Video Components Reference for QuickTime

QuickTime > Media Types & Media Handlers

2006-05-23

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Video Components Reference for QuickTime

Framework:	Frameworks/QuickTime.framework
Declared in	IOMacOSTypes.h IOMacOSVideo.h QuickTimeComponents.h QuickdrawTypes.h

Overview

Video digitizer components convert video input into digitized color images that are compatible with the graphics system of a computer.

Functions by Task

Controlling Analog Video

`VDGetBlackLevelValue` (page 32)
Returns the current black level value.

`VDGetBrightness` (page 33)
Returns the current brightness value.

`VDGetContrast` (page 36)
Returns the current contrast value.

`VDGetHue` (page 42)
Returns the current hue value.

`VDGetInputGammaValue` (page 47)
Returns the current gamma values.

`VDGetSaturation` (page 56)
Returns the current saturation value.

`VDGetSharpness` (page 57)
Returns the current sharpness value.

`VDGetVideoDefaults` (page 61)
Returns the recommended values for many of the analog video parameters that may be set by applications.

`VDGetWhiteLevelValue` (page 62)
Returns the current white level value.

VDSetBlackLevelValue (page 73)
Sets the current black level value.

VDSetBrightness (page 74)
Sets the current brightness value.

VDSetContrast (page 77)
Sets the current contrast value.

VDSetHue (page 82)
Sets the current hue value.

VDSetInputGammaValue (page 84)
Sets the gamma values.

VDSetSaturation (page 92)
Sets the saturation value.

VDSetSharpness (page 92)
Sets the sharpness value.

VDSetWhiteLevelValue (page 94)
Sets the white level value.

Controlling Color

VDGetCLUTInUse (page 34)
Obtains the color lookup table used by a video digitizer component.

VDGetDMADepths (page 41)
Determines which pixel depths a digitizer supports.

VDGetInputColorSpaceMode (page 44)
Determines whether a digitizer is operating in color or grayscale mode.

VDSetInputColorSpaceMode (page 83)
Chooses between color and grayscale digitized video.

VDUseThisCLUT (page 95)
Specifies the lookup table for color digitization.

Controlling Compressed Source Devices

VDCompressDone (page 29)
Determines whether the video digitizer has finished digitizing and compressing a frame of image data.

VDCompressOneFrameAsync (page 30)
Instructs the video digitizer to digitize and compress a single frame of image data.

VDGetCompressionTime (page 34)
Confirms or quantifies a video digitizer's compression settings.

VDGetCompressionTypes (page 35)
Determines the image-compression capabilities of the video digitizer.

VDGetImageDescription (page 43)
Retrieves an ImageDescription structure from a video digitizer.

`VDGetSoundInputSource` (page 58)
Instructs your video digitizer component to return the sound input source associated with a particular video input.

`VDReleaseCompressBuffer` (page 71)
Frees a buffer received from VDCompressDone.

`VDResetCompressSequence` (page 72)
Forces the video digitizer to insert a key frame into a temporally compressed image sequence.

`VDSetCompression` (page 76)
Specifies certain compression parameters.

`VDSetCompressionOnOff` (page 77)
Allows an application to start and stop compression by video digitizers that can deliver either compressed or uncompressed image data.

`VDSetDataRate` (page 78)
Instructs your video digitizer component to limit the rate at which it delivers compressed, digitized video data.

`VDSetTimeBase` (page 93)
Establishes the video digitizer's time coordinate system.

Controlling Digitization

`VDCaptureStateChanging` (page 28)
Provides process information from a sequence grabber component to the VDIG.

`VDDone` (page 31)
Determines if VDGrabOneFrameAsync is finished with a specific output buffer.

`VDGetDataRate` (page 38)
Retrieves information that describes the performance capabilities of a video digitizer.

`VDGetTimeCode` (page 59)
Instructs your video digitizer component to return timecode information for the incoming video signal.

`VDGrabOneFrame` (page 63)
Instructs the video digitizer component to digitize a single frame of source video.

`VDGrabOneFrameAsync` (page 64)
Instructs the video digitizer component to start to digitize asynchronously a single frame of source video.

`VDReleaseAsyncBuffers` (page 71)
Releases the buffers that were allocated with VDSetupBuffers.

`VDSetFrameRate` (page 82)
Indicates an application's desired frame rate to the video digitizer.

`VDSetPlayThruOnOff` (page 89)
Controls continuous digitization.

`VDSetPreferredPacketSize` (page 91)
Sets the preferred packet size for video digitizing.

`VDSetupBuffers` (page 93)
Defines output buffers for use with asynchronous grabs.

Controlling the Video Output Display Mode

`QTVideoOutputGetDisplayMode` (page 20)
Returns the current display mode for a video output component.

`QTVideoOutputGetDisplayModeList` (page 20)
Returns a list of the display modes supported by a video output component.

`QTVideoOutputSetDisplayMode` (page 26)
Specifies the display mode to be used by a video output component.

Controlling Video Output

`QTVideoOutputBaseSetEchoPort` (page 14)
Called on the base video output component to inform it about a change in the echo port.

`QTVideoOutputBegin` (page 15)
Obtains exclusive access to the video hardware controlled by a video output component.

`QTVideoOutputCustomConfigureDisplay` (page 16)
Displays a custom video configuration dialog box, which can include settings that are specific to the video device controlled by the video output component.

`QTVideoOutputEnd` (page 17)
Releases access to the video hardware controlled by a video output component.

`QTVideoOutputGetGWorld` (page 21)
Returns a pointer to the graphics world used by a video output component.

`QTVideoOutputGetGWorldParameters` (page 22)
Called by the base video output component as part of its implementation of QTVideoOutputGetGWorld.

`QTVideoOutputSetEchoPort` (page 27)
Specifies a window on the desktop in which to display video sent to the device.

Finding Components Associated With a Video Output

`QTVideoOutputGetClock` (page 18)
Returns a pointer to the clock component associated with the video output component.

`QTVideoOutputGetIndSoundOutput` (page 23)
Determines which sound output components are associated with the video output component.

Getting Information About Video Digitizer Components

`VDGetCurrentFlags` (page 37)
Returns status information about a specified video digitizer component.

`VDGetDeviceNameAndFlags` (page 38)
Returns the current name and device visibility of a video digitizer.

`VDGetDigitizerInfo` (page 39)
Returns capability and status information about a specified video digitizer component.

`VDGetUniqueIDs` (page 60)
Returns a unique identifier for a particular video digitizer device.

`VDSelectUniqueIDs` (page 72)
Selects a video digitizer device by ID.

Registering the Name of Video Output Software

`QTVideoOutputGetClientName` (page 18)
Obtains the name of the application or other software that is registered with an instance of a video output component.

`QTVideoOutputGetCurrentClientName` (page 19)
Returns the name of the software, if any, that has exclusive access to the video hardware controlled by a video output component.

`QTVideoOutputSetClientName` (page 25)
Registers the name of an application or other software with an instance of a video output component.

Saving and Restoring Component Configurations

`QTVideoOutputRestoreState` (page 24)
Restores the previously saved state of a video output component.

`QTVideoOutputSaveState` (page 24)
Saves state information for an instance of a video output component.

Selecting an Input Source

`VDGetInput` (page 44)
Returns data that identifies the currently active input video source.

`VDGetInputFormat` (page 45)
Determines the format of the video signal provided by a specified video input source.

`VDGetNumberOfInputs` (page 53)
Returns the number of input video sources that a video digitizer component supports.

`VDSetInput` (page 83)
Selects the input video source for a video digitizer component.

`VDSetInputStandard` (page 85)
Specifies the input signaling standard to digitize.

Selectively Displaying Video

`VDAddKeyColor` (page 27)
Adds a key color to a component's list of active key colors.

`VDGetKeyColor` (page 48)
Obtains the index value of the active key color.

VDGetKeyColorRange (page 49)
Obtains the currently defined key color range.

VDGetMaskandValue (page 49)
Obtains the appropriate alpha channel or blend mask value for a desired level of video blending.

VDGetMaskPixMap (page 50)
Retrieves the pixel map data for a component's blend mask.

VDGetNextKeyColor (page 52)
Obtains the index value of the active key colors in cases where the digitizer component supports multiple key colors.

VDSetKeyColor (page 86)
Sets the key color for video digitizing.

VDSetKeyColorRange (page 86)
Defines a key color range for video digitizing.

VDSetMasterBlendLevel (page 87)
Sets the blend level value for the input video signal.

Setting Source Characteristics

VDGetActiveSrcRect (page 31)
Obtains size and location information for the active source rectangle used by a video digitizer component.

VDGetDigitizerRect (page 40)
Returns the current digitizer rectangle.

VDGetMaxSrcRect (page 51)
Returns the maximum source rectangle.

VDGetVBlankRect (page 60)
Returns the vertical blanking rectangle.

VDSetDigitizerRect (page 79)
Sets the current video digitizer rectangle.

Setting Video Destinations

VDGetMaxAuxBuffer (page 51)
Obtains access to buffers that are located on special hardware.

VDGetPlayThruDestination (page 53)
Obtains information about the current video destination.

VDPreflightDestination (page 69)
Verifies that a video digitizer component can support a set of destination settings intended for use with VDSetPlayThruDestination.

VDPreflightGlobalRect (page 70)
Verifies that a video digitizer component can support a set of destination settings intended for use with VDSetPlayThruGlobalRect.

VDSetPlayThruDestination (page 88)
Establishes the destination settings for a video digitizer component.

VDSetPlayThruGlobalRect (page 89)
Establishes the destination settings for a video digitizer component that is to digitize into a global rectangle.

Video Clipping

VDClearClipRgn (page 28)
Disables all or part of a clipping region that was previously set with VDSetClipRgn.

VDGetClipState (page 33)
Determines whether clipping is enabled.

VDSetClipRgn (page 74)
Defines a clipping region for a video digitizer.

VDSetClipState (page 75)
Controls whether clipping is enabled.

Video Digitizer Utilities

VDGetFieldPreference (page 42)
Determines which field is being used in cases where the image is vertically scaled to half its original size.

VDGetPLLFilterType (page 54)
Determines which phase locked loop (PLL) mode is currently active for a video digitizer.

VDGetPreferredTimeScale (page 56)
Determines a digitizer's preferred time scale.

VDGetSoundInputDriver (page 57)
Retrieves information about a video digitizer's sound input driver.

VDSetDigitizerUserInterrupt (page 80)
Sets custom interrupt functions.

VDSetFieldPreference (page 81)
Specifies which field to use in cases where the vertical scaling is less than half size.

VDSetPLLFilterType (page 90)
Specifies which phase locked loop (PLL) is to be active.

Supporting Functions

QTVideoOutputCopyIndAudioOutputDeviceUID (page 16)
Identifies the audio device being used by a video output component.

QTVideoOutputGetIndImageDecompressor (page 23)
Undocumented

VDGetInputGammaRecord (page 46)
Retrieves a pointer to the active input VDGammaRecord structure for a video digitizer.

VDGetInputName (page 47)
Gets the name of a video input.

`VDGetPreferredImageDimensions` (page 55)
: Gets the preferred image dimensions for a video digitizer.

`VDIIDCGetCSRData` (page 64)
: Reads a camera's CSR registers directly.

`VDIIDCGetDefaultFeatures` (page 65)
: Places atoms in a QuickTime atom container that specify the default capabilities and default state of a camera's IIDC features.

`VDIIDCGetFeatures` (page 66)
: Places atoms in a QuickTime atom container that specify the current capabilities of a camera and the state of its IIDC features.

`VDIIDCGetFeaturesForSpecifier` (page 67)
: Places atoms in a QuickTime atom container that specify the current state of a single camera IIDC feature or group of features.

`VDIIDCSetCSRData` (page 67)
: Writes to a camera's CSR registers directly.

`VDIIDCSetFeatures` (page 68)
: Changes the state of a camera's IIDC features.

`VDSetDestinationPort` (page 79)
: Sets the destination port for a video digitizer.

`VDSetInputGammaRecord` (page 84)
: Changes the active input gamma data structure.

`VDSetPreferredImageDimensions` (page 91)
: Sets the preferred image dimensions for a video digitizer.

`VDUseSafeBuffers` (page 95)
: Instructs a video digitizer to use protected buffers.

Functions

QTVideoOutputBaseSetEchoPort

Called on the base video output component to inform it about a change in the echo port.

```
ComponentResult QTVideoOutputBaseSetEchoPort (
   QTVideoOutputComponent vo,
   CGrafPtr echoPort
);
```

Parameters

vo
: The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

echoPort
: The window on the computer's desktop in which to display the video.

Return Value

See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
SoftVideoOutputComponent

Declared In
`QuickTimeComponents.h`

QTVideoOutputBegin

Obtains exclusive access to the video hardware controlled by a video output component.

```
ComponentResult QTVideoOutputBegin (
   QTVideoOutputComponent vo
);
```

Parameters

vo

The instance of a video output component. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error. If this function returns the `videoOutputInUseErr` result code that indicates that the video hardware is currently in use, your software can get the name of the application or other software that is using the hardware by calling `QTVideoOutputGetCurrentClientName` (page 19). You can then display an alert to the user that says that the video hardware is in use and specifies the name of the software using the video hardware.

Discussion
When your software calls this function, the video output component acquires exclusive access to the video hardware controlled by the specified video output component or returns the `videoOutputInUseErr` result code if the video hardware is currently in use. If the video hardware is available, the video output component also enables the display mode last set with `QTVideoOutputSetDisplayMode` (page 26) and enables the video settings, if any, that were most recently specified by the user in a custom video configuration dialog box. If the video output component supports `QTVideoOutputCustomConfigureDisplay` (page 16), your software can call the function to display a custom video configuration dialog box. When your software no longer needs the video output component, release it by calling `QTVideoOutputEnd` (page 17).

Special Considerations

If your software needs to change the display mode, it must change it before calling this function. It cannot change the display mode between calls to this function and to `QTVideoOutputEnd` (page 17).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Live DV

Quartz Composer QCTV
SoftVideoOutputComponent

Declared In
`QuickTimeComponents.h`

QTVideoOutputCopyIndAudioOutputDeviceUID

Identifies the audio device being used by a video output component.

```
ComponentResult QTVideoOutputCopyIndAudioOutputDeviceUID (
   QTVideoOutputComponent vo,
   long index,
   CFStringRef *audioDeviceUID
);
```

Parameters

vo
: Video output component whose audio output is being asked about.

index
: Which of video output component's audio outputs is being asked about.

audioDeviceUID
: Returned unique identifier for the audio device. If the UID is `NIL`, the movie is playing to the default device.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error. Returns `badComponentInstance` if `vo` is not a valid `ComponentInstance`. Returns `badComponentSelector` if `vo` doesn't support this function. Returns `paramErr` if `audioDeviceUID` is `NIL`, or if there is no device with the passed index.

Discussion
The returned `audioDeviceUID` has already been retained for the caller, using standard Core Foundation copy semantics.

Version Notes
Introduced in QuickTime 6.4.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputCustomConfigureDisplay

Displays a custom video configuration dialog box, which can include settings that are specific to the video device controlled by the video output component.

```
ComponentResult QTVideoOutputCustomConfigureDisplay (
   QTVideoOutputComponent vo,
   ModalFilterUPP filter
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

filter

A `ModalFilterProc` callback for the video output component to use for the dialog box. The filter allows the software to process events while the dialog box is displayed.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Your software can determine if a video output component supports this function by calling `ComponentFunctionImplemented` for the component with the routine selector `kQTVideoOutputCustomConfigureDisplaySelect`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputEnd

Releases access to the video hardware controlled by a video output component.

```
ComponentResult QTVideoOutputEnd (
   QTVideoOutputComponent vo
);
```

Parameters

vo

The instance of a video output component. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Your software should release access to a video output component as soon as it is done using the video hardware controlled by the component. If you close the instance of a video output component that currently has exclusive access to video hardware, the video output component automatically calls this function to release the hardware.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Live DV
Quartz Composer QCTV
SoftVideoOutputComponent

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetClientName

Obtains the name of the application or other software that is registered with an instance of a video output component.

```
ComponentResult QTVideoOutputGetClientName (
   QTVideoOutputComponent vo,
   Str255 str
);
```

Parameters

vo

The instance of a video output component for the request. Your software obtains this reference when it calls `OpenComponent` or `OpenDefaultComponent`.

str

The name of the application or other software that is registered with the component instance.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetClock

Returns a pointer to the clock component associated with the video output component.

```
ComponentResult QTVideoOutputGetClock (
   QTVideoOutputComponent vo,
   ComponentInstance *clock
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

clock

A pointer to the clock component associated with the video output component.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Your software can use the clock component returned by this function to synchronize video and sound for a movie to the rate of the display. To associate the instance of the clock component with a movie, call `SetMovieMasterClock`. Because a change to the display mode could affect a clock component, your software should call this function only between calls to `QTVideoOutputBegin` (page 15) and `QTVideoOutputEnd` (page 17), when it is not possible to change the display mode.

Special Considerations

When your software calls `QTVideoOutputEnd` (page 17), the video output component disposes of the instance of the clock component returned by this function. Because of this, software that uses the clock to control a movie must reset the clock for the movie to the default clock, by calling `SetMovieMasterClock` with `NIL` as the value of the clock component, before calling `QTVideoOutputEnd`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetCurrentClientName

Returns the name of the software, if any, that has exclusive access to the video hardware controlled by a video output component.

```
ComponentResult QTVideoOutputGetCurrentClientName (
   QTVideoOutputComponent vo,
   Str255 str
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

str

The name of the software that has exclusive access to the video hardware controlled by a video output component, or a zero-length string if no software currently has access.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
If video hardware is unavailable because other software is using it, your software can inform users by getting the name of the software with this function and displaying the name in an alert box.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetDisplayMode

Returns the current display mode for a video output component.

```
ComponentResult QTVideoOutputGetDisplayMode (
   QTVideoOutputComponent vo,
   long *displayModeID
);
```

Parameters

vo

The instance of a video output component. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

displayModeID

A pointer to the ID of the current display mode, or 0 if no display mode has been selected. The ID specifies a QT atom of type `kQTVODisplayModeItem` in the QT atom container returned by `QTVideoOutputGetDisplayModeList` (page 20).

Return Value
See `Error Codes`. Returns `noErr` if there is no error. If this function returns an atom ID of 0, it indicates that no display mode has been selected.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
SoftVideoOutputComponent

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetDisplayModeList

Returns a list of the display modes supported by a video output component.

```
ComponentResult QTVideoOutputGetDisplayModeList (
   QTVideoOutputComponent vo,
   QTAtomContainer *outputs
);
```

Parameters

vo

The instance of a video output component. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

outputs

A pointer to the QT atom container that lists the video modes supported by this component.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
After your software calls this function, it must dispose of the QT atom container returned by the function by calling `QTDisposeAtomContainer`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetGWorld

Returns a pointer to the graphics world used by a video output component.

```
ComponentResult QTVideoOutputGetGWorld (
   QTVideoOutputComponent vo,
   GWorldPtr *gw
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

gw

A pointer to the graphics world used by the video output component to display images.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
If the pixel format of the graphics world is 1, 2, 4, 8, 16, or 32, your software can use either QuickDraw or QuickTime to draw graphics to it. If the graphics world has any other pixel format, your software must use QuickTime functions draw to it. Your software can pass the pointer returned by this function to the `SetMovieGWorld`, `DecompressSequenceBegin`, `DecompressSequenceBeginS`, `DecompressImage`, and `FDecompressImage` functions.

Your software can call `QTVideoOutputGetGWorld` only between calls to `QTVideoOutputBegin` (page 15) and `QTVideoOutputEnd` (page 17). When your software calls `QTVideoOutputEnd`, the video output component automatically disposes of the graphics world. If your software needs to use the graphics world after calling `QTVideoOutputEnd`, it must call this function again after the next time it calls `QTVideoOutputBegin`.

Special Considerations

Your software must not dispose of the graphics world used by a video output component. The video output component automatically disposes of the graphics world when your software calls `QTVideoOutputEnd` (page 17).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Live DV
Quartz Composer QCTV

Declared In
`QuickTimeComponents.h`

QTVideoOutputGetGWorldParameters

Called by the base video output component as part of its implementation of QTVideoOutputGetGWorld.

```
ComponentResult QTVideoOutputGetGWorldParameters (
   QTVideoOutputComponent vo,
   Ptr *baseAddr,
   long *rowBytes,
   CTabHandle *colorTable
);
```

Parameters

vo
: An instance of your video output component.

baseAddr
: The address at which to display pixels. If your video output component does not display pixels, return 0 for this parameter.

rowBytes
: The width of each scan line in bytes. If your video output component does not display pixels, return the width of the current display mode.

colorTable
: The `ColorTable` structure to be used. If your video output component does not use a color table, return `NIL`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
This function is not called by applications or other client software.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

QTVideoOutputGetIndImageDecompressor

Undocumented

```
ComponentResult QTVideoOutputGetIndImageDecompressor (
   QTVideoOutputComponent vo,
   long index,
   Component *codec
);
```

Parameters

vo
Undocumented

index
Undocumented

codec
Undocumented

Return Value
See Error Codes. Returns noErr if there is no error.

Version Notes
Introduced in QuickTime 5.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

QTVideoOutputGetIndSoundOutput

Determines which sound output components are associated with the video output component.

```
ComponentResult QTVideoOutputGetIndSoundOutput (
   QTVideoOutputComponent vo,
   long index,
   Component *outputComponent
);
```

Parameters

vo
The instance of a video output component for this request. Your software obtains this reference when calling OpenComponent or OpenDefaultComponent.

index
Specifies which of the sound output components to return. The index of the first component is 1.

outputComponent
A pointer to a sound output component associated with the video output component that is specified by the index parameter.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Your software can display sound output components returned by this function in a dialog box and let the user choose which outputs to use for movie playback.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputRestoreState

Restores the previously saved state of a video output component.

```
ComponentResult QTVideoOutputRestoreState (
   QTVideoOutputComponent vo,
   QTAtomContainer state
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

state

A QT atom container, returned earlier by `QTVideoOutputSaveState` (page 24), that contains state information for the video output component.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
If your software saves state information to disk, it must read the QT atom container structure from disk before calling this function. When your software restores state information for a video output component, the current display mode may change. Because of this, your software must call this function before calling `QTVideoOutputBegin` (page 15).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputSaveState

Saves state information for an instance of a video output component.

```
ComponentResult QTVideoOutputSaveState (
   QTVideoOutputComponent vo,
   QTAtomContainer *state
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

state

A pointer to complete information about the video output component's current configuration.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
When your software saves state information for an instance of a video output component, it can restore this information when reconnecting to the component by calling `QTVideoOutputRestoreState` (page 24).

Special Considerations

When your software calls this function, it must dispose of the QT atom container returned by the function by calling `QTDisposeAtomContainer`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

QTVideoOutputSetClientName

Registers the name of an application or other software with an instance of a video output component.

```
ComponentResult QTVideoOutputSetClientName (
   QTVideoOutputComponent vo,
   ConstStr255Param str
);
```

Parameters

vo

The instance of a video output component for the request. Your software obtains this reference when it calls `OpenComponent` or `OpenDefaultComponent`.

str

The name of the application or other software to be registered.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
The name you specify with this function can later be used by `QTVideoOutputGetCurrentClientName` (page 19) to specify which software has exclusive access to the video output device controlled by the component.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Live DV
Quartz Composer QCTV

Declared In
`QuickTimeComponents.h`

QTVideoOutputSetDisplayMode

Specifies the display mode to be used by a video output component.

```
ComponentResult QTVideoOutputSetDisplayMode (
   QTVideoOutputComponent vo,
   long displayModeID
);
```

Parameters

vo

The instance of a video output component for the request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

displayModeID

The ID of the display mode to use. The ID specifies a QT atom of type `kQTVODisplayModeItem` in the QT atom container returned by `QTVideoOutputGetDisplayModeList` (page 20).

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
When software changes the display mode with this function, the change does not take effect until the next time the software calls `QTVideoOutputBegin` (page 15) for the video output component. This lets the software change other output settings before displaying the video.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Quartz Composer Live DV
Quartz Composer QCTV

Declared In
`QuickTimeComponents.h`

QTVideoOutputSetEchoPort

Specifies a window on the desktop in which to display video sent to the device.

```
ComponentResult QTVideoOutputSetEchoPort (
   QTVideoOutputComponent vo,
   CGrafPtr echoPort
);
```

Parameters

vo

The instance of a video output component for this request. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

echoPort

The window on the computer's desktop in which to display the video.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
When your software sends video to the window you specify, the video is both displayed in the window and sent to the normal output of the video output device. When an output device can display video both on an external video display and in a window on a computer's desktop, the video displayed on the desktop is often at a smaller size and/or lower frame rate.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDAddKeyColor

Adds a key color to a component's list of active key colors.

```
VideoDigitizerError VDAddKeyColor (
   VideoDigitizerComponent ci,
   long *index
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

index

A pointer to the color to add to the key color list. The value of the `index` field corresponds to a color in the current color lookup table.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDCaptureStateChanging

Provides process information from a sequence grabber component to the VDIG.

```
VideoDigitizerError VDCaptureStateChanging (
   VideoDigitizerComponent ci,
   UInt32 inStateFlags
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inStateFlags

Constants (see below) that tell the VDIG what is about to happen. See these constants:

`vdFlagCaptureStarting`
`vdFlagCaptureStopping`
`vdFlagCaptureIsForPreview`
`vdFlagCaptureIsForRecord`
`vdFlagCaptureLowLatency`
`vdFlagCaptureAlwaysUseTimeBase`

Return Value
An error return of type `ComponentResult`. See `Error Codes`. Returns `noErr` if there is no error.

Discussion
It has long been a problem for VDIG writers that the sequence grabber can make a series of calls to a VDIG and it is not always clear what their intent is. This function lets you provide additional information about what is happening at the sequence grabber level to the VDIG, so it can take this into account. In particular, the settings bracketing calls are designed for the VDIG to update a series of parameters without reinitializing.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`QuickTimeComponents.h`

VDClearClipRgn

Disables all or part of a clipping region that was previously set with VDSetClipRgn.

```
VideoDigitizerError VDClearClipRgn (
   VideoDigitizerComponent ci,
   RgnHandle clipRegion
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

clipRegion

A handle to a `MacRegion` structure that defines the clipping region to clear. This region must correspond to all or part of the clipping region established previously with `VDSetClipRgn` (page 74).

Return Value

See `Error Codes`. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

VDCompressDone

Determines whether the video digitizer has finished digitizing and compressing a frame of image data.

```
VideoDigitizerError VDCompressDone (
   VideoDigitizerComponent ci,
   UInt8 *queuedFrameCount,
   Ptr *theData,
   long *dataSize,
   UInt8 *similarity,
   TimeRecord *t
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

queuedFrameCount

A pointer to the number of queued frames yet to be done. 0 means no frames. Some VDIGs may return 2 even if more than 2 frames are available, and some will return 1 if any number more than 0 are available.

theData

A pointer to a field that is to receive a pointer to the compressed image data. The digitizer returns a pointer that is valid in the application's current memory mode.

dataSize

A pointer to a field to receive a value indicating the number of bytes of compressed image data.

similarity

A pointer to a field to receive an indication of the relative similarity of this image to the previous image in a sequence. A value of 0 indicates that the current frame is a key frame in the sequence. A value of 255 indicates that the current frame is identical to the previous frame. Values from 1 through 254 indicate relative similarity, ranging from very different (1) to very similar (254). This field is only filled in if the temporal quality passed in with `VDSetCompression` (page 76) is not 0; that is, if it is not frame-differenced.

t

A pointer to a `TimeRecord` structure. When the operation is complete, the digitizer fills in this structure with information indicating when the frame was grabbed. The time value stored in this structure is in the time base that the application sets with `VDSetTimeBase` (page 93).

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDCompressOneFrameAsync

Instructs the video digitizer to digitize and compress a single frame of image data.

```
VideoDigitizerError VDCompressOneFrameAsync (
   VideoDigitizerComponent ci
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Unlike `VDGrabOneFrameAsync` (page 64), this function causes the video digitizer to handle all details of managing data buffers.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDDone

Determines if VDGrabOneFrameAsync is finished with a specific output buffer.

```
VideoDigitizerError VDDone (
   VideoDigitizerComponent ci,
   short buffer
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

buffer

Identifies the buffer for the operation. The value of this parameter must correspond to a valid index into the list of buffers you supply when your application calls `VDSetupBuffers` (page 93). This value is zero-based; that is, you must set this parameter to 0 to refer to the first buffer in the buffer list.

Return Value

Returns a long integer indicating whether the specified asynchronous frame grab is complete. If the returned value is 0, the video digitizer component is still working on the frame. If the returned value is nonzero, the digitizer component is finished with the frame and the application can perform its processing.

Discussion

Applications can determine whether a video digitizer component supports asynchronous frame grabbing by examining the output capability flags of the digitizer component, using `VDGetCurrentFlags` (page 37). Specifically, if the `digiOutDoesAsyncGrabs` flag is set to 1, the digitizer component supports both this function and `VDGrabOneFrameAsync` (page 64).

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

VDGetActiveSrcRect

Obtains size and location information for the active source rectangle used by a video digitizer component.

```
VideoDigitizerError VDGetActiveSrcRect (
   VideoDigitizerComponent ci,
   short inputStd,
   Rect *activeSrcRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputStd

A short integer that specifies the input video signal associated with this maximum source rectangle.

activeSrcRect

A pointer to a `Rect` structure that is to receive the size and location information for the active source rectangle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Special Considerations

All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetBlackLevelValue

Returns the current black level value.

```
VideoDigitizerError VDGetBlackLevelValue (
   VideoDigitizerComponent ci,
   unsigned short *blackLevel
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

blackLevel

A pointer to an integer field that is to receive the current black level value. Black level values range from 0 to 65,535, where 0 represents the maximum black value and 65,535 represents the minimum black value.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ExampleVideoPanel
ExampleVideoPanel.win

Declared In
`QuickTimeComponents.h`

VDGetBrightness

Returns the current brightness value.

```
VideoDigitizerError VDGetBrightness (
   VideoDigitizerComponent ci,
   unsigned short *brightness
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

brightness

A pointer to an integer field that is to receive the current brightness value. Brightness values range from 0 to 65,535, where 0 is the darkest possible setting and 65,535 is the lightest possible setting.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetClipState

Determines whether clipping is enabled.

```
VideoDigitizerError VDGetClipState (
   VideoDigitizerComponent ci,
   short *clipEnable
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

clipEnable

A pointer to a short integer field that is to receive a value indicating whether clipping is enabled. The video digitizer component places 0 into the field if clipping is disabled, and 1 if it is enabled.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDGetCLUTInUse

Obtains the color lookup table used by a video digitizer component.

```
VideoDigitizerError VDGetCLUTInUse (
   VideoDigitizerComponent ci,
   CTabHandle *colorTableHandle
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

colorTableHandle

A pointer to a field that is to receive a handle to a `ColorTable` structure. The video digitizer component returns a handle to its color lookup table. Applications can then set the destination to use this returned `ColorTable` structure. Your application is responsible for disposing of this handle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDGetCompressionTime

Confirms or quantifies a video digitizer's compression settings.

```
VideoDigitizerError VDGetCompressionTime (
   VideoDigitizerComponent ci,
   OSType compressionType,
   short depth,
   Rect *srcRect,
   CodecQ *spatialQuality,
   CodecQ *temporalQuality,
   unsigned long *compressTime
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

compressionType

A compressor type. This value corresponds to the `component` subtype of the compressor component. See `Codec Identifiers`.

depth

The depth at which the image is to be compressed. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images.

srcRect

A pointer to a `Rect` structure that defines the portion of the source image to compress.

spatialQuality

A pointer to a field containing the desired compressed image quality (see below). The compressor sets this field to the closest actual quality that it can achieve. A value of `NIL` indicates that the client does not want this information. See these constants:

```
codecMinQuality
codecLowQuality
codecNormalQuality
codecHighQuality
codecMaxQuality
codecLosslessQuality
```

temporalQuality

A pointer to a field containing the desired sequence temporal quality (see below). The compressor sets this field to the closest actual quality that it can achieve. A value of `NIL` indicates that the client does not want this information.

compressTime

A pointer to a field to receive the compression time, in milliseconds. Your component should return a long integer indicating the maximum number of milliseconds it would require to compress the specified image. If your component cannot determine the amount of time required to compress the image, set this field to 0. A value of `NIL` indicates that the client does not want this information.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
The sequence grabber's video compression settings dialog box uses this function to snap the quality slider to the correct value when working with a compression type that is specified by the video digitizer.

Version Notes
In QuickTime 1.5, video digitizers could provide compressed data directly to clients; however, there was no way to preflight the settings for compression. In QuickTime 2.1, this function was added to allow the video digitizer to quantify the compression time for the actual quality levels that will be used.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetCompressionTypes

Determines the image-compression capabilities of the video digitizer.

```
VideoDigitizerError VDGetCompressionTypes (
   VideoDigitizerComponent ci,
   VDCompressionListHandle h
);
```

Parameters

ci

Identifies an application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

h

A handle to receive the compression information in one or more `VDCompressionList` structures. If the digitizer supports more than one compression type, it creates an array of structures in this handle. The video digitizer returns information about its capabilities by formatting these structures.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Special Considerations

There must be a decompressor component of the appropriate type available in the system if an application is to display images from a compressed image sequence.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetContrast

Returns the current contrast value.

```
VideoDigitizerError VDGetContrast (
   VideoDigitizerComponent ci,
   unsigned short *contrast
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

contrast

A pointer to an integer field that is to receive the current contrast value. The contrast value ranges from 0 to 65,535, where 0 represents no change to the basic image and larger values increase the contrast of the video image (they increase the slope of the transform).

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetCurrentFlags

Returns status information about a specified video digitizer component.

```
VideoDigitizerError VDGetCurrentFlags (
   VideoDigitizerComponent ci,
   long *inputCurrentFlag,
   long *outputCurrentFlag
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputCurrentFlag

A pointer to a long integer that is to receive the current input state flags for the video digitizer component; see `Video Digitizer Capabilities`.

outputCurrentFlag

A pointer to a long integer that is to receive the current output state flags for the video digitizer component; see `Video Digitizer Capabilities`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
This function is often more convenient than `VDGetDigitizerInfo` (page 39). For example, this function provides a simple mechanism for determining whether a video digitizer is receiving a valid input signal. An application can retrieve the current input state flags and test the high-order bit by examining the sign of the returned value. If the value is negative (that is, the high-order bit, `digiInSignalLock`, is set to 1), the digitizer component is receiving a valid input signal.

Special Considerations

All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
QT Internals

Declared In
QuickTimeComponents.h

VDGetDataRate

Retrieves information that describes the performance capabilities of a video digitizer.

```
VideoDigitizerError VDGetDataRate (
   VideoDigitizerComponent ci,
   long *milliSecPerFrame,
   Fixed *framesPerSecond,
   long *bytesPerSecond
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

milliSecPerFrame

A pointer to a long integer. The video digitizer returns a value that indicates the number of milliseconds of synchronous overhead involved in digitizing a single frame. This value includes the average delay incurred between the time when the digitizer requests a frame from its associated device, and the time at which the device delivers the frame.

framesPerSecond

A pointer to a fixed value. The video digitizer supplies the maximum rate at which it can capture video. Note that this value may differ from the rate that the application set with `VDSetFrameRate` (page 82).

bytesPerSecond

A pointer to a long integer. Video digitizers that can return compressed image data return a value that indicates the approximate number of bytes per second that the digitizer is generating compressed data, given the current compression and frame rate settings.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BrideOfMungGrab
Fiendishthngs

Declared In
QuickTimeComponents.h

VDGetDeviceNameAndFlags

Returns the current name and device visibility of a video digitizer.

```
VideoDigitizerError VDGetDeviceNameAndFlags (
   VideoDigitizerComponent ci,
   Str255 outName,
   UInt32 *outNameFlags
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

outName

The video digitizer device name.

outNameFlags

A pointer to a constant (see below) that determines whether to show or hide the VDIG device. See these constants:

`vdDeviceFlagShowInputsAsDevices`

`vdDeviceFlagHideDevice`

Return Value
An error return of type `ComponentResult`. See `Error Codes`. Returns `noErr` if there is no error.

Discussion
This routine is designed to give the VDIG more control over how it is presented to the user, and to clarify the distinction between devices and inputs. Historically, the assumption has been that there is one component registered per device and that the component name is displayed. This function lets a component choose its name after registration. When this function is called, it is also a good time to check for hardware and register further VDIG components if needed, allowing for lazy initialization when the application needs to find a VDIG rather than initializing at every launch or replug.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDGetDigitizerInfo

Returns capability and status information about a specified video digitizer component.

```
VideoDigitizerError VDGetDigitizerInfo (
   VideoDigitizerComponent ci,
   DigitizerInfo *info
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

info

A pointer to a `DigitizerInfo` structure. The function returns information describing the capabilities of the specified video digitizer into this structure.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Special Considerations

All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DigitizerShell

Declared In
`QuickTimeComponents.h`

VDGetDigitizerRect

Returns the current digitizer rectangle.

```
VideoDigitizerError VDGetDigitizerRect (
   VideoDigitizerComponent ci,
   Rect *digitizerRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

digitizerRect

A pointer to a `Rect` structure that is to receive the size and location information for the current digitizer rectangle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetDMADepths

Determines which pixel depths a digitizer supports.

```
VideoDigitizerError VDGetDMADepths (
   VideoDigitizerComponent ci,
   long *depthArray,
   long *preferredDepth
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

depthArray

A pointer to a long integer. The video digitizer returns a value that indicates the depths it can support. Each depth is represented by a single bit in this field. More than one bit may be set to 1.

preferredDepth

A pointer to a long integer. Video digitizers that have a preferred depth value return that value in this field, using one of the possible values of the `depthArray` parameter. Digitizers that do not prefer any given value set this field to 0.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
The flags returned by this function augment the information that an application can obtain from the digitizer's output capability flags in the `DigitizerInfo` structure. If a digitizer does not support this function but does support DMA, an application may assume that the digitizer can handle offscreen buffers at all of the depths indicated in its output capabilities flags. Applications may use the following enumerators to set bits in the field referred to by the `depthArray` parameter.

```
enum {
    dmaDepth1        =1     /* supports black and white */
    dmaDepth2        =2     /* supports 2-bit color */
    dmaDepth4        =4     /* supports 4-bit color */
    dmaDepth8        =8     /* supports 8-bit color */
    dmaDepth16       =16    /* supports 16-bit color */
    dmaDepth32       =32    /* supports 32-bit color */
    dmaDepth2Gray    =64    /* supports 2-bit grayscale */
    dmaDepth4Gray    =128   /* supports 4-bit grayscale */
    dmaDepth8Gray    =256   /* supports 8-bit grayscale */
};
```

Special Considerations

Before a program that uses a video digitizer creates an offscreen buffer, it should call the this function to determine the pixel depths supported by the digitizer. If possible, the program should use the preferred depth, in order to obtain the best possible display performance.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetFieldPreference

Determines which field is being used in cases where the image is vertically scaled to half its original size.

```
VideoDigitizerError VDGetFieldPreference (
   VideoDigitizerComponent ci,
   short *fieldFlag
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

fieldFlag

Points to a field that is to receive a value (see below) indicating which field is being used. See these constants:

`vdUseAnyField`

`vdUseOddField`

`vdUseEvenField`

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetHue

Returns the current hue value.

```
VideoDigitizerError VDGetHue (
   VideoDigitizerComponent ci,
   unsigned short *hue
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

hue

A pointer to an integer that is to receive the current hue value. Hue is similar to the tint control on a television, and it is specified in degrees with complementary colors set 180 degrees apart (red is 0 degrees, green is +120 degrees, and blue is -120 degrees). Video digitizer components support hue values that range from 0 (-180 degrees shift in hue) to 65,535 (+179 degrees shift in hue), where 32,767 represents a 0 degree shift in hue.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetImageDescription

Retrieves an ImageDescription structure from a video digitizer.

```
VideoDigitizerError VDGetImageDescription (
   VideoDigitizerComponent ci,
   ImageDescriptionHandle desc
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

desc

A handle. The video digitizer fills this handle with an `ImageDescription` structure containing information about the digitizer's current compression settings. The digitizer resizes the handle appropriately. It is the application's responsibility to dispose of this handle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDGetInput

Returns data that identifies the currently active input video source.

```
VideoDigitizerError VDGetInput (
   VideoDigitizerComponent ci,
   short *input
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

input

A pointer to a short integer that is to receive the identifier for the currently active input video source. Video digitizer components number video sources sequentially, starting at 0. So, if the first source is active, this function sets the field referred to by the `input` parameter to 0.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Special Considerations

All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDGetInputColorSpaceMode

Determines whether a digitizer is operating in color or grayscale mode.

```
VideoDigitizerError VDGetInputColorSpaceMode (
   VideoDigitizerComponent ci,
   short *colorSpaceMode
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

colorSpaceMode

A pointer to a value that indicates whether the digitizer is operating in color (1) or grayscale (0) mode.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Applications can determine whether a digitizer component supports grayscale or color digitization by examining the digitizer component's input capability flags. Specifically, if the `digiInDoesColor` flag is set to 1, the digitizer component supports color digitization. Similarly, if the `digiInDoesBW` flag is set to 1, the digitizer component supports grayscale digitization. Applications can use `VDGetCurrentFlags` (page 37) to obtain the input capability flags of a digitizer component.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetInputFormat

Determines the format of the video signal provided by a specified video input source.

```
VideoDigitizerError VDGetInputFormat (
   VideoDigitizerComponent ci,
   short input,
   short *format
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

input

The input video source for this request. Video digitizer components number video sources sequentially, starting at 0. So, to request information about the first video source, an application sets this parameter to 0. Applications can get the number of video sources supported by a video digitizer component by calling `VDGetNumberOfInputs` (page 53).

format

A pointer to a short integer that is to receive a constant (see below) that specifies the video format of the specified input source. See these constants:

```
compositeIn
sVideoIn
rgbComponentIn
```

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetInputGammaRecord

Retrieves a pointer to the active input VDGammaRecord structure for a video digitizer.

```
VideoDigitizerError VDGetInputGammaRecord (
   VideoDigitizerComponent ci,
   VDGamRecPtr *inputGammaPtr
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputGammaPtr

A pointer to a field that is to receive a pointer to an input `VDGammaRecord` structure.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Gamma structures give applications complete control over color filtering transforms and are therefore more precise than the gamma values that can be set by calling `VDSetInputGammaValue` (page 84).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetInputGammaValue

Returns the current gamma values.

```
VideoDigitizerError VDGetInputGammaValue (
   VideoDigitizerComponent ci,
   Fixed *channel1,
   Fixed *channel2,
   Fixed *channel3
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

channel1

A pointer to a fixed integer field that is to receive the gamma value for the red component of the input video signal.

channel2

A pointer to a fixed integer field that is to receive the gamma value for the green component of the input video signal.

channel3

A pointer to a fixed integer field that is to receive the gamma value for the blue component of the input video signal.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetInputName

Gets the name of a video input.

```
VideoDigitizerError VDGetInputName (
   VideoDigitizerComponent ci,
   long videoInput,
   Str255 name
);
```

Parameters

ci

Specifies the video digitizer component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

videoInput

The input video source for this request. Video digitizer components number video sources sequentially, starting at 0. So, to request information about the first video source, an application sets this parameter to 0. Applications can get the number of video sources supported by a video digitizer component by calling `VDGetNumberOfInputs` (page 53).

name

The video input source's name string.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDGetKeyColor

Obtains the index value of the active key color.

```
VideoDigitizerError VDGetKeyColor (
   VideoDigitizerComponent ci,
   long *index
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

index

A pointer to a field that is to receive the index of the key color. This index value identifies the key color within the currently active color lookup table. If there are several active key colors, the video digitizer returns the first color from the key color list. Subsequently, applications use `VDGetNextKeyColor` (page 52) to obtain other colors from the list. If there is no active key color, the function sets the field to -1.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Special Considerations

All video digitizer components that support key colors must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDGetKeyColorRange

Obtains the currently defined key color range.

```
VideoDigitizerError VDGetKeyColorRange (
   VideoDigitizerComponent ci,
   RGBColor *minRGB,
   RGBColor *maxRGB
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from OpenComponent or OpenDefaultComponent.

minRGB

A pointer to a field that is to receive the lower bound of the key color range. The video digitizer component places the RGBColor structure that corresponds to the lower end of the range in the field referred to by this parameter.

maxRGB

A pointer to a field that is to receive the upper bound of the key color range. The video digitizer component places the RGBColor structure that corresponds to the upper end of the range in the field referred to by this parameter.

Return Value
See Error Codes. Returns noErr if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDGetMaskandValue

Obtains the appropriate alpha channel or blend mask value for a desired level of video blending.

```
VideoDigitizerError VDGetMaskandValue (
   VideoDigitizerComponent ci,
   unsigned short blendLevel,
   long *mask,
   long *value
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from OpenComponent or OpenDefaultComponent.

blendLevel

The desired blend level. Valid values range from 0 to 65,535, where 0 corresponds to no video and 65,535 corresponds to all video.

mask

A pointer to a field that is to receive a value indicating which bits are meaningful in the data returned for the `value` parameter. The video digitizer component sets to 1 the bits that correspond to meaningful bits in the data returned for the `value` parameter.

value

A pointer to a field that is to receive data that can be used to obtain the desired blend level. The data returned for the `mask` parameter indicates which bits are valid in the data returned for this parameter.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
The information returned by the digitizer component differs based on the type of blending supported by the component. In all cases, however, the returned value of the `value` parameter contains the value for the desired blend level, and the returned value of the `mask` parameter indicates which bits in the `value` parameter are meaningful. Bits in the returned mask parameter value that are set to 1 correspond to meaningful bits in the returned value parameter value.

For example, if an application requests a 50 percent video blend level from a digitizer that supports 8-bit alpha channels, the digitizer component might return 0xFF000000 in the `mask` parameter, identifying a full upper byte as the alpha channel, and 0x80000000 in the `value` parameter, specifying a 50 percent blend level.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetMaskPixMap

Retrieves the pixel map data for a component's blend mask.

```
VideoDigitizerError VDGetMaskPixMap (
   VideoDigitizerComponent ci,
   PixMapHandle maskPixMap
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

maskPixMap

A handle to a `PixMap` structure. The video digitizer component returns the pixel map data for its blend mask into the `PixMap` structure specified by this parameter. The video digitizer component resizes the handle as appropriate. Your application is responsible for disposing of this handle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
This function is supported only by digitizer components that support blend masks.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetMaxAuxBuffer

Obtains access to buffers that are located on special hardware.

```
VideoDigitizerError VDGetMaxAuxBuffer (
   VideoDigitizerComponent ci,
   PixMapHandle *pm,
   Rect *r
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

pm

A pointer to a handle to a `PixMap` structure. The video digitizer component returns a handle to the destination `PixMap` structure in the field referred to by this parameter. Do not dispose of this structure. If the digitizer component cannot allocate a buffer, this handle is set to `NIL`.

r

A pointer to a `Rect` structure. The video digitizer component places the coordinates of the largest output rectangle it can support into the structure referred to by this parameter.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetMaxSrcRect

Returns the maximum source rectangle.

```
VideoDigitizerError VDGetMaxSrcRect (
   VideoDigitizerComponent ci,
   short inputStd,
   Rect *maxSrcRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputStd

A short integer that specifies the input video signal associated with this maximum source rectangle.

maxSrcRect

A pointer to a `Rect` structure that is to receive the size and location information for the maximum source rectangle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetNextKeyColor

Obtains the index value of the active key colors in cases where the digitizer component supports multiple key colors.

```
VideoDigitizerError VDGetNextKeyColor (
   VideoDigitizerComponent ci,
   long index
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

index

A field that is to receive the index of the next key color. This index value identifies the key color within the currently active color lookup table. If there are no more colors left in the list, the digitizer component sets the field referred to by the `index` parameter to -1.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components that support multiple key colors must support this function

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetNumberOfInputs

Returns the number of input video sources that a video digitizer component supports.

```
VideoDigitizerError VDGetNumberOfInputs (
   VideoDigitizerComponent ci,
   short *inputs
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputs

A pointer to an integer that is to receive the number of input video sources supported by the specified component. Video digitizer components number video sources sequentially, starting at 0. So, if a digitizer component supports two inputs, this function sets the field referred to by the `inputs` parameter to 1.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDGetPlayThruDestination

Obtains information about the current video destination.

```
VideoDigitizerError VDGetPlayThruDestination (
   VideoDigitizerComponent ci,
   PixMapHandle *dest,
   Rect *destRect,
   MatrixRecord *m,
   RgnHandle *mask
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

dest

A pointer to a handle to a `PixMap` structure. The video digitizer component returns a handle to the destination `PixMap` structure in the field referred to by this parameter. It is the caller's responsibility to dispose of the `PixMap` structure.

destRect

A pointer to a `Rect` structure. The video digitizer component places the coordinates of the output rectangle into the structure referred to by this parameter. If there is no output rectangle defined, the component returns an empty rectangle.

m

A pointer to a `MatrixRecord` structure. The video digitizer component places the transformation matrix into the structure referred to by this parameter.

mask

A pointer to a handle to a `MacRegion` structure. The video digitizer component places a handle to the mask region into the field referred to by this parameter. Applications can use masks to control the video into the destination rectangle. If there is no mask region defined, the digitizer component sets this returned handle to `NIL`. The caller is responsible for disposing of the `MacRegion` structure.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetPLLFilterType

Determines which phase locked loop (PLL) mode is currently active for a video digitizer.

```
VideoDigitizerError VDGetPLLFilterType (
   VideoDigitizerComponent ci,
   short *pllType
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

pllType

Points to a field that is to receive a value indicating which PLL mode is active. Values are 0 for broadcast mode and 1 for videotape recorder mode.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetPreferredImageDimensions

Gets the preferred image dimensions for a video digitizer.

```
VideoDigitizerError VDGetPreferredImageDimensions (
   VideoDigitizerComponent ci,
   long *width,
   long *height
);
```

Parameters

ci

Specifies the video digitizer component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

width

A pointer to the preferred image width.

height

A pointer to the preferred image height.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDGetPreferredTimeScale

Determines a digitizer's preferred time scale.

```
VideoDigitizerError VDGetPreferredTimeScale (
   VideoDigitizerComponent ci,
   TimeScale *preferred
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

preferred

A pointer to a time scale. The video digitizer returns information about its preferred time scale in this structure.

Return Value
If the digitizer does not have a preferred time scale, it returns a result code of `digiUnimpErr`. See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDGetSaturation

Returns the current saturation value.

```
VideoDigitizerError VDGetSaturation (
   VideoDigitizerComponent ci,
   unsigned short *saturation
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

saturation

A pointer to an integer that is to receive the current saturation value. The saturation value controls color intensity. For example, at high saturation levels, red appears to be red; at low saturation, red appears as pink. Valid saturation values range from 0 to 65,535, where 0 is the minimum saturation value and 65,535 specifies maximum saturation.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetSharpness

Returns the current sharpness value.

```
VideoDigitizerError VDGetSharpness (
   VideoDigitizerComponent ci,
   unsigned short *sharpness
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

sharpness

A pointer to an integer that is to receive the current sharpness value. The sharpness value ranges from 0 to 65,535, where 0 represents no sharpness filtering and 65,535 represents full sharpness filtering. Higher values result in a visual impression of increased picture sharpness.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetSoundInputDriver

Retrieves information about a video digitizer's sound input driver.

```
VideoDigitizerError VDGetSoundInputDriver (
   VideoDigitizerComponent ci,
   Str255 soundDriverName
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

soundDriverName

A pointer to a string. The video digitizer returns the name of its sound input driver. If the digitizer does not have an associated driver, it returns a result code of `digiUnimpErr`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDGetSoundInputSource

Instructs your video digitizer component to return the sound input source associated with a particular video input.

```
VideoDigitizerError VDGetSoundInputSource (
   VideoDigitizerComponent ci,
   long videoInput,
   long *soundInput
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

videoInput

The input video source for this request. Video digitizer components number video sources sequentially, starting at 0. So, to request information about the first video source, an application sets this parameter to 0. Applications can get the number of video sources supported by a video digitizer component by calling `VDGetNumberOfInputs` (page 53).

soundInput

The sound input index to use with the sound input driver returned by `VDGetSoundInputDriver` (page 57).

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Some video digitizers may associate different sound inputs with each video input. `VDGetSoundInputDriver` (page 57) returns the name of the sound input driver that the sound input is associated with.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetTimeCode

Instructs your video digitizer component to return timecode information for the incoming video signal.

```
VideoDigitizerError VDGetTimeCode (
   VideoDigitizerComponent ci,
   TimeRecord *atTime,
   void *timeCodeFormat,
   void *timeCodeTime
);
```

Parameters

ci
The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

atTime
A pointer to a `TimeRecord` structure to receive the QuickTime movie time value corresponding to the timecode information.

timeCodeFormat
A pointer to a `TimeCodeDef` structure. Your video digitizer component returns the movie's timecode definition information in this structure.

timeCodeTime
A pointer to a `TimeCodeRecord` structure. Your video digitizer component returns the time value corresponding to the movie time contained in this structure.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Typically, this function is called once, at the beginning of a capture session. The use of this function assumes that the timecoding for the entire capture session will be continuous.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetUniqueIDs

Returns a unique identifier for a particular video digitizer device.

```
VideoDigitizerError VDGetUniqueIDs (
   VideoDigitizerComponent ci,
   UInt64 *outDeviceID,
   UInt64 *outInputID
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

outDeviceID

A pointer to a 64-bit hardware device ID. In the case of a FireWire device, this is the FireWire ID.

outInputID

A pointer to a 64-bit hardware input ID. A return of 0 means you don't have one.

Return Value

An error return of type `ComponentResult`. See `Error Codes`. Returns `noErr` if there is no error.

Discussion

This function is provided so the VDIG can give the sequence grabber information that helps it choose a particular device and input from those available. You might use it, for example, to restore a specific camera from a set of several hot-plugged FireWire cameras. The caller can pass `NIL` if it is not interested in one of the IDs.

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`QuickTimeComponents.h`

VDGetVBlankRect

Returns the vertical blanking rectangle.

```
VideoDigitizerError VDGetVBlankRect (
   VideoDigitizerComponent ci,
   short inputStd,
   Rect *vBlankRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputStd

A short integer (see below) that identifies the signaling standard used in the source video signal. See these constants:

```
ntscIn
palIn
secamIn
```

vBlankRect

A pointer to a `Rect` structure that is to receive the size and location information for the vertical blanking rectangle.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetVideoDefaults

Returns the recommended values for many of the analog video parameters that may be set by applications.

```
VideoDigitizerError VDGetVideoDefaults (
   VideoDigitizerComponent ci,
   unsigned short *blackLevel,
   unsigned short *whiteLevel,
   unsigned short *brightness,
   unsigned short *hue,
   unsigned short *saturation,
   unsigned short *contrast,
   unsigned short *sharpness
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

blackLevel

A pointer to an integer that is to receive the default black level value. Black level values range from 0 to 65,535, where 0 represents the maximum black value and 65,535 represents the minimum black value.

whiteLevel

A pointer to an integer that is to receive the default white level value. White level values range from 0 to 65,535, where 0 represents the minimum white value and 65,535 represents the maximum white value.

brightness

A pointer to an integer that is to receive the default brightness value. Brightness values range from 0 to 65,535, where 0 is the darkest possible setting and 65,535 is the lightest possible setting.

hue

A pointer to an integer that is to receive the default hue value. Hue is similar to the tint control on a television, and it is specified in degrees with complementary colors set 180 degrees apart (red is 0 degrees, green is +120 degrees, and blue is -120 degrees). Video digitizer components support hue values that range from 0 (-180 degrees shift in hue) to 65,535 (+179 degrees shift in hue), where 32,767 represents a 0 degree shift in hue.

saturation

A pointer to an integer that is to receive the default saturation value. The saturation value controls color intensity. For example, at high saturation levels, red appears to be red; at low saturation, red appears as pink. Valid saturation values range from 0 to 65,535, where 0 is the minimum saturation value and 65,535 specifies maximum saturation.

contrast

A pointer to an integer that is to receive the default contrast value. The contrast value ranges from 0 to 65,535, where 0 represents no change to the basic image and larger values increase the contrast of the video image (they increase the slope of the transform).

sharpness

A pointer to an integer that is to receive the default sharpness value. The sharpness value ranges from 0 to 65,535, where 0 represents no sharpness filtering and 65,535 represents full sharpness filtering. Higher values result in a visual impression of increased picture sharpness.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGetWhiteLevelValue

Returns the current white level value.

```
VideoDigitizerError VDGetWhiteLevelValue (
   VideoDigitizerComponent ci,
   unsigned short *whiteLevel
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

whiteLevel

A pointer to an integer that is to receive the current white level value. White level values range from 0 to 65,535, where 0 represents the minimum white value and 65,535 represents the maximum white value.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGrabOneFrame

Instructs the video digitizer component to digitize a single frame of source video.

```
VideoDigitizerError VDGrabOneFrame (
   VideoDigitizerComponent ci
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
If the specified digitizer component is already digitizing continuously when the application calls this function, the digitizer component returns the next digitized frame and then stops. If the digitizer component is stopped, the component digitizes a single frame and then stops. To resume continuous digitization, applications should call `VDSetPlayThruOnOff` (page 89).

Special Considerations

This function supports synchronous single-frame video digitization; that is, the digitizer component does not return control to your application until it has successfully processed the next video frame. Some video digitizer components may also support asynchronous single-frame digitization. Applications can request asynchronous digitization by calling `VDGrabOneFrameAsync` (page 64).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGrabOneFrameAsync

Instructs the video digitizer component to start to digitize asynchronously a single frame of source video.

```
VideoDigitizerError VDGrabOneFrameAsync (
   VideoDigitizerComponent ci,
   short buffer
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

buffer

Identifies the next output buffer. The value of this parameter must correspond to a valid index into the list of buffers that you supply when your application calls `VDSetupBuffers` (page 93). Note that this value is zero-based (that is, you must set this parameter to 0 to refer to the first buffer in the buffer list).

Return Value

See `Error Codes`. Returns `noErr` if there is no error.

Discussion

When calling this function, the application specifies the next destination video buffer, allowing the digitizer component to quickly switch from the current buffer to the next buffer. In this manner, your application's ability to grab video at high frame rates is enhanced. If the specified digitizer component is already digitizing continuously when the application calls this function, the digitizer component returns the next digitized frame and then stops. If the digitizer component is stopped, the component digitizes a single frame and then stops. To resume continuous digitization, applications should call `VDSetPlayThruOnOff` (page 89).

This function also allows applications to use more than one destination buffer for the digitized video. The application defines these buffers by calling `VDSetupBuffers` (page 93). The application specifies one of these destination buffers for the digitized frame when it calls `VDSetPlayThruDestination` (page 88) or `VDSetPlayThruGlobalRect` (page 89).

Special Considerations

Applications can determine whether a video digitizer component supports asynchronous frame grabbing by using `VDGetCurrentFlags` (page 37) to retrieve the digitizer component's output capability flags. If the `digiOutDoesAsyncGrabs` flag is set to 1, the digitizer component supports both this function and `VDDone` (page 31). If a video digitizer component does not support asynchronous digitization, applications must use `VDGrabOneFrame` (page 63) to perform single-frame digitization.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

VDIIDCGetCSRData

Reads a camera's CSR registers directly.

```
VideoDigitizerError VDIIDCGetCSRData (
   VideoDigitizerComponent ci,
   Boolean offsetFromUnitBase,
   UInt32 offset,
   UInt32 *data
);
```

Parameters

ci

The component instance that identifies your connection to a video digitizer component. The digitizer's subtype must be `vdSubtypeIIDC` (`'iidc'`).

offsetFromUnitBase

Pass TRUE if the offset is relative to the initial unit space (FFFF Fxxx xxxx), FALSE if the offset is relative to the initial register space (FFFF F000 0000).

offset

Offset in bytes of the value to read.

data

Location to store the `value` (of type UInt32) that was read.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion
You might want to read a camera's registers directly if you're querying the state of a feature not accessed by `VDIIDCGetFeatures` or if some camera-specific information must be accessed.

Version Notes
Introduced in QuickTime 6.4.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QuickTimeComponents.h`

VDIIDCGetDefaultFeatures

Places atoms in a QuickTime atom container that specify the default capabilities and default state of a camera's IIDC features.

```
VideoDigitizerError VDIIDCGetDefaultFeatures (
   VideoDigitizerComponent ci,
   QTAtomContainer *container
);
```

Parameters

ci

The component instance that identifies your connection to a video digitizer component. The digitizer's subtype must be `vdSubtypeIIDC` (`'iidc'`).

container

Upon return, a pointer to a QuickTime atom container containing atoms of type `vdIIDCAtomTypeFeature` for each IIDC camera feature whose default is known. The container may be empty if defaults cannot be determined.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion
The digitizer will create the QuickTime atom container, and it is the responsibility of the client to delete it if the routine does not return an error.

Version Notes
Introduced in QuickTime 6.4.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QuickTimeComponents.h`

VDIIDCGetFeatures

Places atoms in a QuickTime atom container that specify the current capabilities of a camera and the state of its IIDC features.

```
VideoDigitizerError VDIIDCGetFeatures (
   VideoDigitizerComponent ci,
   QTAtomContainer *container
);
```

Parameters

ci

The component instance that identifies your connection to a video digitizer component. The digitizer's subtype must be `vdSubtypeIIDC` (`'iidc'`).

container

Upon return, a pointer to a QuickTime atom container containing atoms of type `vdIIDCAtomTypeFeature` for each IIDC camera feature. If the camera has not implemented any IIDC features the container returns empty.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion
The digitizer creates the container, and it is the responsibility of the client to ultimately delete it if the routine does not return an error. Since the values that this function retrieves might change underneath the client, they should not be cached but should be retrieved each time they are needed.

Version Notes
Introduced in QuickTime 6.4.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

VDIIDCGetFeaturesForSpecifier

Places atoms in a QuickTime atom container that specify the current state of a single camera IIDC feature or group of features.

```
VideoDigitizerError VDIIDCGetFeaturesForSpecifier (
   VideoDigitizerComponent ci,
   OSType specifier,
   QTAtomContainer *container
);
```

Parameters

ci

The component instance that identifies your connection to a video digitizer component. The digitizer's subtype must be `vdSubtypeIIDC` (`'iidc'`).

specifier

The feature or group of features to be retrieved: // IIDC feature types `vdIIDCFeatureHue` = `'hue'`, `vdIIDCFeatureSaturation` = `'satu'`, `vdIIDCFeatureSharpness` = `'shrp'`, `vdIIDCFeatureBrightness` = `'brit'`, `vdIIDCFeatureGain` = `'gain'`, `vdIIDCFeatureIris` = `'iris'`, `vdIIDCFeatureShutter` = `'shtr'`, `vdIIDCFeatureExposure` = `'xpsr'`, `vdIIDCFeatureWhiteBalanceU` = `'whbu'`, `vdIIDCFeatureWhiteBalanceV` = `'whbv'`, `vdIIDCFeatureGamma` = `'gmma'`, `vdIIDCFeatureTemperature` = `'temp'`, `vdIIDCFeatureZoom` = `'zoom'`, `vdIIDCFeatureFocus` = `'fcus'`, `vdIIDCFeaturePan` = `'pan'`, `vdIIDCFeatureTilt` = `'tilt'`, `vdIIDCFeatureOpticalFilter` = `'opft'`, `vdIIDCFeatureTrigger` = `'trgr'`, `vdIIDCFeatureCaptureSize` = `'cpsz'`, `vdIIDCFeatureCaptureQuality` = `'cpql'`, `vdIIDCFeatureFocusPoint` = `'fpnt'`, `vdIIDCFeatureEdgeEnhancement` = `'eden'` `vdIIDCFeatureLightingHint` = `'lhnt'` // IIDC group types `vdIIDCGroupImage` = `'imag'`, `vdIIDCGroupColor` = `'colr'`, `vdIIDCGroupMechanics` = `'mech'`, `vdIIDCGroupTrigger` = `'trig'`

container

Upon return, a pointer to a QuickTime atom container containing atoms of type `vdIIDCAtomTypeFeature` for each IIDC camera feature corresponding to the specifier. If the camera has not implemented any of the specified features the container returns empty.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion

The digitizer creates the container, and it is the responsibility of the client to ultimately delete it if the routine does not return an error. Since the values that this function retrieves might change underneath the client, they should not be cached but should be retrieved each time they are needed.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`QuickTimeComponents.h`

VDIIDCSetCSRData

Writes to a camera's CSR registers directly.

```
VideoDigitizerError VDIIDCSetCSRData (
   VideoDigitizerComponent ci,
   Boolean offsetFromUnitBase,
   UInt32 offset,
   UInt32 data
);
```

Parameters

ci

The component instance that identifies your connection to a video digitizer component. The digitizer's subtype must be `vdSubtypeIIDC` (`'iidc'`).

offsetFromUnitBase

Pass TRUE if the offset is relative to the initial unit space (FFFF Fxxx xxxx), FALSE if the offset is relative to the initial register space (FFFF F000 0000).

offset

Offset in bytes of the value to set.

data

Location of the `value` (of type UInt32) to write.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion
You might want to write to a camera's registers directly if you're setting the state of a feature not accessed by `VDIIDCSetFeatures` or if some camera-specific information must be set.

Version Notes
Introduced in QuickTime 6.4.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QuickTimeComponents.h`

VDIIDCSetFeatures

Changes the state of a camera's IIDC features.

```
VideoDigitizerError VDIIDCSetFeatures (
   VideoDigitizerComponent ci,
   QTAtomContainer container
);
```

Parameters

ci

The component instance that identifies your connection to a video digitizer component. The digitizer's subtype must be `vdSubtypeIIDC` (`'iidc'`).

container

A pointer to a QuickTime atom container populated with atoms of type `vdIIDCAtomTypeFeature`; the container may have one or many atoms in it. An empty container will cause the function to have no effect.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Discussion
It is the responsibility of the client to provide the QuickTime atom container and delete it after use.

Version Notes
Introduced in QuickTime 6.4.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QuickTimeComponents.h`

VDPreflightDestination

Verifies that a video digitizer component can support a set of destination settings intended for use with VDSetPlayThruDestination.

```
VideoDigitizerError VDPreflightDestination (
   VideoDigitizerComponent ci,
   Rect *digitizerRect,
   PixMap **dest,
   RectPtr destRect,
   MatrixRecordPtr m
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

digitizerRect

A pointer to a `Rect` structure that contains the size and location information for the digitizer rectangle. The coordinates of this rectangle must be relative to the maximum source rectangle. In addition, the digitizer rectangle must be within the maximum source rectangle. For a discussion of the relationship between these rectangles, see "Video Digitizer Components" in *Inside Macintosh: QuickTime* Components. If the video digitizer component cannot accommodate the specified rectangle, it changes the coordinates in this structure to specify a rectangle that it can support and sets the result to `qtParamErr`.

dest

A handle to the destination `PixMap` structure.

destRect

A pointer to a `Rect` structure that specifies the size and location of the video destination. This is an optional parameter. Applications may specify a transformation matrix to control the placement and scaling of the video image in the destination `PixMap` structure. In this case, the `destRect` parameter is set to `NIL` and the `m` parameter specifies the matrix. The destination rectangle must be in the coordinate system of the destination `PixMap` structure specified by the `dest` parameter. If the video digitizer component cannot accommodate this rectangle, it changes the coordinates in the structure to specify a rectangle that it can support and sets the result to `qtParamErr`.

m

A pointer to a `MatrixRecord` structure containing the transformation matrix for the destination video image. This is an optional parameter. Applications may specify a destination rectangle to control the placement and scaling of the video image in the destination `PixMap` structure. In this case, the `m` parameter is set to `NIL` and the `destRect` parameter specifies the destination rectangle. If the `destRect` parameter is `NIL`, you can determine the destination rectangle for simple matrices by calling `TransformRect` using the current digitizer rectangle and this matrix. If the video digitizer component cannot accommodate this matrix, it changes the values in the structure to define a matrix that it can support and sets the result to `qtParamErr`. Applications can determine the capabilities of a video digitizer component by calling `VDGetDigitizerInfo` (page 39).

Return Value
The application provides the desired settings as parameters to this function. The video digitizer component then examines those settings. If the digitizer component can support the specified settings, it sets the result code to `noErr`. If the digitizer component cannot support the settings, it alters the input settings to reflect values that it can support and returns a result code of `qtParamErr`. See `Error Codes`.

Discussion
All video digitizer components must support this function. Applications should use this function to test destination settings whenever a video digitizer component cannot support arbitrary scaling.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDPreflightGlobalRect

Verifies that a video digitizer component can support a set of destination settings intended for use with VDSetPlayThruGlobalRect.

```
VideoDigitizerError VDPreflightGlobalRect (
   VideoDigitizerComponent ci,
   GrafPtr theWindow,
   Rect *globalRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

theWindow

A pointer to the destination window.

globalRect

A pointer to a `Rect` structure that specifies the size and location of the video destination. This rectangle must be in the coordinate system of the destination window specified by the `theWindow` parameter. If the video digitizer component cannot accommodate this rectangle, it changes the coordinates in the structure to specify a rectangle that it can support and sets the result to `qtParamErr`.

Return Value
Returns `qtParamErr` if the video digitizer component cannot accommodate the destination rectangle. Returns `digiUnimpErr` if the video digitizer component does not support placing destination video into a rectangle that crosses screens. See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Applications should use this function to determine whether a video digitizer supports placing destination video into a rectangle that crosses screens. Digitizers that do not support this capability return a result of `digiUnimpErr`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDReleaseAsyncBuffers

Releases the buffers that were allocated with VDSetupBuffers.

```
VideoDigitizerError VDReleaseAsyncBuffers (
   VideoDigitizerComponent ci
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDReleaseCompressBuffer

Frees a buffer received from VDCompressDone.

```
VideoDigitizerError VDReleaseCompressBuffer (
   VideoDigitizerComponent ci,
   Ptr bufferAddr
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

bufferAddr

Points to the location of the buffer to be released. This address must correspond to a buffer address that the application obtained from `VDCompressDone` (page 29).

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDResetCompressSequence

Forces the video digitizer to insert a key frame into a temporally compressed image sequence.

```
VideoDigitizerError VDResetCompressSequence (
   VideoDigitizerComponent ci
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSelectUniqueIDs

Selects a video digitizer device by ID.

```
VideoDigitizerError VDSelectUniqueIDs (
   VideoDigitizerComponent ci,
   const UInt64 *inDeviceID,
   const UInt64 *inInputID
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inDeviceID

A pointer to a unique 64-bit hardware device ID.

inInputID

A pointer to a unique 64-bit hardware input ID.

Return Value
An error return of type `ComponentResult`. See `Error Codes`. Returns `vdDontHaveThatUniqueIDErr` if your device doesn't have a match. Returns `noErr` if there is no error.

Discussion
Note this function does selection, not setting. The assumption is that the unique ID is set by the hardware and is not modifiable by the calling application. Passing either a `NIL` pointer or 0 for an ID means you don't care. This should restore the device and input IDs returned by `VDGetUniqueIDs` (page 60).

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`QuickTimeComponents.h`

VDSetBlackLevelValue

Sets the current black level value.

```
VideoDigitizerError VDSetBlackLevelValue (
   VideoDigitizerComponent ci,
   unsigned short *blackLevel
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

blackLevel

A pointer to an integer that contains the new black level value. Black level values range from 0 to 65,535, where 0 represents the maximum black value and 65,535 represents the minimum black value. The digitizer component returns the new value, so that the application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
ExampleVideoPanel
ExampleVideoPanel.win

Declared In
`QuickTimeComponents.h`

VDSetBrightness

Sets the current brightness value.

```
VideoDigitizerError VDSetBrightness (
   VideoDigitizerComponent ci,
   unsigned short *brightness
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

brightness

A pointer to an integer that contains the new brightness value. Brightness values range from 0 to 65,535, where 0 is the darkest possible setting and 65,535 is the lightest possible setting. The digitizer component returns the new value, so that the application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetClipRgn

Defines a clipping region for a video digitizer.

```
VideoDigitizerError VDSetClipRgn (
   VideoDigitizerComponent ci,
   RgnHandle clipRegion
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

clipRegion

A handle to a `MacRegion` structure that defines the clipping region.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetClipState

Controls whether clipping is enabled.

```
VideoDigitizerError VDSetClipState (
   VideoDigitizerComponent ci,
   short clipEnable
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

clipEnable

Controls whether clipping is enabled. Place 0 into the short integer if clipping is disabled, and 1 if it is enabled.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetCompression

Specifies certain compression parameters.

```
VideoDigitizerError VDSetCompression (
   VideoDigitizerComponent ci,
   OSType compressType,
   short depth,
   Rect *bounds,
   CodecQ spatialQuality,
   CodecQ temporalQuality,
   long keyFrameRate
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

compressType

A compressor type. This value corresponds to the `component` subtype of the compressor component; see `Codec Identifiers`.

depth

The depth at which the image is likely to be viewed. Compressors may use this as an indication of the color or grayscale resolution of the image. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 correspond to 1-bit, 2-bit, 4-bit, and 8-bit grayscale images.

bounds

A pointer to a `Rect` structure that defines the desired boundaries of the compressed image.

spatialQuality

A constant (see below) that indicates the desired image quality for each frame in the sequence. See these constants:

```
codecMinQuality
codecLowQuality
codecNormalQuality
codecHighQuality
codecMaxQuality
codecLosslessQuality
```

temporalQuality

A constant (see below) that indicates the desired temporal quality for the sequence as a whole.

keyFrameRate

The maximum number of frames to allow between key frames. This value defines the minimum rate at which key frames are to appear in the compressed sequence; however, the video digitizer may insert key frames more often than an application specifies. If the application requests no temporal compression (that is, the application set the `temporalQuality` parameter to 0), the video digitizer ignores this parameter.

Return Value

See `Error Codes`. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetCompressionOnOff

Allows an application to start and stop compression by video digitizers that can deliver either compressed or uncompressed image data.

```
VideoDigitizerError VDSetCompressionOnOff (
   VideoDigitizerComponent ci,
   Boolean state
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

state

A Boolean value that indicates whether to enable or disable compression. Applications set this parameter to TRUE to enable compression. Setting it to FALSE disables compression.

Return Value
Digitizers that only provide compressed data have their `digiOutDoesCompressOnly` flag set to 1, rather than 0. These digitizers may either ignore this function or return a nonzero result code. See `Error Codes`. Return `noErr` if there is no error.

Discussion
Applications must call this function before they call either `VDSetCompression` (page 76) or `VDCompressOneFrameAsync` (page 30). This allows the video digitizer to prepare for the operation.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetContrast

Sets the current contrast value.

```
VideoDigitizerError VDSetContrast (
   VideoDigitizerComponent ci,
   unsigned short *contrast
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

contrast

A pointer to an integer that contains the new contrast value. The contrast value ranges from 0 to 65,535, where 0 represents no change to the basic image and larger values increase the contrast of the video image (they increase the slope of the transform). The digitizer component returns the new value, so that the application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetDataRate

Instructs your video digitizer component to limit the rate at which it delivers compressed, digitized video data.

```
VideoDigitizerError VDSetDataRate (
   VideoDigitizerComponent ci,
   long bytesPerSecond
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

bytesPerSecond

The maximum data rate requested by the application, in bytes per second. This parameter is set to 0 to remove any data-rate restrictions.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
This function is valid only for video digitizer components that can deliver compressed video; that is, components that support the `VDCompressOneFrameAsync` (page 30) function. Components that support data-rate limiting set the `codecInfoDoesRateConstrain` flag to 1 in the `compressFlags` field of the

`VDCompressionList` structure returned by the component in response to the `VDGetCompressionTypes` (page 35) function. Your video digitizer component should return this data-rate limit in the `bytesPerSecond` parameter of the existing `VDGetDataRate` (page 38) function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetDestinationPort

Sets the destination port for a video digitizer.

```
VideoDigitizerError VDSetDestinationPort (
   VideoDigitizerComponent ci,
   CGrafPtr destPort
);
```

Parameters

ci

Specifies the video digitizer component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

destPort

A pointer to a `CGrafPort` structure.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetDigitizerRect

Sets the current video digitizer rectangle.

```
VideoDigitizerError VDSetDigitizerRect (
   VideoDigitizerComponent ci,
   Rect *digitizerRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

digitizerRect

A pointer to a `Rect` structure that contains the size and location information for the digitizer rectangle. The coordinates of this rectangle must be relative to the maximum source rectangle. In addition, the digitizer rectangle must be within the maximum source rectangle. For a discussion of the relationship between these rectangles, see "Video Digitizer Components" in *Inside Macintosh: QuickTime* Components.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetDigitizerUserInterrupt

Sets custom interrupt functions.

```
VideoDigitizerError VDSetDigitizerUserInterrupt (
   VideoDigitizerComponent ci,
   long flags,
   VdigIntUPP userInterruptProc,
   long refcon
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

flags

Indicates when the interrupt function is to be called. If bit 0 is set to 1, the video digitizer component calls the custom interrupt procedure each time it starts to display an even-line field. If bit 1 is set to 1, the video digitizer component calls the custom interrupt procedure each time it starts to display an odd-line field. Applications may set both bits to 1.

userInterruptProc

A Universal Procedure Pointer to a `VdigIntProc` callback. Applications can set this parameter to `NIL` to remove a `VdigIntProc` callback.

refcon

Contains parameter data that is appropriate for the `callback`. Use this parameter to point to a data structure containing any information your callback needs.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetFieldPreference

Specifies which field to use in cases where the vertical scaling is less than half size.

```
VideoDigitizerError VDSetFieldPreference (
   VideoDigitizerComponent ci,
   short fieldFlag
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

fieldFlag

A constant (see below) that indicates which field to use. See these constants:

`vdUseAnyField`

`vdUseOddField`

`vdUseEvenField`

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetFrameRate

Indicates an application's desired frame rate to the video digitizer.

```
VideoDigitizerError VDSetFrameRate (
   VideoDigitizerComponent ci,
   Fixed framesPerSecond
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

framesPerSecond

The application's desired frame rate. Applications may set this parameter to 0 to return the digitizer to its default frame rate (typically 29.97 frames per second).

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetHue

Sets the current hue value.

```
VideoDigitizerError VDSetHue (
   VideoDigitizerComponent ci,
   unsigned short *hue
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

hue

A pointer to an integer that contains the new hue value. Hue is similar to the tint control on a television, and it is specified in degrees with complementary colors set 180 degrees apart (red is 0 degrees, green is +120 degrees, and blue is -120 degrees). Video digitizer components support hue values that range from 0 (-180 degrees shift in hue) to 65,535 (+179 degrees shift in hue), where 32,767 represents a 0 degree shift in hue. The digitizer component returns the new value, so that the application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetInput

Selects the input video source for a video digitizer component.

```
VideoDigitizerError VDSetInput (
   VideoDigitizerComponent ci,
   short input
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

input

The input video source for this request. Video digitizer components number video sources sequentially, starting at 0. To request the first video source, an application sets this parameter to 0.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetInputColorSpaceMode

Chooses between color and grayscale digitized video.

```
VideoDigitizerError VDSetInputColorSpaceMode (
   VideoDigitizerComponent ci,
   short colorSpaceMode
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

colorSpaceMode

Controls color digitization. Set to 0 for grayscale, 1 for color.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetInputGammaRecord

Changes the active input gamma data structure.

```
VideoDigitizerError VDSetInputGammaRecord (
   VideoDigitizerComponent ci,
   VDGamRecPtr inputGammaPtr
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputGammaPtr

A `VDGammaRecord` structure.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetInputGammaValue

Sets the gamma values.

```
VideoDigitizerError VDSetInputGammaValue (
   VideoDigitizerComponent ci,
   Fixed channel1,
   Fixed channel2,
   Fixed channel3
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from the Component Manager's `OpenComponent` function.

channel1

The gamma value for the red component of the input video signal.

channel2

The gamma value for the green component of the input video signal.

channel3

The gamma value for the blue component of the input video signal.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
These gamma values control the brightness of the input video signal. Your application can implement special color effects, such as turning off specific color channels, by calling this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetInputStandard

Specifies the input signaling standard to digitize.

```
VideoDigitizerError VDSetInputStandard (
   VideoDigitizerComponent ci,
   short inputStandard
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

inputStandard

A short integer (see below) that identifies the input signaling standard. See these constants:

`ntscIn`

`palIn`

`secamIn`

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetKeyColor

Sets the key color for video digitizing.

```
VideoDigitizerError VDSetKeyColor (
   VideoDigitizerComponent ci,
   long index
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

index

The new key color. This value must correspond to a color in the current color lookup table.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components that support key colors must support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetKeyColorRange

Defines a key color range for video digitizing.

```
VideoDigitizerError VDSetKeyColorRange (
   VideoDigitizerComponent ci,
   RGBColor *minRGB,
   RGBColor *maxRGB
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

minRGB

A pointer to a field that contains the lower bound of the key color range. All colors in the color table between the color specified by the `minRGB` parameter and the color specified by the `maxRGB` parameter are considered key colors.

maxRGB

A pointer to a field that contains the upper bound of the key color range. All colors in the color table between the color specified by the `minRGB` parameter and the color specified by the `maxRGB` parameter are considered key colors.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetMasterBlendLevel

Sets the blend level value for the input video signal.

```
VideoDigitizerError VDSetMasterBlendLevel (
   VideoDigitizerComponent ci,
   unsigned short *blendLevel
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

blendLevel

A pointer to a field that specifies the new master blend level. Valid values range from 0 to 65,535, where 0 corresponds to no video and 65,535 corresponds to all video. The digitizer component returns the new value in this field, so your application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetPlayThruDestination

Establishes the destination settings for a video digitizer component.

```
VideoDigitizerError VDSetPlayThruDestination (
   VideoDigitizerComponent ci,
   PixMapHandle dest,
   RectPtr destRect,
   MatrixRecordPtr m,
   RgnHandle mask
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

dest

A handle to the destination `PixMap` structure. This pixel map may be in the video frame buffer of the Macintosh computer, or it may specify an offscreen buffer.

destRect

A pointer to a `Rect` structure that specifies the size and location of the video destination. This rectangle must be in the coordinate system of the destination `PixMap` structure specified by the `dest` parameter.

m

A pointer to a `MatrixRecord` structure containing the transformation matrix for the destination video image. To determine the capabilities of a video digitizer component, you can call `VDGetDigitizerInfo` (page 39) in your application.

mask

A handle to a `MacRegion` structure that defines a mask. Applications can use masks to control clipping of the video into the destination rectangle. This mask region is defined in the destination coordinate space.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
All video digitizer components must support this function.

Special Considerations

Applications set the source digitizer rectangle by calling `VDSetDigitizerRect` (page 79).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDSetPlayThruGlobalRect

Establishes the destination settings for a video digitizer component that is to digitize into a global rectangle.

```
VideoDigitizerError VDSetPlayThruGlobalRect (
   VideoDigitizerComponent ci,
   GrafPtr theWindow,
   Rect *globalRect
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

theWindow

A pointer to the destination window.

globalRect

A pointer to a `Rect` structure that specifies the size and location of the video destination. This rectangle must be in the coordinate system of the destination window specified by the `theWindow` parameter.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
The application provides the desired settings as parameters to this function. Not all video digitizer components support global rectangles.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDSetPlayThruOnOff

Controls continuous digitization.

```
VideoDigitizerError VDSetPlayThruOnOff (
   VideoDigitizerComponent ci,
   short state
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

state

A short integer (see below) that specifies whether to use continuous digitization. When an application stops continuous digitization, the video digitizer component must restore its alpha channel, blending mask, or key color settings to graphics mode. See these constants:

`vdPlayThruOff`

`vdPlayThruOn`

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
When opened, video digitizer components are always set to off, so that no digitization is taking place. Your application can use this function to turn continuous digitization on and off.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetPLLFilterType

Specifies which phase locked loop (PLL) is to be active.

```
VideoDigitizerError VDSetPLLFilterType (
   VideoDigitizerComponent ci,
   short pllType
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

pllType

Indicates which PLL is to be active. Values are 0 for broadcast mode and 1 for videotape recorder mode.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetPreferredImageDimensions

Sets the preferred image dimensions for a video digitizer.

```
VideoDigitizerError VDSetPreferredImageDimensions (
   VideoDigitizerComponent ci,
   long width,
   long height
);
```

Parameters

ci

Specifies the video digitizer component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

width

The preferred image width.

height

The preferred image height.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetPreferredPacketSize

Sets the preferred packet size for video digitizing.

```
VideoDigitizerError VDSetPreferredPacketSize (
   VideoDigitizerComponent ci,
   long preferredPacketSizeInBytes
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

preferredPacketSizeInBytes

The preferred packet size in bytes.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
This function was added in QuickTime 2.5 to support videoconferencing applications.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDSetSaturation

Sets the saturation value.

```
VideoDigitizerError VDSetSaturation (
   VideoDigitizerComponent ci,
   unsigned short *saturation
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

saturation

A pointer to an integer that contains the new saturation value. The saturation value controls color intensity. For example, at high saturation levels, red appears to be red; at low saturation, red appears as pink. Valid saturation values range from 0 to 65,535, where 0 is the minimum saturation value and 65,535 specifies maximum saturation. The video digitizer component attempts to set the saturation value to the value specified by this parameter. The digitizer component returns the new value, so that the application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDSetSharpness

Sets the sharpness value.

```
VideoDigitizerError VDSetSharpness (
   VideoDigitizerComponent ci,
   unsigned short *sharpness
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

sharpness

A pointer to an integer that contains the new sharpness value. The sharpness value ranges from 0 to 65,535, where 0 represents no sharpness filtering and 65,535 represents full sharpness filtering. Higher values result in a visual impression of increased picture sharpness. The video digitizer component attempts to set the sharpness value to the value specified by this parameter. The digitizer component returns the new value, so that the application can avoid using unsupported values in future requests.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetTimeBase

Establishes the video digitizer's time coordinate system.

```
VideoDigitizerError VDSetTimeBase (
   VideoDigitizerComponent ci,
   TimeBase t
);
```

Parameters

ci

Identifies the application's connection to the video digitizer component. An application obtains this value from `OpenComponent` or `OpenDefaultComponent`.

t

A time base identifier. You can get this value from `NewTimeBase`.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetupBuffers

Defines output buffers for use with asynchronous grabs.

```
VideoDigitizerError VDSetupBuffers (
   VideoDigitizerComponent ci,
   VdigBufferRecListHandle bufferList
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

bufferList

A handle to a `VdigBufferRecList` structure. Video digitizer components extract information about the spatial characteristics of the video destinations from these buffers.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Discussion
If you are developing a video digitizer component, note that the `matrix` field in the buffer list structure contains a pointer to the `MatrixRecord` structure. It is your responsibility to copy that matrix structure.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDSetWhiteLevelValue

Sets the white level value.

```
VideoDigitizerError VDSetWhiteLevelValue (
   VideoDigitizerComponent ci,
   unsigned short *whiteLevel
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

whiteLevel

A pointer to an integer that contains the new white level value. White level values range from 0 to 65,535, where 0 represents the minimum white value and 65,535 represents the maximum white value.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDUseSafeBuffers

Instructs a video digitizer to use protected buffers.

```
VideoDigitizerError VDUseSafeBuffers (
   VideoDigitizerComponent ci,
   Boolean useSafeBuffers
);
```

Parameters

ci

Specifies the video digitizer component for this operation. Applications can obtain this reference from OpenComponent or OpenDefaultComponent.

useSafeBuffers

Pass TRUE to use protected buffers; pass FALSE otherwise.

Return Value
See Error Codes. Returns noErr if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

VDUseThisCLUT

Specifies the lookup table for color digitization.

```
VideoDigitizerError VDUseThisCLUT (
   VideoDigitizerComponent ci,
   CTabHandle colorTableHandle
);
```

Parameters

ci

The video digitizer component for the request. Applications obtain this reference from OpenComponent or OpenDefaultComponent.

colorTableHandle

A handle to a ColorTable structure. The video digitizer component uses the color table referred to by this parameter.

Return Value
See Error Codes. Returns noErr if there is no error.

Discussion
This feature is useful only for capturing 8-bit color video.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

Callbacks

Data Types

DigitizerInfo

Contains information about the capabilities and current status of a video digitizer component.

```
struct DigitizerInfo {
     short       vdigType;
     long        inputCapabilityFlags;
     long        outputCapabilityFlags;
     long        inputCurrentFlags;
     long        outputCurrentFlags;
     short       slot;
     GDHandle    gdh;
     GDHandle    maskgdh;
     short       minDestHeight;
     short       minDestWidth;
     short       maxDestHeight;
     short       maxDestWidth;
     short       blendLevels;
     long        reserved;
 };
```

Fields
`vdigType`
Discussion
Constant (see below) that specifies the type of video digitizer component. See these constants:

`vdTypeBasic`
`vdTypeAlpha`
`vdTypeMask`
`vdTypeKey`

`inputCapabilityFlags`

Discussion
Constant (see below) that specifies the capabilities of the video digitizer component with respect to the input video signal. See these constants:

- `digiInDoesNTSC`
- `digiInDoesPAL`
- `digiInDoesSECAM`
- `digiInDoesGenLock`
- `digiInDoesComposite`
- `digiInDoesComponent`
- `digiInVTR_Broadcast`
- `digiInDoesColor`
- `digiInDoesBW`

`outputCapabilityFlags`

Discussion
Constant (see below) that specifies the capabilities of the video digitizer component with respect to the output digitized video information. See these constants:

- `digiOutDoes1`
- `digiOutDoes2`
- `digiOutDoes4`
- `digiOutDoes8`
- `digiOutDoes16`
- `digiOutDoes32`
- `digiOutDoesDither`
- `digiOutDoesStretch`
- `digiOutDoesShrink`
- `digiOutDoesMask`
- `digiOutDoesDouble`
- `digiOutDoesQuad`
- `digiOutDoesQuarter`
- `digiOutDoesSixteenth`
- `digiOutDoesRotate`
- `digiOutDoesHorizFlip`
- `digiOutDoesVertFlip`
- `digiOutDoesSkew`
- `digiOutDoesBlend`
- `digiOutDoesWarp`
- `digiOutDoesHWPlayThru`
- `digiOutDoesILUT`
- `digiOutDoesKeyColor`
- `digiOutDoesAsyncGrabs`
- `digiOutDoesUnreadableScreenBits`
- `digiOutDoesCompress`
- `digiOutDoesCompressOnly`
- `digiOutDoesPlayThruDuringCompress`

`inputCurrentFlags`

Discussion
Specifies the current status of the video digitizer with respect to the input video signal. Video digitizer components report their current input status by returning a flags field that contains 1 bit for each of the applicable `inputCapabilityFlags` constants (see below), plus additional `inputCurrentFlags` constants (see below) as appropriate. The digitizer component sets these flags to reflect its current status. When reporting input status, for example, a video digitizer component sets the `digiInDoesGenLock` flag to 1 whenever the digitizer component is deriving its time signal from the input video. When reporting its input capabilities, the digitizer component sets this flag to 1 to indicate that it can derive its timing from the input video. See these constants:

`digiInSignalLock`

`outputCurrentFlags`

Discussion
Specifies the current status of the video digitizer with respect to the output video signal. Video digitizer components report their current output status by returning a flags field that contains 1 bit for each of the applicable `outputCapabilityFlags` constants (see below)

`slot`

Discussion
Identifies the slot that contains the video digitizer interface card.

`gdh`

Discussion
Contains a handle to the graphics device that defines the screen to which the digitized data is to be written. Set this field to `NIL` if your application is not constrained to a particular graphics device.

`maskgdh`

Discussion
Contains a handle to the graphics device that contains the `mask` plane. This field is used only by digitizers that clip by means of mask planes.

`minDestHeight`

Discussion
Indicates the smallest height value the digitizer component can accommodate in its destination.

`minDestWidth`

Discussion
Indicates the smallest width value the digitizer component can accommodate in its destination.

`maxDestHeight`

Discussion
Indicates the largest height value the digitizer component can accommodate in its destination.

`maxDestWidth`

Discussion
Indicates the largest width value the digitizer component can accommodate in its destination.

`blendLevels`

Discussion
Specifies the number of blend levels the video digitizer component supports.

`reserved`

Discussion
Reserved. Set this field to 0.

Discussion
Your application can retrieve information about the capabilities and current status of a video digitizer component. You call `VDGetDigitizerInfo` (page 39) to retrieve all this information from a video digitizer component. In response, the component formats a `DigitizerInfo` structure. The contents of this structure fully define the capabilities and current status of the video digitizer component.

Related Functions
`VDGetDigitizerInfo` (page 39)

Declared In
`QuickTimeComponents.h`

GrafPort

Defines a complete drawing environment for black-and-white graphics operations.

```
struct GrafPort {
     short         device;
     BitMap        portBits;
     Rect          portRect;
     RgnHandle     visRgn;
     RgnHandle     clipRgn;
     Pattern       bkPat;
     Pattern       fillPat;
     Point         pnLoc;
     Point         pnSize;
     short         pnMode;
     Pattern       pnPat;
     short         pnVis;
     short         txFont;
     StyleField    txFace;
     short         txMode;
     short         txSize;
     Fixed         spExtra;
     long          fgColor;
     long          bkColor;
     short         colrBit;
     short         patStretch;
     Handle        picSave;
     Handle        rgnSave;
     Handle        polySave;
     QDProcsPtr    grafProcs;
 };
```

Fields
`device`

Discussion
See `CGrafPort`.

`portBits`

Discussion
See `CGrafPort`. In a `GrafPort` structure, this field contains a complete 14-byte `BitMap` structure. In a `CGrafPort` structure, this field is partly replaced by the 4-byte `portPixMap` field, which contains a handle to a `PixMap` structure. In what would be the `rowBytes` field of the `BitMap` structure, a `CGrafPort` structure has a 2-byte `portVersion` field in which the two high bits are always set to 1. QuickTime uses these bits to distinguish `CGrafPort` records from `GrafPort` records, in which the two high bits of the `rowBytes` field are always 0. Following the `portBits` field in the `CGrafPort` structure are the `portVersion` and `grafVars` fields. The `grafVars` field contains a handle to a `GrafVars` structure; this handle is not included in the `GrafPort` structure. For information about the `GrafVars` structure, see *Inside Macintosh: Imaging With QuickDraw*.

`portRect`

Discussion
See `CGrafPort`.

`visRgn`

Discussion
See `CGrafPort`.

`clipRgn`

Discussion
See `CGrafPort`.

`bkPat`

Discussion
In a `GrafPort` structure, the `bkPat`, `pnPat`, and `fillPat` fields hold 8-byte bit patterns. In a `CGrafPort` structure, these fields are partly replaced by three 4-byte handles to pixel patterns. The resulting 12 bytes of additional space in the `CGrafPort` structure are taken up by the `rgbFgColor` and `rgbBkColor` fields, which contain 6-byte `RGBColor` structures specifying the optimal foreground and background colors for the color graphics port. Note that the closest matching available colors, which QuickTime actually uses to render the foreground and background, are stored in the `fgColor` and `bkColor` fields of the `CGrafPort` structure.

`fillPat`

Discussion
See the `bkPat` field (above).

`pnLoc`

Discussion
See `CGrafPort`.

`pnSize`

Discussion
See `CGrafPort`.

`pnMode`

Discussion
See `CGrafPort`.

`pnPat`

Discussion
See the `bkPat` field (above).

`pnVis`

Discussion
See `CGrafPort`.

`txFont`

Discussion
See `CGrafPort`.

`txFace`

Discussion
The character `style` of the text, with values from the set defined by the `Style` type, which includes such styles as bold, italic, and shaded. You can apply stylistic variations either alone or in combination. This field is initially set to plain text.

`txMode`

Discussion
See `CGrafPort`.

`txSize`

Discussion
See `CGrafPort`.

`spExtra`

Discussion
See `CGrafPort`.

`fgColor`

Discussion
See `CGrafPort`.

`bkColor`

Discussion
See `CGrafPort`.

`colrBit`

Discussion
See `CGrafPort`.

`patStretch`

Discussion
See `CGrafPort`.

`picSave`

Discussion
See `CGrafPort`.

`rgnSave`

Discussion
See `CGrafPort`.

`polySave`

Discussion
See `CGrafPort`.

`grafProcs`

Discussion
See `CGrafPort`. In a `GrafPort` structure, you can supply this field with a pointer to a `QDProcs` structure; in a `CGrafPort` structure, you provide this field with a pointer to a `CQDProcs` structure.

Discussion
See `CGrafPort`.

Version Notes
The `GrafPort` structure has been largely superseded by the `CGrafPort` structure, which defines a full color environment. The two structures are the same size; QuickTime distinguishes between them by examining the `portBits` field.

Declared In
`QuickTimeComponents.h`

GrafPtr

Represents a type used by the Video Components API.

```
typedef GrafPort * GrafPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickdrawTypes.h`

QTVideoOutputComponent

Represents a type used by the Video Components API.

```
typedef ComponentInstance QTVideoOutputComponent;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

RectPtr

Represents a type used by the Video Components API.

```
typedef Rect * RectPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`IOMacOSTypes.h`

VDCompressionListHandle

Represents a type used by the Video Components API.

```
typedef VDCompressionListPtr * VDCompressionListHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDCompressionListPtr

Represents a type used by the Video Components API.

```
typedef VDCompressionList * VDCompressionListPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VDGammaRecord

Holds a gamma table.

```
struct VDGammaRecord {
     Ptr    csGTable;
 };
```

Fields
`csGTable`

Discussion
A pointer to a gamma table.

Declared In
`QuickTimeComponents.h`

VDGamRecPtr

Represents a type used by the Video Components API.

```
typedef VDGammaRecord * VDGamRecPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`IOMacOSVideo.h`

VdigBufferRecListHandle

Represents a type used by the Video Components API.

```
typedef VdigBufferRecListPtr * VdigBufferRecListHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VdigBufferRecListPtr

Represents a type used by the Video Components API.

```
typedef VdigBufferRecList * VdigBufferRecListPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VideoDigitizerComponent

Represents a type used by the Video Components API.

```
typedef ComponentInstance VideoDigitizerComponent;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

VideoDigitizerError

Represents a type used by the Video Components API.

```
typedef ComponentResult VideoDigitizerError;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

Constants

compositeIn

Constants grouped with compositeIn.

```
enum {
  compositeIn                   = 0,    /* input is composite format */
  sVideoIn                      = 1,    /* input is sVideo format */
  rgbComponentIn                = 2,    /* input is rgb component format */
  rgbComponentSyncIn            = 3,    /* input is rgb component format (sync on
 green?)*/
  yuvComponentIn                = 4,    /* input is yuv component format */
  yuvComponentSyncIn            = 5,    /* input is yuv component format (sync on
 green?) */
  tvTunerIn                     = 6,
  sdiIn                         = 7
};
```

Declared In
`QuickTimeComponents.h`

Video Digitizer Capabilities

Flags that indicate the input and output capabilities of a video digitizer.

```
enum {
  digiInDoesNTSC                = 1L << 0, /* digitizer supports NTSC input format
 */
  digiInDoesPAL                 = 1L << 1, /* digitizer supports PAL input format
 */
  digiInDoesSECAM               = 1L << 2, /* digitizer supports SECAM input format
 */
  digiInDoesGenLock             = 1L << 7, /* digitizer does genlock */
  digiInDoesComposite           = 1L << 8, /* digitizer supports composite input
type */
  digiInDoesSVideo              = 1L << 9, /* digitizer supports S-Video input type
 */
  digiInDoesComponent           = 1L << 10, /* digitizer supports component = rgb,
 input type */
  digiInVTR_Broadcast           = 1L << 11, /* digitizer can differentiate between
 the two */
  digiInDoesColor               = 1L << 12, /* digitizer supports color */
  digiInDoesBW                  = 1L << 13, /* digitizer supports black & white */
                                    /* Digitizer Input Current Flags = these
are valid only during active operating conditions,   */
  digiInSignalLock              = 1L << 31 /* digitizer detects input signal is
locked, this bit = horiz lock || vertical lock */
};
enum {
  digiOutDoes1                  = 1L << 0, /* digitizer supports 1 bit pixels */
  digiOutDoes2                  = 1L << 1, /* digitizer supports 2 bit pixels */
  digiOutDoes4                  = 1L << 2, /* digitizer supports 4 bit pixels */
  digiOutDoes8                  = 1L << 3, /* digitizer supports 8 bit pixels */
  digiOutDoes16                 = 1L << 4, /* digitizer supports 16 bit pixels */
  digiOutDoes32                 = 1L << 5, /* digitizer supports 32 bit pixels */
  digiOutDoesDither             = 1L << 6, /* digitizer dithers in indexed modes
*/
  digiOutDoesStretch            = 1L << 7, /* digitizer can arbitrarily stretch */
  digiOutDoesShrink             = 1L << 8, /* digitizer can arbitrarily shrink */
  digiOutDoesMask               = 1L << 9, /* digitizer can mask to clipping regions
 */
  digiOutDoesDouble             = 1L << 11, /* digitizer can stretch to exactly
double size */
  digiOutDoesQuad               = 1L << 12, /* digitizer can stretch exactly
quadruple size */
  digiOutDoesQuarter            = 1L << 13, /* digitizer can shrink to exactly
quarter size */
  digiOutDoesSixteenth          = 1L << 14, /* digitizer can shrink to exactly
sixteenth size */
  digiOutDoesRotate             = 1L << 15, /* digitizer supports rotate
transformations */
  digiOutDoesHorizFlip          = 1L << 16, /* digitizer supports horizontal flips
 Sx < 0 */
  digiOutDoesVertFlip           = 1L << 17, /* digitizer supports vertical flips
Sy < 0 */
  digiOutDoesSkew               = 1L << 18, /* digitizer supports skew = shear,twist,
 */
  digiOutDoesBlend              = 1L << 19,
  digiOutDoesWarp               = 1L << 20,
  digiOutDoesHW_DMA             = 1L << 21, /* digitizer not constrained to local
 device */
  digiOutDoesHWPlayThru         = 1L << 22, /* digitizer doesn't need time to play
 thru */
```

```
  digiOutDoesILUT              = 1L << 23, /* digitizer does inverse LUT for index
 modes */
  digiOutDoesKeyColor           = 1L << 24, /* digitizer does key color functions
 too */
  digiOutDoesAsyncGrabs         = 1L << 25, /* digitizer supports async grabs */
  digiOutDoesUnreadableScreenBits = 1L << 26, /* playthru doesn't generate readable
 bits on screen*/
  digiOutDoesCompress          = 1L << 27, /* supports alternate output data types
 */
  digiOutDoesCompressOnly       = 1L << 28, /* can't provide raw frames anywhere
*/
  digiOutDoesPlayThruDuringCompress = 1L << 29, /* digi can do playthru while
providing compressed data */
  digiOutDoesCompressPartiallyVisible = 1L << 30, /* digi doesn't need all bits
visible on screen to do hardware compress */
  digiOutDoesNotNeedCopyOfCompressData = 1L << 31 /* digi doesn't need any
bufferization when providing compressed data */
};
```

Constants

`digiInDoesNTSC`

The video digitizer supports National Television System Committee (NTSC) format input video signals. This flag is set to 1 if the digitizer component supports NTSC video.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesPAL`

The video digitizer component supports Phase Alternation Line (PAL) format input video signals. This flag is set to 1 if the digitizer component supports PAL video.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesSECAM`

The video digitizer component supports Systeme Electronique Couleur avec Memoire (SECAM) format input video signals. This flag is set to 1 if the digitizer component supports SECAM video.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesGenLock`

The video digitizer component supports genlock; that is, the digitizer can derive its timing from an external time base. This flag is set to 1 if the digitizer component supports genlock.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesComposite`

The video digitizer component supports composite input video. This flag is set to 1 if the digitizer component supports composite input.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesComponent`

The video digitizer component supports RGB input video. This flag is set to 1 if the digitizer component supports RGB input.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInVTR_Broadcast`

The video digitizer component can distinguish between an input signal that emanates from a videotape player and a broadcast signal. This flag is set to 1 if the digitizer component can differentiate between the two different signal types.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesColor`

The video digitizer component supports color input. This flag is set to 1 if the digitizer component can accept color input.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInDoesBW`

The video digitizer component supports grayscale input. This flag is set to 1 if the digitizer component can accept grayscale input.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiInSignalLock`

The video digitizer component is locked onto the input signal. If this flag is set to 1, the digitizer component detects either vertical or horizontal signal lock.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoes1`

The video digitizer component can work with pixel maps that contain 1-bit pixels. If this flag is set to 1, then the digitizer component can write images that contain 1-bit pixels. If this flag is set to 0, then the digitizer component cannot handle such images.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoes2`

The video digitizer component can work with pixel maps that contain 2-bit pixels. If this flag is set to 1, then the digitizer component can write images that contain 2-bit pixels. If this flag is set to 0, then the digitizer component cannot handle such images.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoes4`

The video digitizer component can work with pixel maps that contain 4-bit pixels. If this flag is set to 1, then the digitizer component can write images that contain 4-bit pixels. If this flag is set to 0, then the digitizer component cannot handle such images.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoes8`

The video digitizer component can work with pixel maps that contain 8-bit pixels. If this flag is set to 1, then the digitizer component can write images that contain 8-bit pixels. If this flag is set to 0, then the digitizer component cannot handle such images.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoes16`

The video digitizer component can work with pixel maps that contain 16-bit pixels. If this flag is set to 1, then the digitizer component can write images that contain 16-bit pixels. If this flag is set to 0, then the digitizer component cannot handle such images.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoes32`

The video digitizer component can work with pixel maps that contain 32-bit pixels. If this flag is set to 1, then the digitizer component can write images that contain 32-bit pixels. If this flag is set to 0, then the digitizer component cannot handle such images.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesDither`

The video digitizer component supports dithering. If this flag is set to 1, the component supports dithering of colors. If this flag is set to 0, the digitizer component does not support dithering.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesStretch`

The video digitizer component can stretch images to arbitrary sizes. If this flag is set to 1, the digitizer component can stretch images. If this flag is set to 0, the digitizer component does not support stretching.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesShrink`

The video digitizer component can shrink images to arbitrary sizes. If this flag is set to 1, the digitizer component can shrink images. If this flag is set to 0, the digitizer component does not support shrinking.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesMask`

The video digitizer component can handle clipping regions. If this flag is set to 1, the digitizer component can mask to an arbitrary clipping region. If this flag is set to 0, the digitizer component does not support clipping regions.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesDouble`

The video digitizer component supports stretching to quadruple size when displaying the output video. The parameters for the stretch operation are specified in the matrix structure for the request; the component modifies the scaling attributes of the matrix (see the chapter "Movie Toolbox" in Inside Macintosh: QuickTime for information about transformation matrices). If this flag is set to 1, the digitizer component can stretch an image to exactly four times its original size, up to the maximum size specified by the `maxDestHeight` and `maxDestWidth` fields in the digitizer information structure. If this flag is set to 0, the digitizer component does not support stretching to quadruple size.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesQuad`

The video digitizer component supports stretching an image to 16 times its original size when displaying the output video. The parameters for the stretch operation are specified in the matrix structure for the request; the component modifies the scaling attributes of the matrix (see the chapter "Movie Toolbox" in Inside Macintosh: QuickTime for information about transformation matrices). If this flag is set to 1, the digitizer component can stretch an image to exactly 16 times its original size, up to the maximum size specified by the `maxDestHeight` and `maxDestWidth` fields in the digitizer information structure. If this flag is set to 0, the digitizer component does not support this capability.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesQuarter`

The video digitizer component can shrink an image to one-quarter of its original size when displaying the output video. The parameters for the shrink operation are specified in the matrix structure for the request; the component modifies the scaling attributes of the matrix (see the chapter "Movie Toolbox" in Inside Macintosh: QuickTime for information about transformation matrices). If this flag is set to 1, the digitizer component can shrink an image to exactly one-quarter of its original size, down to the minimum size specified by the `minDestHeight` and `minDestWidth` fields in the digitizer information structure. If this flag is set to 0, the digitizer component does not support this capability.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesSixteenth`

The video digitizer component can shrink an image to 1/16 of its original size when displaying the output video. The parameters for the shrink operation are specified in the matrix structure for the request; the digitizer component modifies the scaling attributes of the matrix (see the chapter "Movie Toolbox" in Inside Macintosh: QuickTime for information about transformation matrices). If this flag is set to 1, the digitizer component can shrink an image to exactly 1/16 of its original size, down to the minimum size specified by the `minDestHeight` and `minDestWidth` fields in the digitizer information structure. If this flag is set to 0, the digitizer component does not support this capability.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesRotate`

The video digitizer component can rotate an image when displaying the output video. The parameters for the rotation are specified in the matrix structure for an operation. If this flag is set to 1, the digitizer component can rotate the image. If this flag is set to 0, the digitizer component cannot rotate the resulting image.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesHorizFlip`

The video digitizer component can flip an image horizontally when displaying the output video. The parameters for the horizontal flip are specified in the matrix structure for an operation. If this flag is set to 1, the digitizer component can flip the image. If this flag is set to 0, the digitizer component cannot flip the resulting image.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesVertFlip`

The video digitizer component can flip an image vertically when displaying the output video. The parameters for the vertical flip are specified in the matrix structure for an operation. If this flag is set to 1, the digitizer component can flip the image. If this flag is set to 0, the digitizer component cannot flip the resulting image.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesSkew`

The video digitizer component can skew an image when displaying the output video. Skewing an image distorts it linearly along only a single axis; for example, drawing a rectangular image into a parallelogram-shaped region. The parameters for the skew operation are specified in the matrix structure for the request. If this flag is set to 1, the digitizer component can skew an image. If this flag is set to 0, the digitizer component does not support this capability.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesBlend`

The video digitizer component can blend the resulting image with a matte when displaying the output video. The matte is provided by the application by defining either an alpha channel or a mask plane. If this flag is set to 1, the digitizer component can blend. If this flag is set to 0, the digitizer component does not support this capability.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesWarp`

The video digitizer component can warp an image when displaying the output video. Warping an image distorts it along one or more axes, perhaps nonlinearly, in effect "bending" the result region. The parameters for the warp operation are specified in the matrix structure for the request. If this flag is set to 1, the digitizer component can warp an image. If this flag is set to 0, the digitizer component does not support this capability.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesHWPlayThru`

The video digitizer component does not need idle time in order to display its video. If this flag is set to 1, your application does not need to grant processor time to the digitizer component at normal display speeds.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`digiOutDoesILUT`
: The video digitizer component supports inverse lookup tables for indexed color modes. If this flag is set to 1, the digitizer component uses inverse lookup tables when appropriate.

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

`digiOutDoesKeyColor`
: The video digitizer component supports clipping by means of key colors. If this flag is set to 1, the digitizer component can clip to a region defined by a key color.

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

`digiOutDoesAsyncGrabs`
: The video digitizer component can operate asynchronously. If this flag is set to 1, your application can use the `VDSetupBuffers` and `VDGrabOneFrameAsync` functions (described on page 0-669 and page 0-671, respectively).

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

`digiOutDoesUnreadableScreenBits`
: The video digitizer may place pixels on the screen that cannot be used when compressing images.

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

`digiOutDoesCompress`
: The video digitizer component supports compressed source devices. These devices provide compressed data directly, without having to use the Image Compression Manager. See "Controlling Compressed Source Devices" beginning on page 0-657 for more information about the functions that applications can use to work with compressed source devices.

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

`digiOutDoesCompressOnly`
: The video digitizer component only provides compressed image data; the component cannot provide displayable data. This flag only applies to digitizers that support compressed source devices.

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

`digiOutDoesPlayThruDuringCompress`
: The video digitizer component can draw images on the screen at the same time that it is delivering compressed image data. This flag only applies to digitizers that support compressed source devices.

  Available in Mac OS X v10.0 and later.

  Declared in `QuickTimeComponents.h`.

Declared In
`QuickTimeComponents.h`

currentIn

Constants grouped with currentIn.

```
enum {
  ntscIn                        = 0,    /* current input format */
  currentIn                     = 0,    /* ntsc input format */
  palIn                         = 1,    /* pal input format */
  secamIn                       = 2,    /* secam input format */
  ntscReallyIn                  = 3     /* ntsc input format */
};
```

Declared In
QuickTimeComponents.h

VDGetDeviceNameAndFlags Values

Constants passed to VDGetDeviceNameAndFlags.

```
enum {
  vdDeviceFlagShowInputsAsDevices = (1 << 0), /* Tell the Panel to promote Inputs
 to Devices*/
  vdDeviceFlagHideDevice        = (1 << 1) /* Omit this Device entirely from the
list*/
};
```

Declared In
QuickTimeComponents.h

vdFlagCaptureAlwaysUseTimeBase

Constants grouped with vdFlagCaptureAlwaysUseTimeBase.

```
enum {
  vdFlagCaptureStarting         = (1 << 0), /* Capture is about to start; allocate
 bandwidth */
  vdFlagCaptureStopping         = (1 << 1), /* Capture is about to stop; stop
queuing frames*/
  vdFlagCaptureIsForPreview     = (1 << 2), /* Capture is just to screen for preview
 purposes*/
  vdFlagCaptureIsForRecord      = (1 << 3), /* Capture is going to be recorded*/
  vdFlagCaptureLowLatency       = (1 << 4), /* Fresh frames are more important than
 delivering every frame - don't queue too much*/
  vdFlagCaptureAlwaysUseTimeBase = (1 << 5), /* Use the timebase for every frame;
 don't worry about making durations uniform*/
  vdFlagCaptureSetSettingsBegin = (1 << 6), /* A series of calls are about to be
made to restore settings.*/
  vdFlagCaptureSetSettingsEnd   = (1 << 7) /* Finished restoring settings; any set
 calls after this are from the app or UI*/
};
```

Declared In
QuickTimeComponents.h

VDSetPlayThruOnOff Values

Constants passed to VDSetPlayThruOnOff.

```
enum {
  vdPlayThruOff                 = 0,
  vdPlayThruOn                  = 1
};
```

Declared In
`QuickTimeComponents.h`

VdigType Values

Constants passed to VdigType.

```
enum {
  vdTypeBasic                   = 0,    /* basic, no clipping */
  vdTypeAlpha                   = 1,    /* supports clipping with alpha channel */
  vdTypeMask                    = 2,    /* supports clipping with mask plane */
  vdTypeKey                     = 3     /* supports clipping with key color(s) */
};
```

Constants

`vdTypeBasic`
Basic video digitizer; does not support any clipping.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`vdTypeAlpha`
Supports clipping by means of an alpha channel.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`vdTypeMask`
Supports clipping by means of a mask plane.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

Declared In
`QuickTimeComponents.h`

VDSetFieldPreference Values

Constants passed to VDSetFieldPreference.

```
enum {
  vdUseAnyField                 = 0,    /* Digitizers choice on field use */
  vdUseOddField                 = 1,    /* Use odd field for half size vert and
smaller */
  vdUseEvenField                = 2     /* Use even field for half size vert and
smaller */
};
```

Declared In
`QuickTimeComponents.h`

Document Revision History

This table describes the changes to *Video Components Reference for QuickTime*.

Date	Notes
2006-05-23	New document, based on previously published material, that describes the API for QuickTime video components.

Document Revision History

Index

C

D

G

Q

R

V