **midiOutLongMsg**, and **OutputDebugStr**.

For **VIDEO_EXTERNALIN** channels, this function is used to initiate capturing of data

to the frame buffer.

For **VIDEO_EXTERNALOUT** channels which support overlay, this function is used to enable the overlay.

See Also videoOpen, videoStreamFini, videoClose

---

videoStreamPrepareHeader

Syntax DWORD **videoStreamPrepareHeader**(*hVideo*, *lpvideoHdr*, *dwSize*)

This function prepares a buffer for video streaming.

Parameters

HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPVIDEOHDR *lpvideoHdr*

Specifies a pointer to a **VIDEOHDR** structure that identifies the buffer to be prepared.

DWORD *dwSize*

Specifies the size of the **VIDEOHDR** structure.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

DV_ERR_NOMEM

Unable to allocate or lock memory.

Comments Use this function after **videoStreamInit** or after **videoStreamReset** to prepare the data buffers for streaming data.

The **VIDEOHDR** data structure and the data block pointed to by its **lpData** field must be allocated with **GlobalAlloc** using the GMEM_MOVEABLE and GMEM_SHARE flags, and locked with **GlobalLock**. Preparing a header that has already been prepared will have no effect and the function will return zero. Typically, this function is used to insure that the buffer will be available for use at interrupt time.

See Also videoStreamUnprepareHeader

---

videoStreamReset

Syntax DWORD **videoStreamReset**(*hVideo*)

This function stops streaming on the specified video device channel and resets the current position to zero. All pending buffers are marked as done and are returned to the application.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_INVALHANDLE

The device handle specified is invalid.

See Also videoStreamReset, videoStreamStop, videoStreamAddBuffer, videoStreamClose

---

videoStreamStart

Syntax DWORD **videoStreamStart**(*hVideo*)

This function starts streaming on the specified video device channel.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_INVALHANDLE

The device handle specified is invalid.

See Also videoStreamReset, videoStreamStop, videoStreamAddBuffer, videoStreamClose

---

videoStreamStop

Syntax DWORD **videoStreamStop**(*hVideo*)

This function stops streaming on a video channel.

Parameters HVIDEO *hVideo*

Specifies a handle to the video device channel.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_INVALHANDLE

Specified device handle is invalid.

Comments If there are any buffers in the queue, the current buffer will be marked as done (the **dwBytesRecorded** field in the **VIDEOHDR** header will contain the actual length of data), but any empty buffers in the queue will remain there. Calling this function when the channel is not started has no effect, and the function returns zero.

See Also videoStreamStart, videoStreamReset

---

videoStreamUnprepareHeader

Syntax DWORD **videoStreamUnprepareHeader**(*hVideo, lpvideoHdr, dwSize*)

This function cleans up the preparation performed by **videoStreamPrepareHeader**.

Parameters

HVIDEO *hVideo*

Specifies a handle to the video device channel.

LPVIDEOHDR *lpvideoHdr*

Specifies a pointer to a **VIDEOHDR** structure identifying the data buffer to be cleaned up.

DWORD *dwSize*

Specifies the size of the **VIDEOHDR** structure.

Return Value Returns zero if the function was successful. Otherwise, it returns an error number. The following errors are defined:

DV_ERR_INVALHANDLE

The device handle specified is invalid.

DV_ERR_STILLPLAYING

The structure identified by *lpvideoHdr* is still in the queue.

Comments This function is the complementary function to **videoStreamPrepareHeader**. You must call this function before freeing the data buffer with **GlobalFree**. After passing a buffer to the device driver with **videoStreamAddBuffer**, you must wait until the driver is finished with the buffer before calling **videoStreamUnprepareHeader**. Unpreparing a buffer that has not been prepared has no effect, and the function returns zero.

See Also videoStreamPrepareHeader

---

videoUpdate

Syntax DWORD **videoUpdate**(*hVideo, hWnd, hDC*)

This function directs a channel to repaint the display. It applies only to VIDEO_EXTERNALOUT channels.

Parameters

HVIDEO *hVideo*

Specifies a handle to the video device channel.

HWND *hWnd*

Specifies the handle of the window to be used by the channel for image display.

HDC *hDC*

Specifies a handle to a device context.

Return Value

Returns zero if the function was successful. Otherwise, it returns an error number. The following error is defined:

DV_ERR_UNSUPPORTED

Specified message is unsupported.

Comments

This message is normally sent whenever the client window receives a **WM_MOVE**, **WM_SIZE**, or **WM_PAINT** message.

Video Capture Data Structure Reference

This section lists data structures used by video capture applications. The data structures are presented in alphabetical order. The structure definition is given, followed by a description of each field.

Video Capture Data Structure Alphabetic Reference

CHANNEL_CAPS

The **CHANNEL_CAPS** structure is used with **videoGetChannelCaps** to return the capabilities of a channel to an application.

```
typedef struct channel_caps_tag {
   DWORD  dwFlags;
   DWORD  dwSrcRectXMod;
   DWORD  dwSrcRectYMod;
   DWORD  dwSrcRectWidthMod;
   DWORD  dwSrcRectHeightMod;
   DWORD  dwDstRectXMod;
   DWORD  dwDstRectYMod;
   DWORD  dwDstRectWidthMod;
   DWORD  dwDstRectHeightMod;
} CHANNEL_CAPS;
```

Fields

The **CHANNEL_CAPS** structure has the following fields:

dwFlags

Returns flags giving information about the channel. The following flags are defined:

VCAPS_OVERLAY
: Indicates the channel is capable of overlay. This flag is used only for VIDEO_EXTERNALOUT channels.

VCAPS_SRC_CAN_CLIP
: Indicates that the source rectangle can be set smaller than the maximum dimensions.

VCAPS_DST_CAN_CLIP
: Indicates that the destination rectangle can be set smaller than the maximum dimensions.

VCAPS_CAN_SCALE
: Indicates that the source rectangle can be a different size than the destination rectangle.

dwSrcRectXMod

Returns the granularity allowed when positioning the source rectangle in the horizontal direction.

dwSrcRectYMod

Returns the granularity allowed when positioning the source rectangle in the vertical direction.

dwSrcRectWidthMod

Returns the granularity allowed when setting the width of the source rectangle.

dwSrcRectHeightMod

Returns the granularity allowed when setting the height of the source rectangle.

dwDstRectXMod

Returns the granularity allowed when positioning the destination rectangle in the horizontal direction.

dwDstRectYMod

Returns the granularity allowed when positioning the destination rectangle in the vertical direction.

dwDstRectWidthMod

Returns the granularity allowed when setting the width of the destination rectangle.

dwDstRectHeightMod

Returns the granularity allowed when setting the height of the source rectangle.

Comments

Some channels can only use source and destination rectangles which fall on 2, 4, or 8 pixel boundaries. Similarly, some channels only accept capture rectangles widths and heights that are multiples of a fixed value. Rectangle dimensions indicated by modulus operators are considered advisory. When requesting a particular rectangle, the application must always check the return value to insure the request was accepted by the driver. For example, if **dwDstRectWidthMod** is set to 64, the application might try to set destination rectangles with widths of 64, 128, 192, 256, ..., and 640 pixels. The driver might actually support a subset of these sizes and indicates the supported sizes with the return value of the **DVM_DST_RECT** message. If a channel supports arbitrarily positioned rectangles, with arbitrary sizes, the values above should all be set to 1.

VIDEOHDR

The **VIDEOHDR** structure defines the header used to identify a video data buffer.

```
typedef struct videohdr_tag {
   LPSTR  lpData;
   DWORD  dwBufferLength;
   DWORD  dwBytesUsed;
   DWORD  dwTimeCaptured;
   DWORD  dwUser;
   DWORD  dwFlags;
   DWORD  dwReserved[4];
} VIDEOHDR;
```

The **VIDEOHDR** structure has the following fields:

lpData

Specifies a far pointer to the video data buffer.

dwBufferLength

Specifies the length of the data buffer.

dwBytesUsed

Specifies the number of bytes used in the data buffer.

dwTimeCaptured

Specifies the time (in milliseconds) when the frame was captured relative to the first frame in the stream.

dwUser

Specifies 32 bits of user data.

dwFlags

Specifies flags giving information about the data buffer. The following flags are defined for this field:

VHDR_DONE
: Set by the device driver to indicate it is finished with the data buffer and it is returning the buffer to the application.

VHDR_PREPARED
: Set by Windows to indicate the data buffer has been prepared with **videoStreamPrepareHeader**.

VHDR_INQUEUE
: Set by Windows to indicate the data buffer is queued for playback.

VHDR_KEYFRAME
: Set by the device driver to indicate a key frame.

dwReserved[4]

Reserved for use by the device driver. Typically, these maintain a linked list of buffers in the queue.