

## **RGBEasy SDK Programmer's Guide**

V3.1.4

## Document Revision History

1.0.0	06/06/08	First Release.
1.1.0	13/02/09	Added downscaling and output size functions.
1.1.4	23/02/10	Updated for VisionSD4+1 and VisionSD8
1.4.1	03/03/10	Added RGBGetCroppingOverScan
1.5.0	09/05/10	Added On Screen Display functions. Changed userData passed through callback functions to be of type ULONG_PTR to enable future 64bit support.
1.8.0	19/05/11	Added support for 64bit RGBEASY applications.
1.9.0	08/02/12	Added: Additional On Screen Display functions, Equalisation functions, and SDI and Dual Link DVI supported functions.
2.0.0	08/11/13	Added colour domain and audio functions
2.0.1	24/11/13	Added Chassis structure for GetInputInfo
2.0.2	25/11/14	Added FrameCaptureEx and MediaSample functions.
2.1.0	03/12/14	Added DirectGPU functions
3.0.0	28/04/17	Added or updated the following functions: RGBLoadEx, RGBInputIsHDMISupported, RGBInputIsDisplayPortSupported, RGBInputIsMediaSampleTypeSupported, RGBGetColourDomainDetect, RGBGetPixelGamut, RGBSetPixelGamut, RGBGetPixelRange, RGBSetPixelRange, RGBInputIsGangingSupported, RGBInputIsGangingTypeSupported, RGBInputGetGangingType, RGBInputSetGangingType, RGBIsHDCPSupported, RGBInputIsHDCPSupported, RGBInputIsHDCPProtected, RGBInputGetHDCPEEnabled, RGBInputSetHDCPEEnabled, RGBEnableInputSnapshot, RGBIsInputSnapshotEnabled and RGBInputGetStatus.
3.1.0	03/07/17	Updated the SDK Contents and SDK samples section.  Added Document Revision History.  Added or updated the following functions: RGBInputGetDeviceType, RGBGetLinkRate, RGBSetLinkRate, RGBInputIsLinkRateSupported,

		<p>RGBGetCaptureRate, RGBSetCaptureRate and RGBInputGetMaximumPixelClock.</p> <p>Added or updated the following structures and enumerations: RGBMODECHANGEDINFO, CAPTURECARD, RGBDEVICETYPE, CAPTURESTATE, PIXELGAMUT, PIXELRANGE, PIXELFORMAT, DEINTERLACE, SIGNALTYPE, BUFFERTYPE, RGB_FRAME_FLAGS, ANALOG_INPUT_TYPE, DIGITAL_INPUT_TYPE, VGA_INPUT_FLAGS, SDI_INPUT_FLAGS, RGBOSD_TYPE, ROTATIONANGLE, NVIDIAOP, SIGNALDETECT, COLOURDOMAINDETECT, LIVESTREAM, LIVESTREAM_SUPPORT, LINKRATE, RGBGANG_TYPE, DGCMEIASAMPLETYPE, DGCMEIASAMPLESUBTYPE and RGBMODEINFO.</p>
3.1.2	15/11/17	Added or updated the following functions: RGBInputGetModelInfo.
3.1.3	18/05/18	Added or updated the following functions and enumerations: RGBMODECHANGEDINFO, RGMODEINFO and DISPLAYPORT_INPUT_FLAGS.
3.1.4	16/04/19	Added RGBInputGetBitmapSnapshot and RGBINPUTBITMAPSNAPSHOTFN.

## Table of Contents

Document Revision History.....	2
Table of Contents .....	4
Overview.....	16
SDK Contents .....	17
RGBEasy SDK Sample applications .....	19
TimeStamps .....	21
Callbacks.....	22
User supplied buffers.....	23
RGBEasy Rendering.....	24
RGBEasy Function Reference .....	25
RGBLoad.....	26
RGBLoadEx.....	27
RGBFree.....	28
RGBGetCaptureCard .....	29
RGBGetNumberOfInputs.....	30
RGBIsDirectDMASupported .....	31
RGBIsDeinterlaceSupported .....	32
RGBIsYUVSupported .....	33
RGBInputIsVGASupported.....	34
RGBInputIsDVISupported .....	35
RGBInputIsComponentSupported.....	36
RGBInputIsCompositeSupported .....	37
RGBInputIsSVideoSupported .....	38
RGBGetInputSignalType.....	39
RGBInputGetDeviceType.....	40
RGBOpenInput .....	41
RGBCloseInput .....	42
RGBCloseInputs.....	43
RGBDetectInput .....	44

RGBSetInput.....	45
RGBGetInput .....	46
RGBSetWindow.....	47
RGBGetWindow .....	48
RGBResetCapture.....	49
RGBGetHorPositionMinimum .....	50
RGBGetHorPositionMaximum.....	51
RGBGetHorPositionDefault .....	52
RGBGetHorPosition .....	53
RGBSetHorPosition.....	54
RGBGetHorScaleMinimum .....	55
RGBGetHorScaleMaximum .....	56
RGBGetHorScaleDefault.....	57
RGBGetHorScale .....	58
RGBSetHorScale.....	59
RGBGetCaptureWidthMinimum.....	60
RGBGetCaptureWidthMaximum.....	61
RGBGetCaptureWidthDefault .....	62
RGBGetCaptureWidth.....	63
RGBTestCaptureWidth.....	64
RGBSetCaptureWidth .....	65
RGBGetVerPositionMinimum.....	66
RGBGetVerPositionMaximum .....	67
RGBGetVerPositionDefault .....	68
RGBGetVerPosition.....	69
RGBSetVerPosition .....	70
RGBGetCaptureHeightMinimum.....	71
RGBGetCaptureHeightMaximum.....	72
RGBGetCaptureHeightDefault .....	73
RGBGetCaptureHeight.....	74

RGBSetCaptureHeight .....	75
RGBGetBrightnessMinimum.....	76
RGBGetBrightnessMaximum.....	77
RGBGetBrightnessDefault .....	78
RGBGetBrightness .....	79
RGBSetBrightness.....	80
RGBGetContrastMinimum.....	81
RGBGetContrastMaximum .....	82
RGBGetContrastDefault .....	83
RGBGetContrast.....	84
RGBSetContrast .....	85
RGBGetColourBalanceMinimum .....	86
RGBGetColourBalanceMaximum .....	87
RGBGetColourBalanceDefault.....	88
RGBGetColourBalance .....	89
RGBSetColourBalance .....	90
RGBGetBlackLevelMinimum .....	91
RGBGetBlackLevelMaximum.....	92
RGBGetBlackLevelDefault.....	93
RGBGetBlackLevel .....	94
RGBSetBlackLevel.....	95
RGBGetPhaseMinimum .....	96
RGBGetPhaseMaximum .....	97
RGBGetPhaseDefault.....	98
RGBGetPhase .....	99
RGBSetPhase.....	100
RGBGetSaturationMinimum.....	101
RGBGetSaturationMaximum.....	102
RGBGetSaturationDefault .....	103
RGBGetSaturation.....	104

RGBSetSaturation .....	105
RGBGetHueMinimum .....	106
RGBGetHueMaximum.....	107
RGBGetHueDefault .....	108
RGBGetHue .....	109
RGBSetHue.....	110
RGBGetEqualisationMinimum.....	111
RGBGetEqualisationMaximum .....	112
RGBGetEqualisationDefault .....	113
RGBGetEqualisation.....	114
RGBSetEqualisation .....	115
RGBInputIsEqualisationSupported .....	116
RGBGetLinkRate .....	117
RGBSetLinkRate.....	118
RGBInputIsLinkRateSupported.....	119
RGBInputIsSDISupported .....	120
RGBInputIsDualLinkDVISupported.....	121
RGBInputIsHDMISupported.....	122
RGBInputIsDisplayPortSupported .....	123
RGBGetVideoStandard .....	124
RGBTestVideoStandard .....	125
RGBSetVideoStandard.....	126
RGBGetFrameDroppingMinimum .....	127
RGBGetFrameDroppingMaximum .....	128
RGBGetFrameDroppingDefault.....	129
RGBGetFrameDropping.....	130
RGBSetFrameDropping .....	131
RGBGetFrameRate.....	132
RGBGetCaptureRate .....	133
RGBSetCaptureRate.....	134

RGBGetCroppingMinimum .....	135
RGBGetCroppingMaximum .....	136
RGBGetCroppingDefault.....	137
RGBGetCroppingOverscan .....	138
RGBGetCropping.....	139
RGBTestCropping.....	140
RGBSetCropping .....	141
RGBIsCroppingEnabled.....	142
RGBEnableCropping.....	143
RGBGetDeinterlace .....	144
RGBSetDeinterlace .....	145
RGBPauseCapture .....	146
RGBResumeCapture.....	147
RGBGetCaptureState.....	148
RGBGetMessageDelay .....	149
RGBSetMessageDelay .....	150
RGBGetPixelFormat.....	151
RGBSetPixelFormat .....	152
RGBSaveCurrentFrame .....	153
RGBGetDirectDMA .....	154
RGBSetDirectDMA.....	155
RGBSetFrameCapturedFn .....	156
RGBSetFrameCapturedFnEx .....	157
RGBSetMediaSampleCapturedFn .....	158
RGBDrawFrame.....	159
RGBSaveBitmap.....	160
RGBSetModeChangedFn.....	161
RGBSetNoSignalFn .....	162
RGBNoSignal.....	163
RGBSetDrawNoSignalFn .....	164

RGBSetInvalidSignalFn .....	165
RGBInvalidSignal .....	166
RGBSetDrawInvalidSignalFn .....	167
RGBSetValueChangedFn .....	168
RGBsetErrorFn .....	169
RGBStartCapture.....	170
RGBStopCapture .....	171
RGBInputIsLiveStreamSupported .....	172
RGBGetLiveStream .....	173
RGBSetLiveStream .....	174
RGBInputIsMediaSampleTypeSupported .....	175
RGBGetModelInfo .....	176
RGBInputGetModelInfo .....	177
RGBSetDownScaling.....	178
RGBGetDownScaling .....	179
RGBSetOutputSize.....	180
RGBGetOutputSize .....	181
RGBUseOutputBuffers.....	182
RGBChainOutputBuffers.....	183
RGBChainOutputBufferEx .....	184
RGBChainMediaSample.....	185
RGBDirectGPUInit .....	186
RGBDirectGPUclose .....	187
RGBDirectGPUNVIDIAOp .....	188
RGBCreateOSD .....	189
RGBDeleteOSD .....	190
RGBAttachOSD .....	191
RGBDetachOSD .....	192
RGBGetFirstOSD .....	193
RGBGetNextOSD .....	194

RGBSetOSDType .....	195
RGBGetOSDType.....	196
RGBSetOSDScaling.....	197
RGBGetOSDScaling .....	198
RGBSetOSDBackground.....	199
RGBGetOSDBackground .....	200
RGBSetOSDText .....	201
RGBGetOSDTextLength.....	202
RGBGetOSDText.....	203
RGBSetOSDWrapping .....	204
RGBSetOSDBitmapFilename .....	205
RGBGetOSDWrapping.....	206
RGBSetOSDFont .....	207
RGBGetOSDFont.....	208
RGBSetOSDMargins .....	209
RGBGetOSDMargins.....	210
RGBSetOSDAlignment .....	211
RGBGetOSDAlignment.....	212
RGBIsOSDAccelerated.....	213
RGBSetOSDOwnerDrawnFn .....	214
RGBSetOSDArea.....	215
RGBGetOSDArea .....	216
RGBSetKeyColour .....	217
RGBGetKeyColour.....	218
RGBGetInputInfo.....	219
RGBSetRotation .....	220
RGBGetRotation.....	221
RGBIsRotationSupported.....	222
RGBSetNoSignalText.....	223
RGBSetInvalidSignalText.....	224

RGBSetNoSignalBackground .....	225
RGBSetInvalidSignalBackground .....	226
RGBAddModeToModeStore.....	227
RGBDeleteModeFromModeStore .....	228
RGBIsEDIDSupported .....	229
RGBGetEDID.....	230
RGBSetEDID .....	231
RGBIsEDIDEnabled .....	232
RGBResetEDID .....	233
RGBInputGetMaximumPixelClock.....	234
RGBSetSignalDetectionMethod .....	235
RGBGetSignalDetectionMethod .....	236
RGBIsSignalDetectionMethodSupported .....	237
RGBSetColourDomain .....	238
RGBGetColourDomain .....	239
RGBGetColourDomainDefault.....	240
RGBGetColourDomainDetect.....	241
RGBGetPixelGamut.....	242
RGBSetPixelGamut .....	243
RGBGetPixelRange.....	244
RGBSetPixelRange .....	245
RGBInputIsGangingSupported .....	246
RGBInputIsGangingTypeSupported .....	247
RGBInputGetGangingType .....	248
RGBInputSetGangingType.....	249
RGBIsHDCPSupported .....	250
RGBInputIsHDCPSupported .....	251
RGBInputIsHDCPProtected .....	252
RGBInputGetHDCPEEnabled.....	253
RGBInputSetHDCPEEnabled .....	254

RGBEnableInputSnapshot .....	255
RGBIsInputSnapshotEnabled.....	256
RGBInputGetStatus.....	257
RGBInputGetBitmapSnapshot.....	258
RGBAudioIsAudioSupported .....	259
RGBAudioIsHDMISupported.....	260
RGBAudioIsSDISupported .....	261
RGBAudioIsAnalogueSupported .....	262
RGBAudioGetCapabilities .....	263
RGBAudioGetCapabilitiesCount .....	264
RGBAudioSetCapabilities .....	265
RGBAudioOpenInput .....	266
RGBAudioCloseInput.....	267
RGBAudioChainOutputBuffer.....	268
RGBAudioLoadOutputBuffer.....	269
RGBAudioReleaseOutputBuffers.....	270
RGBAudioSetState.....	271
RGBAudioGetState .....	272
RGBAudioSetDigitalChannelPair.....	273
RGBAudioGetDigitalChannelPair .....	274
RGBAudioSetDigitalMute .....	275
RGBAudioGetDigitalMute.....	276
RGBAudioSetAnalogueBalancedGain .....	277
RGBAudioGetAnalogueBalancedGain.....	278
RGBAudioGetAnalogueBalancedGainMinimum.....	279
RGBAudioGetAnalogueBalancedGainMaximum.....	280
RGBAudioGetAnalogueBalancedGainDefault .....	281
RGBAudioGetAnalogueBalancedGainScale.....	282
RGBAudioSetAnalogueBalancedGainBoost .....	283
RGBAudioGetAnalogueBalancedGainBoost.....	284

RGBAudioSetAnalogueBalancedMute .....	285
RGBAudioGetAnalogueBalancedMute .....	286
RGBAudioSetAnalogueUnbalancedGain .....	287
RGBAudioGetAnalogueUnbalancedGain .....	288
RGBAudioGetAnalogueUnbalancedGainMinimum .....	289
RGBAudioGetAnalogueUnbalancedGainMaximum .....	290
RGBAudioGetAnalogueUnbalancedGainDefault.....	291
RGBAudioGetAnalogueUnbalancedGainScale .....	292
RGBAudioSetAnalogueUnbalancedMute .....	293
RGBAudioGetAnalogueUnbalancedMute.....	294
RGBAudioSetADCGain .....	295
RGBAudioGetADCGain .....	296
RGBAudioGetADCGainMinimum .....	297
RGBAudioGetADCGainMaximum .....	298
RGBAudioGetADCGainDefault.....	299
RGBAudioGetADCGainScale .....	300
RGBAudioSetADCMute .....	301
RGBAudioGetADCMute.....	302
RGBAudioIsLineOutSupported.....	303
RGBAudioSetLineOutSource.....	304
RGBAudioGetLineOutSource .....	305
RGBAudioSetLineOutGain.....	306
RGBAudioGetLineOutGain .....	307
RGBAudioGetLineOutGainMinimum .....	308
RGBAudioGetLineOutGainMaximum .....	309
RGBAudioGetLineOutGainDefault.....	310
RGBAudioGetLineOutGainScale .....	311
RGBAudioSetLineOutMute .....	312
RGBAudioGetLineOutMute.....	313
RGBEasy Callback Functions.....	314

RGBFRAMECAPTUREDFN.....	315
RGBFRAMECAPTUREDFNEX .....	316
RGBMEDIASAMPLECAPTUREDFN.....	317
RGBMODECHANGEDFN .....	318
RGBNOSIGNALFN .....	319
RGBDRAWNOSIGNALFN .....	320
RGBINVALIDSIGNALFN.....	321
RGBDRAWINVALIDSIGNALFN .....	322
RGBERRORFN .....	323
RGBVALUECHANGEDFN .....	324
RGBOSDDRAWFN .....	325
RGBINPUTBITMAPSNAPSHOTFN.....	326
RGBEasy Enumerations .....	327
CAPTURECARD .....	328
RGBDEVICETYPE .....	329
CAPTURESTATE .....	330
PIXELGAMUT .....	331
PIXELRANGE .....	332
PIXELFORMAT .....	333
DEINTERLACE.....	335
SIGNALTYPE .....	336
BUFFERTYPE.....	337
RGB_FRAME_FLAGS .....	338
ANALOG_INPUT_TYPE.....	339
DIGITAL_INPUT_TYPE .....	340
VGA_INPUT_FLAGS .....	341
SDI_INPUT_FLAGS .....	342
DISPLAYPORT_INPUT_FLAGS .....	343
RGBOSD_TYPE .....	344
ROTATIONANGLE .....	345

RGBOSD flags.....	346
GRAPHICSHARDWARE.....	347
NVIDIAOP .....	348
SIGNALDETECT .....	349
COLOURDOMAINDETECT .....	350
LIVESTREAM .....	352
LIVESTREAM_SUPPORT .....	353
LINKRATE.....	354
RGBGANG_TYPE .....	355
DGCMEDIASAMPLETYPE .....	356
DGCMEDIASAMPLESUBTYPE .....	357
RGBEasy Structures .....	359
RGBFRAMEDATA.....	360
RGBMODECHANGEDINFO .....	361
RGBMODEINFO .....	363
RGBVALUECHANGEDINFO.....	365
SIGNEDVALUE .....	367
UNSIGNEDVALUE.....	368
RGBDRIVERVER .....	369
RGBLOCATION .....	370
RGBCHASSIS .....	371
RGBINPUTINFO .....	372
DGCMEDIASAMPLE.....	374
DGCVIDEOHEADER.....	375
DGCMEMORYBUFFERHEADER .....	376
DGCMEMORYBUFFER.....	377
GPUTRANSFERDESCRIPTOR.....	378

## Overview

RGBEasy is an application interface for the capture and display of video and audio data from the Vision family capture cards. RGBEasy is implemented within the rgbeasy.dll, a dynamic link library that is part of the Vision capture card driver.

## SDK Contents

Support for the RGEasy is included within the drivers for a Datapath capture cards.

The RGEasy SDK is implemented in C and has the following structure.

### COMMON\API.C

This file implements functions to help with dynamically loading RGEasy and obtaining pointers to functions.

### COMMON\AUDIOCAP.C

This file implements audio functions that can be used to play captured audio through the Windows WASAPI interface.

### COMMON\DGCMEDIAHELPER.C

This file implements helper functions for use with DGCMEDIASAMPLESUBTYPE media samples.

### INCLUDE\API.H

This file defines functions to help with dynamically loading RGEasy and obtaining pointers to the functions.

### INCLUDE\AUDIO.H

This file defines the audio specific structures and constants used within RGEasy. Errors specific to RGEasy are defined within this file.

### INCLUDE\AUDIOAPI.H

This file defines all the audio functions available within RGEasy.

### INCLUDE\AUDIOCAP.H

This file defines audio functions that can be used to play captured audio through the Windows WASAPI interface.

### INCLUDE\DGCMEDIA.H

This file defines the DGCMEDIASAMPLESUBTYPE media sample structure that adds support for planar NV12 and YV12.

### INCLUDE\DGCMEDIAHELPER.H

This file defines helper functions for use with DGCMEDIASAMPLESUBTYPE media samples.

### INCLUDE\DGCTYPES.H

This file contains definitions of types used by DGCEDIASAMPLESUBTYPE media samples for compilers that do not include *<stdint.h>*.

INCLUDE\RGB.H

This file defines the video specific structures and constants used within RGEasy. Errors specific to RGEasy are defined within this file.

INCLUDE\RGBAPI.H

This file defines all the video functions available within RGEasy.

INCLUDE\RGBERROR.H

This file defines error codes that can be returned by the capture card device driver.

INCLUDE\VIDSTD.H

This file defines the video standards that may be supported by an input.

LIB\WIN32\RELEASE\RGEASY.LIB

LIB\X64\RELEASE\RGEASY.LIB

The RGEasy import library, client applications that wish to automatically load the RGEasy on start-up should link to this import library.

DOCS\RGEASY SDK Programmer.pdf

This document.

SAMPLE Directories

These directories contain sample Visual Studio applications demonstrating how to use RGEasy, see above for further details.

## RGBEasy SDK Sample applications

### Sample1

Demonstrates how to display a RGB window using static linking. The sample also demonstrates how to enable Input Ganging on supported capture cards. The application is statically linked to RGBEasy.

### Sample1A

Demonstrates how to display a RGB window using dynamic linking.

### Sample1B

Demonstrates how to display a RGB window containing on screen display messages.

### Sample1C

Demonstrates how to display a RGB window containing owner drawn OSD messages on suitable graphics devices. Only graphics devices supporting the accelerated OSD interface are supported.

### Sample2

Demonstrates how to display video within a RGB window and play audio through the Windows WASAPI interface. The application provides dialogs to configure both the video and audio settings.

### Sample3

Demonstrates how to capture RGB data using a RGBFRAMECAPTUREDFN frame captured callback function and a single SDK supplied buffer.

### Sample3A

Demonstrates how to capture RGB data using a RGBFRAMECAPTUREDFN frame captured callback and multiple application provided buffers.

### Sample3B

Demonstrates how to capture RGB data using a RGBFRAMECAPTUREDFNEX frame captured callback and multiple application provided buffers. Demonstrates the use of a RGBMODECHANGEDFN mode changed call back function to maintain a 1:1 buffer size with respect to the input signal resolution.

### Sample3C

Demonstrates how to capture RGB data using a RGBFRAMECAPTUREDFN frame captured callback and multiple application provided DirectX buffers. Demonstrates the use of on screen display with DirectX buffers and the use of LiveStream to reduce latency.

### Sample3D

Demonstrates how to capture RGB data using a RGBMEDIASAMPLECAPTUREDFN media sample captured callback and multiple application provided NV12 planar buffers

#### Sample4

Demonstrates how to display captured RGB data within a control on a dialog.

#### Sample5

Demonstrates how to display a RGB window using AMD DirectGMA and NVIDIA GPUDirect technologies via the RGBEasy DirectGPU functions.

## TimeStamps

The Vision capture card time stamp corresponds to the time at which the capture of a field or frame completes within the Vision hardware, based upon a local hardware clock that is synchronised to the system's High Precision Performance Counter (QueryPerformanceCounter) every 5 seconds. The units of the timestamp are in 100ns 'ticks', but the values are presented with a resolution of 100 microseconds masking the five-second synchronization. For example, multiple gen locked captures have the same time stamp for two independent buffers.

Sample3B found within the RGBEasy SDK details a frame delivery call back function, FrameCapturedFnEx, see Sample3B and rgb.h within the SDK for further information.

## Callbacks

The following user supplied callbacks are available for the RGBEasy SDK user:

- RGBSetFrameCapturedFn
- RGBSetFrameCapturedFnEx
- RGBSetMediaSampleCapturedFn
- RGBSetModeChangedFn
- RGBSetNoSignalFn
- RGBSetDrawNoSignalFn
- RGBSetInvalidSignalFn
- RGBSetDrawInvalidSignalFn
- RGBSetErrorFn
- RGBSetValueChangedFn
- RGBSetOSDOwnerDrawnFn

## User supplied buffers

A call to RGBStopCapture will block until the last frame of data submitted to the Vision driver has completed. Therefore, it is possible to receive a frame capture callback in a separate thread whilst this call is blocked. However after RGBStopCapture returns no further callbacks will occur.

User defined buffers will be held by the RGBEasy subsystem until a call to RGBUseOutputBuffers with bEnableBuffers set to FALSE.

We recommend the following sequence:

- RGBUseOutputBuffers(FALSE)
- RGBStopCapture
- RGBCloseInput

No threads are ever terminated. The above function calls are blocked until any current DMA is completed.

If users require control of the capture card data buffer, for example performing graphics commands on the buffer, we advise the use of the RGBEasy SDK. Sample3C of the Vision RGBEasy SDK allows the user to allocate a buffer on a third party graphics device using Direct3D. The Direct3D buffer can then be passed to the RGBEasy subsystem and filled with capture data (in this case DMA'd directly to graphics device physical memory) a call back function then notifies the application on completion. OpenGL technologies, Sample5 can easily be used in place of Direct3D using RGBEasy.

## RGBEasy Rendering

The Vision RGBEasy application employs the Microsoft Direct3D interface for rendering capture data to third party graphic devices (not including Datapath graphics devices). By default, captured frames are DMA'd via system memory prior to any third party graphics device physical memory unless the user has supplied their own rendering technique. This default functionality is preferred for stability across a selection of independent graphics devices. Configuration via the registry is possible to DMA directly to a graphics devices video memory or back buffer. Systems that display Vision capture data on a Datapath display card do not require DirectX as an intermediate technology. A private interface is used to render Datapath captured data to a Datapath display device. For a complete Datapath capture and display system the collaboration of Microsoft and the third party graphics vendor to transfer the data is no longer required. The private Datapath transfer interface allows for both minimum input/output latency and increased PCIe throughput by up scaling on the display card.

## RGBEasy Function Reference

This section documents the functions available within RGBEasy

## RGBLoad

```
unsigned long  
RGBLoad (  
    PHRGBDLL phRGBDLL )
```

This function loads the RGBEasy interface. It returns a handle to the RGBEasy interface that must be used when closing the interface. RGBLoad must be the first RGBEasy function called by an application. A return value of RGBERROR\_NO\_ERROR indicates that one or more compatible capture cards have been found, configured and are ready to be used. This function will fail if there are no Datapath RGB capture cards installed or if the driver is not correctly installed.

### *Parameters*

phRGBDLL  
A pointer to the HRGBDLL variable that receives the handle to the RGBEasy interface.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBLoadEx

```
unsigned long  
RGBLoadEx (  
    PHRGBDLL    phRGBDLL,  
    CAPTURECARD captureCard )
```

This function loads the RGEasy interface for a particular capture card type. It returns a handle to the RGEasy interface that must be used when closing the interface. RGBLoad must be the first RGEasy function called by an application. A return value of RGBERROR\_NO\_ERROR indicates that one or more compatible capture cards have been found, configured and are ready to be used. This function will fail if there are no Datapath RGB capture cards installed or if the driver is not correctly installed.

### *Parameters*

phRGBDLL  
A pointer to the HRGBDLL variable that receives the handle to the RGEasy interface.

captureCard  
The type of capture card to load.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBFree

```
unsigned long  
RGBFree (  
    HRGBDLL hRGBDLL )
```

This function closes the RGBEasy interface and releases all resources claimed by the interface. RGBFree must be the last RGBEasy function called by an application. RGBFree must not be called if RGBLoad failed.

### *Parameters*

hRGBDLL  
The handle returned by RGBLoad.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureCard

```
unsigned long  
RGBGetCaptureCard (  
    PCAPTURECARD pCaptureCard )
```

This function returns the type of capture card currently under control of the RGBEasy interface.

### *Parameters*

pCaptureCard

A pointer to a variable that receives the type of capture card detected.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetNumberOfInputs

```
unsigned long  
RGBGetNumberOfInputs (  
    unsigned long *pNumberOfInputs )
```

This function returns the number of available inputs.

### *Parameters*

`pNumberOfInputs`

A pointer to a variable that receives the number of available inputs.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBIsDirectDMASupported

```
unsigned long  
RGBIsDirectDMASupported (  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the captured data can transfer directly from the capture card to the display device using DMA.

### *Parameters*

`pBIsSupported`  
Pointer to a variable that receives a value that indicates whether direct DMA is supported. A value of zero indicates that direct DMA is not supported. A value of one indicates that direct DMA is supported.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBIsDeinterlaceSupported

```
unsigned long  
RGBIsDeinterlaceSupported (  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the deinterlacing of interlaced video is supported.

### *Parameters*

pBIsSupported

Pointer to a variable that receives a value that indicates whether deinterlacing is supported. A value of zero indicates that deinterlacing is not supported. A value of one indicates deinterlacing is supported.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsYUVSupported

```
unsigned long  
RGBIsYUVSupported (  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the drawing of YUV data is supported by the display card.

### *Parameters*

pBIsSupported

Pointer to a variable that receives a value that indicates whether the drawing of YUV data is supported. A value of zero indicates that YUV drawing is not supported. A value of one indicates YUV drawing is supported.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsVGASupported

```
unsigned long  
RGBIsVGASupported (  
    unsigned long input,  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the input is capable of capturing VGA sources.

### *Parameters*

**Input**  
Specifies the input to query.

**pBIsSupported**  
Pointer to a variable that receives a value that indicates whether the input is capable of capturing VGA sources. A value of zero indicates that VGA capture is not supported. A value of one indicates VGA capture is supported.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsDVISupported

```
unsigned long  
RGBIsDVISupported (  
    unsigned long input,  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the input is capable of capturing DVI sources.

### *Parameters*

**Input**  
Specifies the input to query.

**pBIsSupported**  
Pointer to a variable that receives a value that indicates whether the input is capable of capturing DVI sources. A value of zero indicates that DVI capture is not supported. A value of one indicates DVI capture is supported.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBInputIsComponentSupported

```
unsigned long  
RGBIsComponentSupported (  
    unsigned long input,  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the input is capable of capturing component sources.

### *Parameters*

**Input**  
Specifies the input to query.

**pBIsSupported**  
Pointer to a variable that receives a value that indicates whether the input is capable of capturing component sources. A value of zero indicates that component capture is not supported. A value of one indicates component capture is supported.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBInputIsCompositeSupported

```
unsigned long  
RGBIsCompositeSupported (  
    unsigned long input,  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the input is capable of capturing composite sources.

### *Parameters*

**Input**  
Specifies the input to query.

**pBIsSupported**  
Pointer to a variable that receives a value that indicates whether the input is capable of capturing composite sources. A value of zero indicates that composite capture is not supported. A value of one indicates composite capture is supported.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBInputIsSVideoSupported

```
unsigned long  
RGBIsCompositeSupported (  
    unsigned long input,  
    signed long *pBIsSupported )
```

Returns a value that indicates whether the input is capable of capturing S-Video sources.

### *Parameters*

**Input**  
Specifies the input to query.

**pBIsSupported**  
Pointer to a variable that receives a value that indicates whether the input is capable of capturing S-Video sources. A value of zero indicates that S-Video capture is not supported. A value of one indicates S-Video capture is supported.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetInputSignalType

```
unsigned long  
RGBGetInputSignalType (  
    unsigned long input,  
    PSIGNALTYPE  pSignalType,  
    unsigned long *pCaptureWidth,  
    unsigned long *pCaptureHeight,  
    unsigned long *pRefreshRate )
```

Returns information about the type of source connected an input.

### Parameters

input

A pointer to a variable that receives a value that indicates the type of source connected.

pSignalType

Pointer to a variable that receives the parameter describing the SIGNALTYPE of the source connected.

pCaptureWidth

Pointer to a variable that receives the capture width of the source connected.

pCaptureHeight

Pointer to a variable that receives the capture height of the source connected.

pRefreshRate

Pointer to a variable that receives the vertical refresh rate of the source connected.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetDeviceType

```
unsigned long  
RGBInputGetDeviceType (  
    unsigned long input,  
    RGBDEVICETYPE *pDeviceType )
```

Returns information about the type of capture device the input belongs to.

### *Parameters*

input

A pointer to a variable that receives a value that indicates the type of source connected.

pDeviceType

Pointer to a variable that receives the RGBDEVICETYPE parameter.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBOpenInput

```
unsigned long  
RGBOpenInput (  
    unsigned long uInput,  
    PHRGB pHRGB )
```

Opens a capture on the specified input.

### *Parameters*

uInput

The input to open. The input must be a value in the range 0 to numberOfInputs - 1. The number of inputs can be obtained by calling RGBGetNumberOfInputs.

pHRGB

A pointer to the variable that receives the handle that identifies the capture.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBCloseInput

```
unsigned long  
RGBCloseInput (  
    HRGB hRGB )
```

Closes a capture and frees the resources allocated to it.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBCloseInputs

```
unsigned long  
RGBCloseInputs (  
    PHRGB phRGBArray,  
    unsigned long uInputs )
```

Closes an array of captures and frees the resources allocated to them. If no error is returned all the RGB captures will have finished by the time this function returns.

### *Parameters*

phRGBArray

An array of handles returned by RGBOpenInput. Captures that are successfully closed are set to NULL in the returning array. If an error is returned there may be captures left open.

uInput

The number of capture handles to be closed.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDetectInput

```
unsigned long  
RGBSetInput (  
    HRGB hRGB )
```

Detects the video mode of the specified RGB capture and, if set, runs one of the user defined callback functions.

### *Parameters*

hRGB  
The RGB capture handle.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetInput

```
unsigned long  
RGBSetInput (  
    HRGB hRGB,  
    unsigned long uInput )
```

Sets the input number of the specified RGB capture.

### *Parameters*

hRGB

The RGB capture handle.

uInput

Specifies the new input number to open. The input must be a value in the range 0 to numberOfInputs - 1. The number of inputs can be obtained by calling RGBGetNumberOfInputs.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetInput

```
unsigned long  
RGBGetInput (  
    HRGB hRGB,  
    unsigned long *puInput )
```

Gets the current input number of the specified RGB capture.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

puInput  
A pointer to the variable that receives the current input number.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetWindow

```
unsigned long  
RGBSetWindow (  
    HRGB hRGB,  
    HWND hWnd )
```

Sets the window in which the specified RGB capture is to be displayed.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

hWnd  
The window handle.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetWindow

```
unsigned long  
RGBSetWindow (  
    HRGB hRGB,  
    HWND hWnd )
```

Gets the window handle of the window in which the specified RGB capture is being displayed.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

hWnd

A pointer to the variable that receives the window handle.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBResetCapture

```
unsigned long  
RGBResetCapture (  
    HRGB hRGB )
```

Reset the capture parameters to their initially detected values.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHorPositionMinimum

```
unsigned long  
RGBGetHorPositionMinimum (  
    HRGB hRGB,  
    signed long *pHorPosition )
```

Returns the minimum value that can be set for the number of pixels between the horizontal sync and the beginning of capture.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pHorPosition`  
A pointer to the variable that receives the minimum horizontal position.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetHorPositionMaximum

```
unsigned long  
RGBGetHorPositionMaximum (  
    HRGB hRGB,  
    signed long *pHorPosition )
```

Returns the maximum value that can be set for the number of pixels between the horizontal sync and the beginning of capture.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pHorPosition`  
A pointer to the variable that receives the maximum horizontal position.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetHorPositionDefault

```
unsigned long  
RGBGetHorPositionDefault (  
    HRGB hRGB,  
    signed long *pHorPosition )
```

Returns the default value for the number of pixels between the horizontal sync and the beginning of capture.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pHorPosition`  
A pointer to the variable that receives the default horizontal position.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetHorPosition

```
unsigned long  
RGBGetHorPosition (  
    HRGB hRGB,  
    signed long *pHorPosition )
```

Returns the currently set value that for the number of pixels between the horizontal sync and the beginning of capture.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pHorPosition`  
A pointer to the variable that receives the current horizontal position.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBSetHorPosition

```
unsigned long  
RGBSetHorPosition (  
    HRGB hRGB,  
    signed long horPosition )
```

Sets the number of pixels between the horizontal sync and the beginning of capture.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

horPosition  
The horizontal position to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHorScaleMinimum

```
unsigned long  
RGBGetHorScaleMinimum (  
    HRGB hRGB,  
    unsigned long *pHorScale )
```

Returns the minimum value that can be set for the total number of pixels on a line.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pHorScale**  
A pointer to the variable that receives the minimum horizontal scale.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHorScaleMaximum

```
unsigned long  
RGBGetHorScaleMaximum (  
    HRGB hRGB,  
    unsigned long *pHorScale )
```

Returns the maximum value that that can be set for the total number of pixels on a line.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pHorScale**  
A pointer to the variable that receives the maximum horizontal scale.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHorScaleDefault

```
unsigned long  
RGBGetHorScaleDefault (  
    HRGB hRGB,  
    unsigned long *pHorScale )
```

Returns the default value for the total number of pixels on a line.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pHorScale**  
A pointer to the variable that receives the default horizontal scale.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHorScale

```
unsigned long  
RGBGetHorScale (  
    HRGB hRGB,  
    unsigned long *pHorScale )
```

Returns the currently set value that for the total number of pixels on a line.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pHorScale**  
A pointer to the variable that receives the current horizontal scale.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetHorScale

```
unsigned long  
RGBSetHorScale (  
    HRGB hRGB,  
    unsigned long horScale )
```

Sets the total number of pixels on a line.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

horScale  
The horizontal scale to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureWidthMinimum

```
unsigned long  
RGBGetCaptureWidthMinimum (  
    HRGB hRGB,  
    unsigned long *pCaptureWidth )
```

Returns the minimum value that can be set for the number of pixels on each line that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureWidth**  
A pointer to the variable that receives the minimum capture width.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureWidthMaximum

```
unsigned long  
RGBGetCaptureWidthMaximum (  
    HRGB hRGB,  
    unsigned long *pCaptureWidth )
```

Returns the maximum value that that can be set for the number of pixels on each line that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureWidth**  
A pointer to the variable that receives the maximum capture width.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureWidthDefault

```
unsigned long  
RGBGetCaptureWidthDefault (  
    HRGB hRGB,  
    unsigned long *pCaptureWidth )
```

Returns the default value for the number of pixels on each line that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureWidth**  
A pointer to the variable that receives the default capture width.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureWidth

```
unsigned long  
RGBGetCaptureWidth (  
    HRGB hRGB,  
    unsigned long *pCaptureWidth )
```

Returns the currently set value that for the number of pixels on each line that are to be captured.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pCaptureWidth  
A pointer to the variable that receives the current capture width.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBTestCaptureWidth

```
unsigned long  
RGBTestCaptureWidth (  
    HRGB hRGB,  
    unsigned long captureWidth )
```

The RGB capture cards require the number of pixels on each line to be aligned to certain natural boundaries. This alignment value may not be the same for the different RGB capture cards. This function tests whether a value is correct for the RGB capture card in use.

### Parameters

hRGB

The handle returned by RGBOpenInput.

captureWidth

The capture width to test.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR.

If the captureWidth value is not aligned correctly then one of the following errors will be returned:

- RGBERROR\_HORADDRTIME\_NOT\_WORD\_DIVISIBLE
- RGBERROR\_HORADDRTIME\_NOT\_DWORD\_DIVISIBLE
- RGBERROR\_HORADDRTIME\_NOT\_QWORD\_DIVISIBLE

If the function fails for any other reason an appropriate RGB error defined in RGBERROR.H or a standard windows error code will be returned.

## RGBSetCaptureWidth

```
unsigned long  
RGBSetCaptureWidth (  
    HRGB hRGB,  
    unsigned long captureWidth )
```

Sets the number of pixels on each line that are to be captured. Use `RGBTestCaptureWidth` to verify that the capture width value is aligned appropriately for the RGB capture card in use.

### Parameters

`hRGB`  
The handle returned by `RGBOpenInput`.

`captureWidth`  
The capture width to set.

### Return Values

If the function succeeds, the return value is `RGBERROR_NO_ERROR`.

If the `captureWidth` value is not aligned correctly then one of the following errors will be returned:

- `RGBERROR_HORADDRTIME_NOT_WORD_DIVISIBLE`
- `RGBERROR_HORADDRTIME_NOT_DWORD_DIVISIBLE`
- `RGBERROR_HORADDRTIME_NOT_QWORD_DIVISIBLE`

If the function fails for any other reason an appropriate RGB error defined in `RGBERROR.H` or a standard windows error code will be returned.

## RGBGetVerPositionMinimum

```
unsigned long  
RGBGetVerPositionMinimum (  
    HRGB hRGB,  
    signed long *pVerPosition )
```

Returns the minimum value that can be set for the number of lines between the vertical sync and the beginning of capture.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pVerPosition`  
A pointer to the variable that receives the minimum vertical position.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetVerPositionMaximum

```
unsigned long  
RGBGetVerPositionMaximum (  
    HRGB hRGB,  
    signed long *pVerPosition )
```

Returns the maximum value that can be set for the number of lines between the vertical sync and the beginning of capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pVerPosition**  
A pointer to the variable that receives the maximum vertical position.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetVerPositionDefault

```
unsigned long  
RGBGetVerPositionDefault (  
    HRGB hRGB,  
    signed long *pVerPosition )
```

Returns the default value for the number of lines between the vertical sync and the beginning of capture.

### *Parameters*

*hRGB*  
The handle returned by RGBOpenInput.

*pVerPosition*  
A pointer to the variable that receives the default vertical position.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetVerPosition

```
unsigned long  
RGBGetVerPosition (  
    HRGB hRGB,  
    signed long *pVerPosition )
```

Returns the currently set value that for the number of lines between the vertical sync and the beginning of capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pVerPosition**  
A pointer to the variable that receives the current vertical position.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetVerPosition

```
unsigned long  
RGBSetVerPosition (  
    HRGB hRGB,  
    signed long verPosition )
```

Sets the number of lines between the vertical sync and the beginning of capture.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

verPosition  
The vertical position to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureHeightMinimum

```
unsigned long  
RGBGetCaptureHeightMinimum (  
    HRGB hRGB,  
    unsigned long *pCaptureHeight )
```

Returns the minimum value that can be set for the number of lines that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureHeight**  
A pointer to the variable that receives the minimum capture height.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureHeightMaximum

```
unsigned long  
RGBGetCaptureHeightMaximum (  
    HRGB hRGB,  
    unsigned long *pCaptureHeight )
```

Returns the maximum value that that can be set for the number of lines that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureHeight**  
A pointer to the variable that receives the maximum capture height.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureHeightDefault

```
unsigned long  
RGBGetCaptureHeightDefault (  
    HRGB hRGB,  
    unsigned long *pCaptureHeight )
```

Returns the default value for the number of lines that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureHeight**  
A pointer to the variable that receives the default capture height.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureHeight

```
unsigned long  
RGBGetCaptureHeight (  
    HRGB hRGB,  
    unsigned long *pCaptureHeight )
```

Returns the currently set value that for the number of lines that are to be captured.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pCaptureHeight**  
A pointer to the variable that receives the current capture height.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetCaptureHeight

```
unsigned long  
RGBSetCaptureHeight (  
    HRGB hRGB,  
    unsigned long captureHeight )
```

Sets the number of lines that are to be captured.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`captureHeight`  
The capture height to set.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetBrightnessMinimum

```
unsigned long  
RGBGetBrightnessMinimum (  
    HRGB hRGB,  
    signed long *pBrightness )
```

Returns the minimum value that can be set for the brightness.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBrightness**  
A pointer to the variable that receives the minimum brightness.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBrightnessMaximum

```
unsigned long  
RGBGetBrightnessMaximum (  
    HRGB hRGB,  
    signed long *pBrightness )
```

Returns the maximum value that can be set for the brightness.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBrightness**  
A pointer to the variable that receives the maximum brightness.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBrightnessDefault

```
unsigned long  
RGBGetBrightnessDefault (  
    HRGB hRGB,  
    signed long *pBrightness )
```

Returns the default value for the brightness.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBrightness**  
A pointer to the variable that receives the default brightness.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBrightness

```
unsigned long  
RGBGetBrightness (  
    HRGB hRGB,  
    signed long *pBrightness )
```

Returns the currently set value that for the brightness.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBrightness**  
A pointer to the variable that receives the current brightness.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetBrightness

```
unsigned long  
RGBSetBrightness (  
    HRGB hRGB,  
    signed long brightness )
```

Sets the brightness.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

brightness  
The brightness to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetContrastMinimum

```
unsigned long  
RGBGetContrastMinimum (  
    HRGB hRGB,  
    signed long *pContrast )
```

Returns the minimum value that can be set for the Contrast.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pContrast  
A pointer to the variable that receives the minimum contrast.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetContrastMaximum

```
unsigned long  
RGBGetContrastMaximum (  
    HRGB hRGB,  
    signed long *pContrast )
```

Returns the maximum value that can be set for the Contrast.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pContrast  
A pointer to the variable that receives the maximum contrast.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetContrastDefault

```
unsigned long  
RGBGetContrastDefault (  
    HRGB hRGB,  
    signed long *pContrast )
```

Returns the default value for the Contrast.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pContrast  
A pointer to the variable that receives the default contrast.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetContrast

```
unsigned long  
RGBGetContrast (  
    HRGB hRGB,  
    signed long *pContrast )
```

Returns the currently set value that for the Contrast.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pContrast  
A pointer to the variable that receives the current contrast.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetContrast

```
unsigned long  
RGBSetContrast (  
    HRGB hRGB,  
    signed long contrast )
```

Sets the Contrast.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

contrast  
The contrast to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourBalanceMinimum

```
unsigned long  
RGBGetColourBalanceMinimum (  
    HRGB          hRGB,  
    signed long *pBrightnessRed,  
    signed long *pBrightnessGreen,  
    signed long *pBrightnessBlue,  
    signed long *pContrastRed,  
    signed long *pContrastGreen,  
    signed long *pContrastBlue )
```

Returns the minimum values that can be set for the colour balance.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pBrightnessRed

A pointer to the variable that receives the minimum red brightness.

pBrightnessGreen

A pointer to the variable that receives the minimum green brightness.

pBrightnessBlue

A pointer to the variable that receives the minimum blue brightness.

pContrastRed

A pointer to the variable that receives the minimum red contrast.

pContrastGreen

A pointer to the variable that receives the minimum green contrast.

pContrastBlue

A pointer to the variable that receives the minimum blue contrast.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourBalanceMaximum

```
unsigned long  
RGBGetColourBalanceMaximum (  
    HRGB          hRGB,  
    signed long *pBrightnessRed,  
    signed long *pBrightnessGreen,  
    signed long *pBrightnessBlue,  
    signed long *pContrastRed,  
    signed long *pContrastGreen,  
    signed long *pContrastBlue )
```

Returns the maximum values that can be set for the colour balance.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pBrightnessRed

A pointer to the variable that receives the maximum red brightness.

pBrightnessGreen

A pointer to the variable that receives the maximum green brightness.

pBrightnessBlue

A pointer to the variable that receives the maximum blue brightness.

pContrastRed

A pointer to the variable that receives the maximum red contrast.

pContrastGreen

A pointer to the variable that receives the maximum green contrast.

pContrastBlue

A pointer to the variable that receives the maximum blue contrast.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourBalanceDefault

```
unsigned long  
RGBGetColourBalanceDefault (  
    HRGB          hRGB,  
    signed long *pBrightnessRed,  
    signed long *pBrightnessGreen,  
    signed long *pBrightnessBlue,  
    signed long *pContrastRed,  
    signed long *pContrastGreen,  
    signed long *pContrastBlue )
```

Returns the default values that can be set for the colour balance.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pBrightnessRed

A pointer to the variable that receives the default red brightness.

pBrightnessGreen

A pointer to the variable that receives the default green brightness.

pBrightnessBlue

A pointer to the variable that receives the default blue brightness.

pContrastRed

A pointer to the variable that receives the default red contrast.

pContrastGreen

A pointer to the variable that receives the default green contrast.

pContrastBlue

A pointer to the variable that receives the default blue contrast.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourBalance

```
unsigned long  
RGBGetColourBalance (  
    HRGB          hRGB,  
    signed long *pBrightnessRed,  
    signed long *pBrightnessGreen,  
    signed long *pBrightnessBlue,  
    signed long *pContrastRed,  
    signed long *pContrastGreen,  
    signed long *pContrastBlue )
```

Returns the current values for the colour balance.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pBrightnessRed

A pointer to the variable that receives the red brightness.

pBrightnessGreen

A pointer to the variable that receives the green brightness.

pBrightnessBlue

A pointer to the variable that receives the blue brightness.

pContrastRed

A pointer to the variable that receives the red contrast.

pContrastGreen

A pointer to the variable that receives the green contrast.

pContrastBlue

A pointer to the variable that receives the blue contrast.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetColourBalance

```
unsigned long  
RGBSetColourBalance (  
    HRGB          hRGB,  
    signed long brightnessRed,  
    signed long brightnessGreen,  
    signed long brightnessBlue,  
    signed long contrastRed,  
    signed long contrastGreen,  
    signed long contrastBlue )
```

Sets the colour balance.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pBrightnessRed

The red brightness to set.

pBrightnessGreen

The green brightness to set.

pBrightnessBlue

The blue brightness to set.

pContrastRed

The red contrast to set.

pContrastGreen

The green contrast to set.

pContrastBlue

The blue contrast to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBlackLevelMinimum

```
unsigned long  
RGBGetBlackLevelMinimum (  
    HRGB hRGB,  
    signed long *pBlackLevel )
```

Returns the minimum value that can be set for the black level sample position.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBlackLevel**  
A pointer to the variable that receives the minimum black level.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBlackLevelMaximum

```
unsigned long  
RGBGetBlackLevelMaximum (  
    HRGB hRGB,  
    signed long *pBlackLevel )
```

Returns the maximum value that can be set for the black level sample position.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBlackLevel**  
A pointer to the variable that receives the maximum black level.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBlackLevelDefault

```
unsigned long  
RGBGetBlackLevelDefault (  
    HRGB hRGB,  
    signed long *pBlackLevel )
```

Returns the default value for the black level sample position.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBlackLevel**  
A pointer to the variable that receives the default black level.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetBlackLevel

```
unsigned long  
RGBGetBlackLevel (  
    HRGB hRGB,  
    signed long *pBlackLevel )
```

Returns the currently set value that for the black level sample position.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pBlackLevel**  
A pointer to the variable that receives the current black level.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetBlackLevel

```
unsigned long  
RGBSetBlackLevel (  
    HRGB hRGB,  
    signed long blackLevel )
```

Sets the black level.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

blackLevel  
The black level to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetPhaseMinimum

```
unsigned long  
RGBGetPhaseMinimum (  
    HRGB hRGB,  
    signed long *pPhase )
```

Returns the minimum value that can be set for the phase of the sampling clock in relation to the pixel clock.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pPhase  
A pointer to the variable that receives the minimum phase.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetPhaseMaximum

```
unsigned long  
RGBGetPhaseMaximum (  
    HRGB hRGB,  
    signed long *pPhase )
```

Returns the maximum value that can be set for the phase of the sampling clock in relation to the pixel clock.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pPhase`  
A pointer to the variable that receives the maximum phase.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetPhaseDefault

```
unsigned long  
RGBGetPhaseDefault (  
    HRGB hRGB,  
    signed long *pPhase )
```

Returns the default value for the phase of the sampling clock in relation to the pixel clock.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pPhase  
A pointer to the variable that receives the default phase.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetPhase

```
unsigned long  
RGBGetPhase (  
    HRGB hRGB,  
    signed long *pPhase )
```

Returns the currently set value that for the phase of the sampling clock in relation to the pixel clock.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pPhase  
A pointer to the variable that receives the current phase.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetPhase

```
unsigned long  
RGBSetPhase (  
    HRGB hRGB,  
    signed long phase )
```

Sets the phase of the sampling clock in relation to the pixel clock.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

phase  
The phase to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetSaturationMinimum

```
unsigned long  
RGBGetSaturationMinimum (  
    HRGB hRGB,  
    signed long *pSaturation )
```

Returns the minimum value that can be set for the saturation.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pSaturation  
A pointer to the variable that receives the minimum saturation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetSaturationMaximum

```
unsigned long  
RGBGetSaturationMaximum (  
    HRGB hRGB,  
    signed long *pSaturation )
```

Returns the maximum value that can be set for the saturation.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pSaturation  
A pointer to the variable that receives the maximum saturation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetSaturationDefault

```
unsigned long  
RGBGetSaturationDefault (  
    HRGB hRGB,  
    signed long *pSaturation )
```

Returns the default value for the saturation.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pSaturation  
A pointer to the variable that receives the default saturation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetSaturation

```
unsigned long  
RGBGetSaturation (  
    HRGB hRGB,  
    signed long *pSaturation )
```

Returns the currently set value that for the saturation.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pSaturation  
A pointer to the variable that receives the current saturation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetSaturation

```
unsigned long  
RGBSetSaturation (  
    HRGB hRGB,  
    signed long saturation )
```

Sets the Saturation.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

saturation  
The saturation to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHueMinimum

```
unsigned long  
RGBGetHueMinimum (  
    HRGB hRGB,  
    signed long *pHue )
```

Returns the minimum value that can be set for the hue.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pHue  
A pointer to the variable that receives the minimum hue.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHueMaximum

```
unsigned long  
RGBGetHueMaximum (  
    HRGB hRGB,  
    signed long *pHue )
```

Returns the maximum value that can be set for the hue.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pHue  
A pointer to the variable that receives the maximum hue.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHueDefault

```
unsigned long  
RGBGetHueDefault (  
    HRGB hRGB,  
    signed long *pHue )
```

Returns the default value for the hue.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pHue  
A pointer to the variable that receives the default hue.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetHue

```
unsigned long  
RGBGetHue (  
    HRGB hRGB,  
    signed long *pHue )
```

Returns the currently set value that for the hue.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pHue  
A pointer to the variable that receives the current hue.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetHue

```
unsigned long  
RGBSetHue (  
    HRGB hRGB,  
    signed long hue )
```

Sets the Hue.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

hue  
The hue to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetEqualisationMinimum

```
unsigned long  
RGBGetEqualisationMinimum (  
    HRGB hRGB,  
    unsigned long *pEqualisation )
```

Returns the minimum value that can be set for the equalisation.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pEqualisation**  
A pointer to the variable that receives the minimum equalisation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetEqualisationMaximum

```
unsigned long  
RGBGetEqualisationMaximum (  
    HRGB hRGB,  
    unsigned long *pEqualisation )
```

Returns the maximum value that can be set for the equalisation.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pEqualisation**  
A pointer to the variable that receives the maximum equalisation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetEqualisationDefault

```
unsigned long  
RGBGetEqualisationDefault (  
    HRGB hRGB,  
    unsigned long *pEqualisation )
```

Returns the default value for the equalisation.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pEqualisation**  
A pointer to the variable that receives the default equalisation.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetEqualisation

```
unsigned long  
RGBGetEqualisation (  
    HRGB hRGB,  
    unsigned long *pEqualisation )
```

Returns the currently set value that for the equalisation.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pEqualisation**  
A pointer to the variable that receives the current equalisation value.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetEqualisation

```
unsigned long  
RGBSetEqualisation (  
    HRGB hRGB,  
    unsigned long equalisation )
```

Sets the equalisation. Setting the equalisation can allow longer DVI cables to be used when on hardware which supports the setting.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

equalisation  
The equalisation value to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsEqualisationSupported

```
unsigned long  
RGBInputIsEqualisationSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determines whether the equalisation setting is supported on the given input.

### *Parameters*

input  
The input to query.

pBIsSupported  
The location where the result of the query is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetLinkRate

```
unsigned long  
RGBGetLinkRate (  
    HRGB hRGB,  
    PLINKRATE pValue )
```

Returns the currently set value that for the DisplayPort Link Rate.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pValue  
A pointer to the variable that receives the current LINKRATE value.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetLinkRate

```
unsigned long  
RGBSetLinkRate (  
    HRGB hRGB,  
    LINKRATE value)
```

Sets the DisplayPort Link Rate.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

value  
The LINKRATE value to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsLinkRateSupported

```
unsigned long  
RGBInputIsLinkRateSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determines whether the Link Rate setting is supported on the given input.

### *Parameters*

input  
The input to query.

pBIsSupported  
The location where the result of the query is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsSDISupported

```
unsigned long  
RGBInputIsSDISupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determines whether SDI sources are supported on the given input.

### *Parameters*

input  
The input to query.

pBIsSupported  
The location where the result of the query is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsDualLinkDVISupported

```
unsigned long  
RGBInputIsDualLinkSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determines whether dual link DVI sources are supported on the given input.

### *Parameters*

input  
The input to query.

pBIsSupported  
The location where the result of the query is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsHDMISupported

```
unsigned long  
RGBInputIsHDMISupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determines whether hdmi sources are supported on the given input.

### *Parameters*

input  
The input to query.

pBIsSupported  
The location where the result of the query is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsDisplayPortSupported

```
unsigned long  
RGBInputIsDisplayPortSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determines whether DisplayPort sources are supported on the given input.

### *Parameters*

input  
The input to query.

pBIsSupported  
The location where the result of the query is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetVideoStandard

```
unsigned long  
RGBGetVideoStandard (  
    HRGB hRGB,  
    unsigned long *pVideoStandard )
```

Returns the currently set value that for the video Standard.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pVideoStandard**  
A pointer to the variable that receives the current video Standard.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBTestVideoStandard

```
unsigned long  
RGBTestVideoStandard (  
    HRGB hRGB,  
    unsigned long videoStandard )
```

Checks whether a video standard is supported by the input.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

videoStandard  
The video standard to test.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetVideoStandard

```
unsigned long  
RGBSetVideoStandard (  
    HRGB hRGB,  
    unsigned long videoStandard )
```

Sets the video standard.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

videoStandard  
The video standard to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetFrameDroppingMinimum

```
unsigned long  
RGBGetFrameDroppingMinimum (  
    HRGB hRGB,  
    unsigned long *pFrameDropping )
```

Returns the minimum value that can be set for the number of frames that will be dropped between captures.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pFrameDropping`  
A pointer to the variable that receives the minimum frame dropping.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetFrameDroppingMaximum

```
unsigned long  
RGBGetFrameDroppingMaximum (  
    HRGB hRGB,  
    unsigned long *pFrameDropping )
```

Returns the maximum value that that can be set for the number of frames that will be dropped between captures.

### *Parameters*

`hRGB`  
The handle returned by `RGBOpenInput`.

`pFrameDropping`  
A pointer to the variable that receives the maximum frame dropping

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetFrameDroppingDefault

```
unsigned long  
RGBGetFrameDroppingDefault (  
    HRGB hRGB,  
    unsigned long *pFrameDropping )
```

Returns the default value for the number of frames that will be dropped between captures.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pFrameDropping**  
A pointer to the variable that receives the default frame dropping

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetFrameDropping

```
unsigned long  
RGBGetFrameDropping (  
    HRGB hRGB,  
    unsigned long *pFrameDropping )
```

Returns the currently set value that for the number of frames that will be dropped between captures.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pFrameDropping  
A pointer to the variable that receives the current frame dropping

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetFrameDropping

```
unsigned long  
RGBSetFrameDropping (  
    HRGB hRGB,  
    unsigned long frameDropping )
```

Sets the number of frames that will be dropped between captures.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

frameDropping  
The frame dropping to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetFrameRate

```
unsigned long  
RGBGetFrameRate (  
    HRGB hRGB,  
    unsigned long *pFrameRate )
```

Returns the current frame rate (in Hertz) of the RGB capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pFrameRate**  
A pointer to the variable that receives the current frame rate

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureRate

```
unsigned long  
RGBGetCaptureRate (  
    HRGB hRGB,  
    unsigned long *pRate )
```

Returns the currently set capture rate. The hardware will automatically drop frames, as appropriate, to maintain the specified capture rate.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pRate

A pointer to the variable that receives the current capture rate in mHz.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetCaptureRate

```
unsigned long  
RGBSetCaptureRate (  
    HRGB hRGB,  
    unsigned long rate )
```

Sets the capture rate of the current capture. The hardware will automatically drop frames, as appropriate, to maintain the specified capture rate. This function can be used as an alternative to the frame dropping functions to set an overall capture rate, e.g. 29970 mHz

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

rate

The capture rate in mHz dropping to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCroppingMinimum

```
unsigned long  
RGBGetCroppingMinimum (  
    HRGB hRGB,  
    signed long *pTop,  
    signed long *pLeft,  
    unsigned long *pWidth,  
    unsigned long *pHeight )
```

Returns the minimum cropping rectangle that can be specified.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pTop

A pointer to the variable that receives the minimum top line coordinate.

pLeft

A pointer to the variable that receives the minimum left pixel coordinate.

pWidth

A pointer to the variable that receives the minimum pixel width.

pHeight

A pointer to the variable that receives the minimum line height.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCroppingMaximum

```
unsigned long  
RGBGetCroppingMaximum (  
    HRGB hRGB,  
    signed long *pTop,  
    signed long *pLeft,  
    unsigned long *pWidth,  
    unsigned long *pHeight )
```

Returns the maximum cropping rectangle that can be specified.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pTop

A pointer to the variable that receives the maximum top line coordinate.

pLeft

A pointer to the variable that receives the maximum left pixel coordinate.

pWidth

A pointer to the variable that receives the maximum pixel width.

pHeight

A pointer to the variable that receives the maximum line height.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCroppingDefault

```
unsigned long  
RGBGetCroppingDefault (  
    HRGB hRGB,  
    signed long *pTop,  
    signed long *pLeft,  
    unsigned long *pWidth,  
    unsigned long *pHeight )
```

Returns the default cropping rectangle.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pTop

A pointer to the variable that receives the default top line coordinate.

pLeft

A pointer to the variable that receives the default left pixel coordinate.

pWidth

A pointer to the variable that receives the default pixel width.

pHeight

A pointer to the variable that receives the default line height.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCroppingOverscan

```
unsigned long  
RGBGetCroppingOverscan (  
    HRGB hRGB,  
    signed long *pTop,  
    signed long *pLeft,  
    unsigned long *pWidth,  
    unsigned long *pHeight )
```

Returns the overscan cropping rectangle for video windows.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pTop

A pointer to the variable that receives the overscan top line coordinate.

pLeft

A pointer to the variable that receives the overscan left pixel coordinate.

pWidth

A pointer to the variable that receives the overscan pixel width.

pHeight

A pointer to the variable that receives the overscan line height.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCropping

```
unsigned long  
RGBGetCropping (  
    HRGB hRGB,  
    signed long *pTop,  
    signed long *pLeft,  
    unsigned long *pWidth,  
    unsigned long *pHeight )
```

Returns the currently set cropping rectangle.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pTop

A pointer to the variable that receives the current top line coordinate.

pLeft

A pointer to the variable that receives the current left pixel coordinate.

pWidth

A pointer to the variable that receives the current pixel width.

pHeight

A pointer to the variable that receives the current line height.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBTestCropping

```
unsigned long  
RGBTestCropping (  
    HRGB hRGB,  
    signed long top,  
    signed long left,  
    unsigned long width,  
    unsigned long height )
```

Tests a cropping rectangle.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pTop

The top line coordinate to test.

pLeft

The left pixel coordinate to test.

pWidth

The pixel width to test.

pHeight

The line height to test.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetCropping

```
unsigned long  
RGBSetCropping (  
    HRGB hRGB,  
    signed long top,  
    signed long left,  
    unsigned long width,  
    unsigned long height )
```

Sets the cropping rectangle.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pTop

The top line coordinate to set.

pLeft

The left pixel coordinate to set.

pWidth

The pixel width to set.

pHeight

The line height to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsCroppingEnabled

```
unsigned long  
RGBIsCroppingEnabled (  
    HRGB hRGB,  
    unsigned long *pBEnabled )
```

Returns a value that indicates whether cropping is enabled or disabled.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pBEnabled

A pointer to the variable that receives a value that indicates whether cropping is enabled or disabled. A value of zero indicates that cropping is disabled. A value of one indicates that cropping is enabled.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBEnableCropping

```
unsigned long  
RGBEnableCropping (  
    HRGB hRGB,  
    unsigned long enabled )
```

Enables or disables cropping using the cropping rectangle specified RGBSetCropping.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

enabled  
A value of zero disables cropping. A value of one enables cropping.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will either be an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetDeinterlace

```
unsigned long  
RGBGetDeinterlace (  
    HRGB hRGB,  
    unsigned long *pDeinterlace )
```

Returns the currently set value that for deinterlace.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pDeinterlace  
A pointer to the variable that receives the current deinterlace.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetDeinterlace

```
unsigned long  
RGBSetDeinterlace (  
    HRGB          hRGB,  
    DEINTERLACE deinterlace )
```

Sets the deinterlacing for interlaced video sources.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

deinterlace  
The deinterlace value to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBPauseCapture

```
unsigned long  
RGBPauseCapture (  
    HRGB hRGB )
```

Pauses a capture. For use by applications that use RGBSetWindow.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBResumeCapture

```
unsigned long  
RGBResumeCapture (  
    HRGB hRGB )
```

Resumes a previously paused capture. For use by applications that use RGBSetWindow.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetCaptureState

```
unsigned long  
RGBGetCaptureState (  
    HRGB hRGB,  
    PCAPTURESTATE pCaptureState )
```

This function returns the current paused/running state of the RGB capture. For use by applications that use RGBSetWindow.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pCaptureState

Pointer to a variable that receives the current capture state.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetMessageDelay

```
unsigned long  
RGBGetMessageDelay (  
    HRGB hRGB,  
    signed long *pBShowMessages,  
    unsigned long *pDelay )
```

Returns the current values that indicate whether the No Signal, Invalid Signal or Error text is displayed within the window and the delay in seconds before displaying the text.

### *Parameters*

`hRGB`

The handle returned by `RGBOpenInput`.

`pBShowMessages`

Pointer to a variable that receives a value that indicates whether the text is displayed. A value of zero indicates that the text will not be displayed. A value of one indicates that the text will be displayed.

`pDelay`

Pointer to a variable that receives the value of the delay.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBSetMessageDelay

```
unsigned long  
RGBSetMessageDelay (  
    HRGB hRGB,  
    signed long BShowMessages,  
    unsigned long delay )
```

Enables or disables the displaying of and the delay (in seconds) before displaying the No Signal, Invalid Signal or Error text within the window.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

bShowMessages

A value of zero disables the displaying of the text. A value of one enables the displaying of the text.

delay

The delay, in seconds, to set before the text is displayed.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetPixelFormat

```
unsigned long  
RGBGetPixelFormat (  
    HRGB hRGB,  
    PPIXELFORMAT pPixelFormat )
```

Returns the current pixel format of the capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pPixelFormat**  
Pointer to a variable that receives the value of the current pixel format.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetPixelFormat

```
unsigned long  
RGBSetPixelFormat (  
    HRGB hRGB,  
    PIXELFORMAT pixelFormat )
```

Sets the pixel format of the capture.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

pixelFormat  
The pixel format value to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSaveCurrentFrame

```
unsigned long  
RGBSaveCurrentFrameA (  
    HRGB hRGB,  
    const char *pFileName )
```

```
unsigned long  
RGBSaveCurrentFrameW (  
    HRGB hRGB,  
    const wchar_t *pFileName )
```

Saves a single frame of RGB to a windows bitmap file. For use by applications that use RGBSetWindow whilst the capture is paused using RGBPauseCapture.

- For ANSI builds, RGBSaveCurrentFrame is defined as RGBSaveCurrentFrameA.
- For UNICODE builds, RGBSaveCurrentFrame is defined as RGBSaveCurrentFrameW.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pFileName

The full path to the file in which to save RGB data (e.g. C:\MyCaptures\Capture1.bmp). If the file does not exist it will be created. If the file already exists it will be overwritten.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetDirectDMA

```
unsigned long  
RGBGetDMADirect (  
    HRGB hRGB,  
    signed long *pbDMADirect )
```

Returns a value indicating whether the specified capture is using direct DMA to transfer data from the capture card to the display device.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pbDMADirect

Pointer to a variable that receives a value that indicates whether direct DMA is being used. A value of zero indicates that direct DMA is not being used. A value of one indicates that direct DMA is being used.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetDirectDMA

```
unsigned long  
RGBSetDMADirect (  
    HRGB hRGB,  
    signed long bDMADirect )
```

Enables or disables the direct DMA of the data from the capture card to a supported display device.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**bDMADirect**  
A value of zero disables direct DMA. A value of one enables direct DMA.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetFrameCapturedFn

```
unsigned long  
RGBSetFrameCapturedFn (  
    HRGB                hRGB,  
    PRGBFRAMECAPTUREDFN pFrameCapturedFn,  
    ULONG_PTR           userData )
```

Sets and un-sets a Frame Captured callback function. When a frame of RGB data has been captured RGBEasy uses a default handler to draw the RGB frame to the window and set up the next RGB capture. RGBSetFrameCapturedFn allows an application to specify a callback function which will be executed instead of the default handler. From within this callback an application can run the default handler by calling RGBDrawFrame.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pFrameCapturedFn

Pointer to an application defined Frame Captured callback function. If this value is NULL, the default handler is restored.

userData

Application defined context to pass to the Frame Captured function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetFrameCapturedFnEx

```

unsigned long
RGBSetFrameCapturedFnEx (
    HRGB                hRGB,
    PRGBFRAMECAPTUREDFNEX pFrameCapturedFnEx,
    ULONG_PTR           userData )
  
```

When a frame of RGB data has been captured the RGB SDK uses a default handler to draw the RGB frame to the window and set up the next RGB capture. RGBSetFrameCapturedFnEx allows an application to specify a callback function which will be executed instead of the default handler. From within this callback an application can run the default handler by calling RGBDrawFrame.

The callback provides the user with a pointer to a RGBFRAMEDATA structure which contains additional information about the frame including time stamp, frame count and buffer pointer.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pFrameCapturedFn`

Pointer to an application defined Frame Captured callback function. If this value is NULL, the default handler is restored.

`userData`

Application defined context to pass to the Frame Captured function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetMediaSampleCapturedFn

```
unsigned long  
RGBSetFrameCapturedFnEx (  
    HRGB                                hRGB,  
    PRGBMEDIASAMPLECAPTUREDFN pMediaSampleCapturedFn,  
    ULONG_PTR                    userData )
```

Sets and un-sets a video type Media Sample callback function to be executed when a video type media sample has been captured.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pMediaSampleCapturedFn`

Pointer to an application defined Media Sample callback function. If this value is NULL, the default handler is restored.

`userData`

Application defined context to pass to the Media Sample function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDrawFrame

```
unsigned long  
RGBDrawFrame (  
    HRGB hRGB )
```

Calls the default Frame Captured handler. RGBDrawFrame can only be used from within an application defined Frame Captured callback function. The SDK's default Frame Captured function draws the RGB data within the window and initiates the next RGB capture. An application that has implemented a Frame Captured callback function can call RGBDrawFrame to run the default handler. For use by applications that use RGBSetWindow.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSaveBitmap

```

unsigned long
RGBSaveBitmapA (
    HRGB                hRGB,
    LPBITMAPINFOHEADER pBitmapInfo,
    PVOID               pBitmapBits,
    const char         *pFileName )

```

```

unsigned long
RGBSaveBitmapW (
    HRGB                hRGB,
    LPBITMAPINFOHEADER pBitmapInfo,
    PVOID               pBitmapBits,
    const wchar_t      *pFileName )

```

Saves bitmap data passed to a Frame Captured Callback function to a windows bitmap file.

- For ANSI builds, RGBSaveBitmap is defined as RGBSaveBitmapA.
- For UNICODE builds, RGBSaveBitmap is defined as RGBSaveBitmapW.

### Parameters

**hRGB**  
The handle returned by RGBOpenInput.

**pBitmapInfo**  
Pointer to a BITMAPINFOHEADER structure describing the format of the bitmap data.

**pBitmapBits**  
Pointer to the bitmap data.

**pFileName**  
The full path to the file in which to save the bitmap data (e.g. C:\MyCaptures\Capture1.bmp). If the file does not exist it will be created. If the file already exists, it will be overwritten.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetModeChangedFn

```
unsigned long  
RGBSetModeChangedFn (  
    HRGB          hRGB,  
    PRGBMODECHANGEDFN pModeChangedFn,  
    ULONG_PTR     userData )
```

Sets and un-sets a Mode Changed callback function. The SDK's default Mode Changed function initialises the new video mode with default values for capture height, capture width, phase, black level, etc. RGBSetModeChangedFn allows an application to specify a callback function which will be executed after the default values have been initialised but prior to them being sent to the capture card. This allows an application to override the default values with application defined ones. Please see RGB.H for more information regarding the Mode Changed callback function.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pModeChangedFn

Pointer to an application defined Mode Changed callback function. If this value is NULL, the default handler is restored.

userData

Application-defined context to pass to the Mode Changed function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetNoSignalFn

```
unsigned long  
RGBSetNoSignalFn (  
    HRGB          hRGB,  
    PRGBNOSIGNALFN pNoSignalFn,  
    ULONG_PTR     userData )
```

Sets and un-sets a No Signal callback function. When the specified RGB capture becomes No Signal the RGB SDK uses a default handler to re-detect the video source and to display the "No Signal" message. RGBSetNoSignalFn allows an application to specify a callback function which will be executed instead of the default handler. From within this callback an application can run the default handler by calling RGBNoSignal. Please see RGB.H for more information regarding the No Signal callback function.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pNoSignalFn`

Pointer to an application defined No Signal callback function. If this value is NULL, the default handler is restored.

`userData`

Application-defined context to pass to the No Signal function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBNoSignal

```
unsigned long  
RGBNoSignal (  
    HRGB hRGB )
```

Calls the default No Signal handler. Can only be used from within an application defined No Signal callback function. The SDK's default No Signal handler sets up the capture to re-detect the video source and displays the "No Signal" text within the window. An application that has implemented a No Signal callback function can call RGBNoSignal to run the default handler.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetDrawNoSignalFn

```
unsigned long  
RGBSetDrawNoSignalFn (  
    HRGB                hRGB,  
    PRGBDRAWNOSIGNALFN pDrawFn,  
    ULONG_PTR           userData )
```

Sets and un-sets a No Signal drawing callback function. When there is No Signal the SDK's default painting function will draw a "No Signal" message in the window. This function can be used to register a custom callback to override the drawing of the message. The callback is called upon receipt of a WM\_PAINT message.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pDrawFn`

Pointer to an application defined drawing callback function. If this value is NULL, the default handler is restored.

`userData`

Application-defined context to pass to the No Signal drawing callback function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetInvalidSignalFn

```
unsigned long  
RGBSetInvalidSignalFn (  
    HRGB                hRGB,  
    PRGBINVALIDSIGNALFN pInvalidSignalFn,  
    ULONG_PTR           userData )
```

Sets and un-sets an Invalid Signal callback function that is called when the signal detected is beyond the capabilities of the hardware. When the specified RGB capture becomes Invalid Signal the RGB SDK uses a default handler to re-detect the video source and to display the "Invalid Signal" text. RGBSetInvalidSignalFn allows an application to specify a callback function which will be executed instead of the default handler. From within this callback an application can run the default handler by calling RGBInvalidSignal Please see RGB.H for more information regarding the Invalid Signal callback function.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pInvalidSignalFn

Pointer to an application defined Invalid Signal callback function. If this value is NULL, the default handler is restored.

userData

Application-defined context to pass to the Invalid Signal callback function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInvalidSignal

```
unsigned long  
RGBInvalidSignal (  
    HRGB hRGB,  
    unsigned long    horClock,  
    unsigned long    verClock )
```

Calls the default Invalid Signal handler. Can only be used from within an application defined Invalid Signal callback function. The SDK's default Invalid Signal handler sets up the capture to re-detect the video source and displays the "Invalid Signal" text within the window. An application that has implemented an Invalid Signal callback function can call RGBInvalidSignal to run the default handler.

### Parameters

**hRGB**  
The handle returned by RGBOpenInput.

**horClock**  
The horizontal frequency of the source.

**verClock**  
The vertical frequency of the source.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetDrawInvalidSignalFn

```
unsigned long  
RGBSetDrawInvalidSignalFn (  
    HRGB                hRGB,  
    PRGBDRAWINVALIDSIGNALFN pDrawFn,  
    ULONG_PTR           userData )
```

Sets and un-sets an Invalid Signal drawing callback function. When there is an Invalid Signal the SDK's default painting function will draw a message in the window showing the sync rates. This function can be used register a custom callback to override the drawing of the message. The callback is called upon receipt of a WM\_PAINT message.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pDrawFn`

Pointer to an application defined drawing callback function. If this value is NULL, the default handler is restored.

`userData`

Application-defined context to pass to the drawing callback function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetValueChangedFn

```
unsigned long  
RGBSetInvalidSignalFn (   
    HRGB                hRGB,  
    PRGBVALUECHANGEDFN pValueChangedFn,  
    ULONG_PTR           userData )
```

Sets and un-sets a Value Changed function that is called only for Vision capture cards when another capture on the same input as this capture has changed one of the following values: Horizontal Position, Horizontal Scale, Vertical Position, Capture Width, Capture Height, Brightness, Contrast, Black level, Phase, Colour Balance, Saturation, Hue, Video Standard.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pValueChangedFn`

Pointer to an application defined Value Changed callback function. If this value is NULL, the default handler is restored.

`userData`

Application-defined context to pass to the callback function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetErrorFn

```
unsigned long  
RGBSetDrawNoSignalFn (  
    HRGB          hRGB,  
    PRGBERRORFN  pErrorFn,  
    ULONG_PTR    userData )
```

Sets and un-sets an Error callback function that is called when a non-recoverable error has occurred. It is the applications responsibility to close the capture using RGBCloseInput when the Error callback function is executed.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`pErrorFn`

Pointer to an application defined error callback function. If this value is `NULL`, the default handler is restored.

`userData`

Application-defined context to pass to the callback function.

### Return Values

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBStartCapture

```
unsigned long  
RGBStartCapture (  
    HRGB hRGB )
```

Starts the RGB capture for RGB applications **not** using RGBSetWindow.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBStopCapture

```
unsigned long  
RGBStopCapture (  
    HRGB hRGB )
```

Stops the RGB capture for RGB applications **not** using RGBSetWindow.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsLiveStreamSupported

```
unsigned long  
RGBGetModeInfo (  
    unsigned long    input,  
    signed long      *pBIsSupported )
```

Determines whether LiveStream is supported on the given input.

### *Parameters*

Input

Specifies the input to query.

pBIsSupported

Pointer to a variable that receives a value that indicates whether the input is capable of LiveStream capture. A value of zero indicates that LiveStream capture is not supported. A value of one indicates LiveStream capture is supported.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetLiveStream

```
unsigned long  
RGBGetLiveStream (  
    HRGB          hRGB,  
    PLIVESTREAM pValue )
```

Returns the current LiveStream state of the capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
Pointer to a variable that receives the current LiveStream state.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetLiveStream

```
unsigned long  
RGBGetLiveStream (  
    HRGB          hRGB,  
    LIVESTREAM    value )
```

Sets the current LiveStream state of the capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
The LiveStream state to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsMediaSampleTypeSupported

```
unsigned long  
RGBInputIsMediaSampleTypeSupported (  
    unsigned long input,  
    DGCMEDIASAMPLETYPE majorType,  
    DGCMEDIASAMPLESUBTYPE subType,  
    signed long *pBIsSupported )
```

Determines if DGCMEDIASAMPLE type is supported on an input.

### Parameters

Input

Specifies the input to query.

majorType

DGCMEDIASAMPLE major type.

subType

DGCMEDIASAMPLE sub type.

pBIsSupported

Pointer to a variable that receives a value that indicates whether the DGCMEDIASAMPLE type is supported on the input. A value of zero indicates that DGCMEDIASAMPLE type is not supported. A value of one indicates DGCMEDIASAMPLE type is supported.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetModeInfo

```
unsigned long  
RGBGetModeInfo (  
    HRGB          hRGB,  
    PRGBMODEINFO pModeInfo )
```

Function that returns information about the current mode.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pModeInfo

Pointer to the structure to fill with the mode information. The size field must be initialised before this function is called.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetModeInfo

```
unsigned long  
RGBInputGetModeInfo (  
    unsigned long    input,  
    PRGBMODEINFO    pModeInfo )
```

Function that returns information about the current mode.

### *Parameters*

input  
The input number.

pModeInfo  
Pointer to the structure to fill with the mode information. The size field must be initialised before this function is called.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetDownScaling

```
unsigned long  
RGBSetDownScaling (  
    HRGB    hRGB,  
    long    bFastScaling )
```

Changes the scaling used on the RGB data when drawing on a window. If `bFastScaling` is true, and the window is smaller than the capture, the downscaling is done in hardware on the capture card before DMA. If `bFastScaling` is false, the DMA is 1:1 and the scaling is handled by the graphics system. When the capture has been set to DMA to system memory through the `RGBSetDMADirect` call then `bFastScaling` additionally enables a slower, but high quality, drawing algorithm.

### *Parameters*

`hRGB`

The handle returned by `RGBOpenInput`.

`bFastScaling`

A value of zero disables scaling on the capture card. A value of one enables scaling on the capture card.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetDownScaling

```
unsigned long  
RGBSetDownScaling (  
    HRGB    hRGB,  
    long    *pBFastScaling )
```

Returns a value indicating whether the specified RGB capture is scaling the captured data before DMA across the PCI bus.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

bFastScaling

Pointer to the variable that receives a value that indicates whether the RGB data is being scaled on the capture card. A value of zero indicates that scaling on the capture card is disabled. A value of one indicates that scaling on the capture card is enabled.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOutputSize

```
unsigned long  
RGBSetOutputSize (  
    HRGB          hRGB,  
    unsigned long  uWidth,  
    unsigned long  uHeight )
```

Sets the size of the capture for RGB applications not using RGBSetWindow. The output size is set to 1:1 with the incoming source upon a mode change.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

uWidth

The width of the buffer to capture.

uHeight

The height of the buffer to capture.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOutputSize

```
unsigned long  
RGBGetOutputSize (  
    HRGB      hRGB,  
    unsigned long *pWidth,  
    unsigned long *pHeight )
```

Returns the size of the capture for RGB applications not using RGBSetWindow.

### Parameters

hRGB

The handle returned by RGBOpenInput.

pWidth

Pointer to a variable that receives the width of the buffer being captured.

pHeight

Pointer to a variable that receives the height of the buffer being captured.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBUseOutputBuffers

```
unsigned long  
RGBUseOutputBuffers (  
    HRGB hRGB,  
    BOOL bEnableBuffers )
```

This function allows the caller to specify whether they are responsible for allocating their own buffers to capture into.

### *Parameters*

`hRGB`

The handle returned by `RGBOpenInput`.

`bEnableBuffers`

If true the capture is into a user managed buffer. If false the RGBEasy environment manages the allocation of capture buffers to match the capture settings.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBChainOutputBuffers

```
unsigned long  
RGBChainOutputBuffer (  
    HRGB          hRGB,  
    LPBITMAPINFO lpBitmapInfo,  
    PVOID         lpBuffer )
```

This function adds a user managed buffer to capture into when RGBUseOutputBuffers has been enabled. Buffers are returned to the caller through the RGBFRAMECAPTUREDFN once they have been filled. This function can be called multiple times to register a number of buffers. Buffers are returned to the user in the order in which they are chained, however the RGBFRAMECAPTUREDFN is asynchronous and maybe called more than once if the users buffer processing takes longer than the time between captures.

Once a buffer is returned through the RGBFRAMECAPTUREDFN function it is owned by the caller. The buffer will not be reused unless it is returned to the capture system with another call to RGBChainOutputBuffer. If there are no user buffers available to the capture engine when RGBUseOutputBuffers is enabled the capture will stall until a new buffer is chained.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`lpBitmapInfo`

The bitmap info describing the buffer to be filled. The BITMAPINFO is used to setup the capture used to fill the buffer. This will override the settings used for the pixel format programmed in RGBSetPixelFormat, and the buffer size programmed with RGBSetOutputSize.

`lpBuffer`

The buffer to be filled.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBChainOutputBufferEx

```

unsigned long
RGBChainOutputBufferEx (
    HRGB          hRGB,
    LPBITMAPINFO  lpBitmapInfo,
    PVOID         lpBuffer,
    BUFFERTYPE    bufferType )
  
```

This function adds a user managed buffer to capture into when RGBUseOutputBuffers has been enabled. Buffers are returned to the caller through RGBFRAMECAPTUREDFN or RGBFRAMECAPTUREDFNEX once they have been filled. This function can be called multiple times to register a number of buffers. Buffers are returned to the user in the order in which they are chained, however RGBFRAMECAPTUREDFN and RGBFRAMECAPTUREDFNEX are asynchronous and maybe called more than once if the users buffer processing takes longer than the time between captures.

Once a buffer is returned through the RGBFRAMECAPTUREDFN or RGBFRAMECAPTUREDFNEX function it is owned by the caller. The buffer will not be reused unless it is returned to the capture system with another call to RGBChainOutputBuffer. If there are no user buffers available to the capture engine when RGBUseOutputBuffers is enabled the capture will stall until a new buffer is chained.

### Parameters

`hRGB`

The handle returned by RGBOpenInput.

`lpBitmapInfo`

The bitmap info describing the buffer to be filled. The BITMAPINFO is used to setup the capture used to fill the buffer. This will override the settings used for the pixel format programmed in RGBSetPixelFormat, and the buffer size programmed with RGBSetOutputSize.

`lpBuffer`

A pointer to the buffer to be filled.

`bufferType`

The type of buffer to be chained.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBChainMediaSample

```
unsigned long  
RGBChainOutputBufferEx (  
    HRGB                hRGB,  
    PDGCMEDIASAMPLE pMediaSample )
```

This function adds a media sample to capture into when RGBUseOutputBuffers has been enabled. Buffers are returned to the caller through the RGBMEDIASAMPLECAPTUREDFN once they have been filled. This function can be called multiple times to register a number of buffers. Buffers are returned to the user in the order in which they are chained, however RGBMEDIASAMPLECAPTUREDFN is asynchronous and maybe called more than once if the users buffer processing takes longer than the time between captures.

Once a buffer is returned through the RGBMEDIASAMPLECAPTUREDFN function it is owned by the caller. The buffer will not be reused unless it is returned to the capture system with another call to RGBChainMediaSample. If there are no user buffers available to the capture engine when RGBChainMediaSample is enabled the capture will stall until a new buffer is chained.

### Parameters

**hRGB**  
The handle returned by RGBOpenInput.

**pMediaSample**  
A pointer to the media sample to be filled.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDirectGPUInit

```
unsigned long  
RGBDirectGPUInit (  
    HRGB                                hRGB,  
    PGPTRANSFERDESCRIPTOR pGpuDesc )
```

This function initializes a given graphics API for direct memory transfer between a capture card and the graphics card.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

pGpuDesc

A pointer to a structure describing the characteristics of the transfer, such as width, height, weight, colour format, byte format, number of buffers etc. For more details see GPTRANSFERDESCRIPTOR structure.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDirectGPUClose

```
unsigned long  
RGBDirectGPUClose (  
    HRGB hRGB )
```

Closes the graphics API for direct memory transfer between a capture card and the graphics card.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDirectGPUNVIDIAOp

```
unsigned long  
RGBDirectGPUNVIDIAOp (  
    HRGB          hRGB,  
    unsigned int  index,  
    NVIDIAOP      op )
```

Specifies an NVIDIA specific GPUDirect operation to perform.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

Index

The buffer index.

Op

One of the following NVIDIA GPUDirect operations to perform on the specified buffer. NVIDIA\_GPU\_COPY starts a DMA operation from the system memory to the GPU. NVIDIA\_GPU\_WAIT waits until the DMA operation has finished and blocks any DMA operation on the buffer until NVIDIA\_GPU\_END is called.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBCreateOSD

```
unsigned long  
RGBCreateOSD (  
    PHRGBOSD pHOSD )
```

This function creates an On Screen Display object which can be configured and attached to an RGB capture to enable the display of text over the image.

### *Parameters*

pOSD  
The handle to the On Screen Display object created.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDeleteOSD

```
unsigned long  
RGBCreateOSD (  
    HRGBOSD hOSD )
```

This function destroys an On Screen Display object previously created with RGBCreateOSD.

### *Parameters*

hOSD  
The handle to the On Screen Display object to be destroyed.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBAttachOSD

```
unsigned long  
RGBAttachOSD (  
    HRGB      hRGB,  
    HRGBOSD  hOSD )
```

This function attaches an OSD object to a specified RGB capture. An OSD object can only be attached to a single RGB capture at one time. On Screen Display can only be used on those captures which are associated with a window handle.

### *Parameters*

hRGB

The handle to the RGB capture which the OSD object should be associated.

hOSD

The handle to the OSD object to attach.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDetachOSD

```
unsigned long  
RGBDetachOSD (  
    HRGB      hRGB,  
    HRGBOSD  hOSD )
```

This function detaches the OSD object from the capture. Once an object has been detached it can either be deleted or attached to a different capture.

### *Parameters*

hRGB

The handle to the capture that the OSD object is attached.

hOSD

The handle to the On Screen Display object to detach from the capture.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetFirstOSD

```
unsigned long  
RGBGetFirstOSD (  
    HRGB      hRGB,  
    PHRGBOSD pHOSD )
```

This function returns a handle to the first OSD object attached to the capture.

### *Parameters*

hRGB

A handle to the capture that contains the required OSD.

pHOSD

Pointer to a variable that receives the handle of the OSD object being captured.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetNextOSD

```
unsigned long  
RGBGetNextOSD (  
    HRGB      hRGB,  
    HRGBOSD  hOSD,  
    PHRGBOSD pHOSED )
```

This function returns a handle to the next OSD object in the list of objects attached to the capture.

### *Parameters*

hRGB

A handle to the capture that contains the required OSD.

hOSD

A handle to an OSD object attached to the capture.

pHOSED

Pointer to a variable that receives the handle of the OSD object being captured.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDType

```
unsigned long  
RGBSetOSDType (  
    HRGBOSD          hOSD,  
    RGBOSD_TYPE     type )
```

This function sets the type of the OSD. Currently the only type supported is Text. Further types maybe added in future revisions. The display of an OSD object can be disabled without detaching from the capture by setting the type to Disabled.

### *Parameters*

hOSD

A handle to the OSD object.

type

The type of On Screen Display to show in the capture.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDType

```
unsigned long  
RGBGetOSDType (  
    HRGBOSD      hOSD,  
    RGBOSD_TYPE *pType )
```

This function returns the type of the OSD enabled in the object.

### *Parameters*

hOSD

A handle to the OSD object.

pType

Pointer to a variable that receives the type of the OSD object being displayed in the capture.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDScaling

```
unsigned long  
RGBSetOSDScaling (  
    HRGBOSD  hOSD,  
    BOOL      bFixedSize )
```

This function sets the scaling algorithm used by the object.

### *Parameters*

`hOSD`

A handle to the OSD object.

`bFixedSize`

Boolean value which determines the scaling algorithm. When `bFixedSize` is set to true the OSD display is drawn at its native resolution no matter what the size of the window. When `bFixedSize` is false the OSD is scaled by the ratio of the source resolution and the window size.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetOSDScaling

```
unsigned long  
RGBGetOSDScaling (  
    HRGBOSD    hOSD,  
    BOOL       *pBFixedSize )
```

This function returns the scaling algorithm used by the object.

### *Parameters*

hOSD

A handle to the OSD object.

bFixedSize

Pointer to a variable that receives a boolean describing the scaling algorithm in use in the object.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDBackground

```
unsigned long  
RGBSetOSDBackground (  
    HRGBOSD  hOSD,  
    COLORREF  cBackground,  
    BOOL      bTransparent )
```

This function sets the background colour used by the OSD when creating the object.

### *Parameters*

hOSD

A handle to the OSD object.

cBackground

The background colour to use when creating the OSD.

bTransparent

Boolean value which determines whether the background colour should be visible in the OSD. If this value is true the background colour is not drawn onto the display.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDBackground

```
unsigned long  
RGBGetOSDBackground (  
    HRGBOSD  hOSD,  
    COLORREF *pCBackground,  
    BOOL      *pBTransparent )
```

This function gets the background colour used by the OSD when creating the object.

### *Parameters*

hOSD

A handle to the OSD object.

cBackground

Pointer to a variable that receives the background colour of the OSD.

bTransparent

Pointer to a variable that receives a boolean defining whether the OSD is transparent.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDText

```
unsigned long  
RGBSetOSDText (  
    HRGBOSD  hOSD,  
    LPCTSTR  lpString )
```

This function defines the string displayed by the OSD object.

### *Parameters*

hOSD

A handle to the OSD object.

lpString

Pointer to the string to display in the OSD object.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDTextLength

```
unsigned long  
RGBGetOSDTextLength (  
    HRGBOSD      hOSD,  
    unsigned long *pNChars )
```

This function returns the number of characters in the string being displayed by the OSD.

### *Parameters*

hOSD

A handle to the OSD object.

pNChars

Pointer to the variable where the number characters should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDText

```
unsigned long  
RGBGetOSDText (   
    HRGBOSD          hOSD,   
    LPTSTR           lpString,   
    unsigned long    *pNChars )
```

This function returns the string being displayed by the OSD and the number of characters stored in the string. If the buffer provided is not large enough to store the string, an error is returned and the total number of characters are stored.

### Parameters

hOSD

A handle to the OSD object.

lpString

Pointer to the buffer where the string should be stored. If the buffer is not large enough an error will be returned.

pNChars

Pointer to the variable where the number characters should be stored. This variable must be initialised to the maximum number of characters that can be stored in the buffer before calling this function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDWrapping

```
unsigned long  
RGBSetOSDWrapping (  
    HRGBOSD  hOSD,  
    BOOL      bWrapText )
```

This function defines whether the OSD is wrapped when the text width is larger than the width of the display area.

### *Parameters*

hOSD

A handle to the OSD object.

bWrapText

A boolean specifying whether the OSD text should be wrapped onto multiple lines when the display width is larger than the string width. If this value is true the string is displayed on multiple lines. If this value is false the text will be drawn on a single line and clipped as necessary.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDBitmapFilename

```
unsigned long  
RGBSetOSDBitmapFilenameA (  
    HRGBOSD    hOSD,  
    const char *lpFilename )  
  
unsigned long  
RGBSetOSDBitmapFilenameW (  
    HRGBOSD    hOSD,  
    Const wchar_t *lpFilename )
```

This function sets the filename of the bitmap to be used for on screen display.

### Parameters

hOSD  
A handle to the OSD object.

lpFilename  
Pointer to the fully qualified filename of the bitmap.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDWrapping

```
unsigned long  
RGBGetOSDWrapping (  
    HRGBOSD    hOSD,  
    BOOL       *pBWrapping )
```

This function returned whether the OSD is wrapped when the text width is too large to fit on a single line.

### *Parameters*

hOSD

A handle to the OSD object.

pBWrapping

Pointer to the variable where the wrapping value should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDFont

```
unsigned long  
RGBSetOSDFont (  
    HRGBOSD  hOSD,  
    PLOGFONT pFont,  
    COLORREF cForeground )
```

This function defines the font and foreground colour used when displaying the OSD text.

### *Parameters*

hOSD

A handle to the OSD object.

pFont

A pointer to the Windows LOGFONT structure that describes the font to be used when drawing the OSD text.

cForeground

The colour to be used when drawing the OSD text.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDFont

```
unsigned long  
RGBGetOSDFont (  
    HRGBOSD  hOSD,  
    PLOGFONT pFont,  
    COLORREF *pCForeground )
```

This function returns the font and text colour used when drawing the OSD.

### *Parameters*

hOSD

A handle to the OSD object.

pFont

A pointer to a Windows LOGFONT structure into which the current font will be copied.

pCForeground

A pointer to a variable where the foreground colour should be copied.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDMargins

```
unsigned long
RGBSetOSDMargins (
    HRGBOSD  hOSD,
    long     top,
    long     left,
    long     bottom,
    long     right )
```

This function sets the margins in which the OSD is displayed. The margins are defined in source pixel coordinates. They are scaled dependant on the value set for OSD scaling.

### *Parameters*

hOSD

A handle to the OSD object.

top

The top line in which the OSD is drawn.

left

The left hand margin of the OSD display.

bottom

The bottom line of the OSD display.

right

The bottom line within which the OSD is drawn.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDMargins

```
unsigned long  
RGBGetOSDMargins (  
    HRGBOSD  hOSD,  
    long     *pTop,  
    long     *pLeft,  
    long     *pBottom,  
    long     *pRight )
```

This function gets the margins in which the OSD is displayed.

### *Parameters*

hOSD

A handle to the OSD object.

pTop

Pointer to a variable where the top line margin is stored.

pLeft

Pointer to a variable where the left hand margin line is stored.

pBottom

Pointer to a variable where the bottom line margin is stored

pRight

Pointer to a variable where the right hand margin is stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDAlignment

```
unsigned long  
RGBSetOSDAlignment (  
    HRGBOSD  hOSD,  
    ULONG    uHorizontal,  
    ULONG    uVertical )
```

This function sets the alignment used when drawing the OSD text.

### *Parameters*

`hOSD`

A handle to the OSD object.

`uHorizontal`

The horizontal alignment used when drawing the text. This can be one of either `RGBOSD_HOR_LEFT`, `RGBOSD_HOR_CENTRE`, or `RGBOSD_HOR_RIGHT`.

`uVertical`

The vertical alignment used when drawing the OSD text. This can be one of either `RGBOSD_VER_TOP`, `RGBOSD_VER_CENTRE`, or `RGBOSD_VER_BOTTOM`.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetOSDAlignment

```
unsigned long  
RGBGetOSDAlignment (  
    HRGBOSD  hOSD,  
    ULONG    *pUHorizontal,  
    ULONG    *pUVertical )
```

This function returns the alignment used when drawing the OSD text.

### *Parameters*

hOSD

A handle to the OSD object.

pUHorizontal

Pointer to a variable where the horizontal alignment should be stored.

pUVertical

Pointer to a variable where the horizontal alignment should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsOSDAccelerated

```
unsigned long  
RGBIsOSDAccelerated (  
    unsigned long *pBIsSupported )
```

This function returns whether the display hardware is capable of accelerating the drawing of OSD. If the display hardware supports acceleration of the OSD it will be drawn using key colouring. If it does not support acceleration then the capture must be transferred via system memory where the OSD will be drawn into the frame.

### *Parameters*

pBIsSupported  
Pointer to a boolean where the return value should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDOwnerDrawnFn

```
unsigned long  
RGBSetOSDOwnerDrawnFn (  
    HRGBOSD          hOSD,  
    PRGBOSDDRAWFN   pDrawFn,  
    ULONG_PTR        userData )
```

An owner drawn OSD allows the client application to take control of drawing the OSD in the window. This option is only supported when the display hardware is capable of accelerating the drawing of OSD. Where this is not possible the client application must capture the data into a memory buffer and draw the OSD in the RGBFRAMECAPTUREDFN callback. The OSD will be overlaid on the video capture using key colouring.

### *Parameters*

hOSD

A handle to the OSD object.

pDrawFn

A pointer to the function supplied by the client application which will handle drawing the OSD object.

userData

A data value to be passed through to the pDrawFn every time it is called.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetOSDArea

```
unsigned long  
RGBSetOSDArea (  
    HRGBOSD          hOSD,  
    long             top,  
    long             left,  
    unsigned long    width,  
    unsigned long    height )
```

This function defines the area over which the OSD will be colour keyed when the client application is using an Owner Drawn function.

### *Parameters*

hOSD  
A handle to the OSD object.

top  
The top of the OSD area.

left  
The left hand edge of the OSD area.

width  
The width of the OSD area.

height  
The height of the OSD area.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetOSDArea

```
unsigned long  
RGBGetOSDArea (  
    HRGBOSD          hOSD,  
    long              *pTop,  
    long              *pLeft,  
    unsigned long     *pWidth,  
    unsigned long     *pHeight )
```

This function returns the area over which the OSD will be colour keyed when the client application is using an Owner Drawn function.

### Parameters

hOSD

A handle to the OSD object.

pTop

The location where the top of the OSD area should be stored.

left

The location where the left hand edge of the OSD area should be stored.

width

The location where the width of the OSD area should be stored.

height

The location where the height of the OSD area should be stored.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetKeyColour

```
unsigned long  
RGBGetOSDAlignment (  
    HRGB      hRGB  
    COLORREF  cKeyColour )
```

This function sets the key colour used when drawing the OSD on systems which support accelerated OSD.

### *Parameters*

hRGB

A handle to the RGB capture.

cKeyColour

The colour to use as the key colour.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetKeyColour

```
unsigned long  
RGBGetKeyColour (  
    HRGB      hRGB,  
    COLORREF  pCKeyColour )
```

This function returns the key colour used when drawing the OSD into the capture.

### *Parameters*

hRGB

A handle to the RGB capture.

pCKeyColour

Pointer to a variable where the key colour should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetInputInfo

```
unsigned long  
RGBGetInputInfo (  
    unsigned long uInput,  
    PRGBINPUTINFO pInputInfo )
```

Returns information including firmware version numbers and identifiers for a specified input.

### *Parameters*

uInput

Specifies the input to query.

pInputInfo

Pointer to the structure to fill with the input information. The size field must be initialised before this function is called.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetRotation

```
unsigned long  
RGBGetSetRotation (  
    HRGB          hRGB,  
    ROTATIONANGLE rotationAngle )
```

Sets the angle of rotation for the capture.

### *Parameters*

hRGB  
A handle to the RGB capture.

rotationAngle  
A ROTATIONANGLE value.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetRotation

```
unsigned long  
RGBGetSetRotation (  
    HRGB          hRGB,  
    ROTATIONANGLE pRotationAngle )
```

Returns the angle of rotation for the capture.

### *Parameters*

hRGB

A handle to the RGB capture.

pRotationAngle

Pointer to the variable where the current angle of rotation should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsRotationSupported

```
unsigned long  
RGBIsRotationSupported (  
    HRGB hRGB,  
    PBOOL pBSupported )
```

Returns a value that indicates whether rotation is supported for this capture. Rotation is only supported for captures using RGBSetWindow.

### *Parameters*

hRGB

A handle to the RGB capture.

pBSupported

Pointer to the variable that indicates whether rotation is supported.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetNoSignalText

```
unsigned long  
RGBSetNoSignalText(  
    LPTSTR *pNoSignalTxt )
```

Set the background colour of the 'No Signal' message.

### *Parameters*

pNoSignalTxt  
Pointer to the string to display in the No Signal message.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetInvalidSignalText

```
unsigned long  
RGBSetInvalidSignalText(  
    LPTSTR *pInvalidSignalTxt )
```

Set the background colour of the 'Invalid Signal' message.

### *Parameters*

pNoSignalTxt  
Pointer to the string to display in the Invalid Signal message.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetNoSignalBackground

```
unsigned long  
RGBSetNoSignalBackground(  
    COLORREF cBackground )
```

Sets the Global No Signal background colour.

### *Parameters*

cBackground

The colour to be used when drawing the No Signal background.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetInvalidSignalBackground

```
unsigned long  
RGBSetInvalidSignalBackground(  
    COLORREF cBackground )
```

### *Parameters*

cBackground

The colour to be used when drawing the Invalid Signal background.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBAddModeToModeStore

```
unsigned long  
RGBAddModeToModeStore(  
    HRGB hRGB,  
    BOOL bPrivate )
```

Adds the current video mode definition to the mode store.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

bPrivate

A value to indicate if the mode should be public to all inputs in the system or private to the current input only. To set a private mode, set to TRUE. To set a public mode, set to FALSE.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBDeleteModeFromModeStore

```
unsigned long  
RGBDeleteModeFromModeStore(  
    HRGB hRGB,  
    BOOL bPrivate )
```

Deletes the current video mode definition from the mode store.

### *Parameters*

hRGB

The handle returned by RGBOpenInput.

bPrivate

A value to indicate which mode should be deleted. To delete a private mode, set to TRUE. To delete a public mode, set to FALSE.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsEDIDSupported

```
unsigned long  
RGBIsEDIDSupported (  
    signed long *pBIsSupported )
```

Returns a value that indicates whether EDIDs are supported.

### *Parameters*

*pBIsSupported*

A pointer to a variable that indicated whether EDIDs are supported. A value of 0 indicates that EDIDs are not supported. A value of 1 indicates that EDIDs are supported

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBGetEDID

```
unsigned long  
RGBGetEDID (  
    unsigned long    input,  
    char*            pEDID,  
    unsigned long    *pnChars )
```

Returns the EDID for the current input.

### Parameters

input  
Specifies the input.

pEDID  
A pointer to byte array that receives the input's EDID.

pnChars  
A pointer to a value that holds the size of the EDID to read.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetEDID

```
unsigned long  
RGBSetEDID (  
    unsigned long    input,  
    char*            pEDID,  
    unsigned long    nChars )
```

Sets the EDID for the current input.

### *Parameters*

input  
Specifies the input.

pEDID  
A pointer to byte array that contains the EDID to set.

nChars  
A value that holds the size of the EDID to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsEDIDEnabled

```
unsigned long  
RGBIsEDIDEnabled (  
    unsigned long  input,  
    unsigned long* pbEnabled )
```

Returns a value that indicates whether the input supports an EDID.

### *Parameters*

input  
Specifies the input.

pbEnabled  
A pointer to a variable that indicated whether an EDID is supported. A value of zero indicates that an EDID is not supported. A value of one indicates that an EDID is supported.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBResetEDID

```
unsigned long  
RGBResetEDID (  
    unsigned long  input )
```

Resets the input's EDID to the factory default.

### *Parameters*

input  
Specifies the input.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetMaximumPixelClock

```
unsigned long  
RGBInputGetMaximumPixelclock (  
    unsigned long  input,  
    unsigned long  *pPixelClock, )
```

Returns the maximum pixel clock allowed on the specified input.

### *Parameters*

input  
Specifies the input.

input  
Pointer to the variable that receives the maximum pixel clock value.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetSignalDetectionMethod

```
unsigned long  
RGBSetSignalDetectionMethod (  
    unsigned long  input,  
    SIGNALDETECT  method )
```

Set the signal detection method.

### *Parameters*

uInput  
Specifies the input to query.

method  
The method to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetSignalDetectionMethod

```
unsigned long  
RGBGetSignalDetectionMethod (  
    unsigned long  input,  
    PSIGNALDETECT  pMethod )
```

Get the current signal detection method

### *Parameters*

Input

Specifies the input to query.

pMethod

Receives the current method.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsSignalDetectionMethodSupported

```
unsigned long  
RGBIsSignalDetectionMethodSupported (  
    unsigned long    input,  
    signed long      *pBIsSupported )
```

Determine whether signal selecting the signal detection method is supported.

### *Parameters*

Input  
Specifies the input to query.

pBIsSupported  
Boolean value that indicates whether the method is supported.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetColourDomain

```
unsigned long  
RGBSetColourDomain (  
    HRGB                hRGB,  
    COLOURDOMAINDETECT value )
```

Set the colour domain to use.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

value  
The colour domain to use.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourDomain

```
unsigned long  
RGBGetColourDomain (  
    HRGB          hRGB,  
    PCOLOURDOMAINDETECT pValue )
```

Get the current colour domain in use.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
Pointer to the variable where the current colour domain should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourDomainDefault

```
unsigned long  
RGBGetColourDomainDefault (  
    HRGB          hRGB,  
    PCOLOURDOMAINDETECT pValue )
```

Get the default colour domain of the source.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
Pointer to the variable where the default colour domain should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetColourDomainDetect

```
unsigned long  
RGBGetColourDomainDetect (  
    HRGB                hRGB,  
    PCOLOURDOMAINDETECT pValue )
```

Gets the on-the-wire detected colour domain of the source.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
Pointer to the variable where the default colour domain should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetPixelGamut

```
unsigned long  
RGBGetPixelGamut (  
    HRGB          hRGB,  
    PPIXELGAMUT pValue )
```

Returns the current pixel gamut of the capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
Pointer to the variable where the current pixel gamut should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetPixelGamut

```
unsigned long  
RGBSetPixelGamut (  
    HRGB          hRGB,  
    PIXELGAMUT   value )
```

Sets the pixel gamut of the capture.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

value  
The pixel gamut to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBGetPixelRange

```
unsigned long  
RGBGetPixelRange (  
    HRGB          hRGB,  
    PPIXELRANGE  pValue)
```

Sets the pixel range of the capture.

### *Parameters*

**hRGB**  
The handle returned by RGBOpenInput.

**pValue**  
Pointer to the variable where the current pixel range should be stored.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBSetPixelRange

```
unsigned long  
RGBSetPixelRange (  
    HRGB          hRGB,  
    PIXELRANGE   value)
```

Sets the pixel range of the capture.

### *Parameters*

hRGB  
The handle returned by RGBOpenInput.

value  
The pixel range to set.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsGangingSupported

```
unsigned long  
RGBInputIsGangingSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determine if an input is capable of being ganged.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports ganging.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsGangingTypeSupported

```
unsigned long  
RGBGetPixelRange (  
    unsigned long input,  
    RGBGANG_TYPE type,  
    signed long *pBIsSupported )
```

Tests if a particular ganging type is supported by an input.

### *Parameters*

input  
Specifies the input to query.

type  
The ganging type to test.

pValue  
If TRUE, the input supports ganging.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetGangingType

```
unsigned long  
RGBInputGetGangingType (  
    unsigned long input,  
    RGBGANG_TYPE *pValue)
```

Gets the current selected ganging type of the input.

### *Parameters*

*input*  
Specifies the input to query.

*pValue*  
Pointer to the variable where the current ganging type should be stored.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBInputSetGangingType

```
unsigned long  
RGBInputSetGangingType (  
    unsigned long input,  
    RGBGANG_TYPE value)
```

Sets the ganging type for the specified input.

### *Parameters*

*input*  
Specifies the input to query.

*value*  
The ganging type to set.

### *Return Values*

If the function succeeds, the return value is `RGBERROR_NO_ERROR`. If the function fails, the return value will be either an error code defined in `RGBERROR.H` or a standard windows error code.

## RGBIsHDCPSupported

```
unsigned long  
RGBInputIsGangingSupported (  
    signed long    *pBIsSupported )
```

Determine if the system is capable of capturing and displaying HDCP protected content.

### *Parameters*

pBIsSupported  
If TRUE, the system supports HDCP.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsHDCPSupported

```
unsigned long  
RGBInputIsGangingSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determine if an input is capable of capturing HDCP protected content.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports HDCP.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputIsHDCPProtected

```
unsigned long  
RGBInputIsGangingSupported (  
    unsigned long  input,  
    signed long    *pBProtected)
```

Determine if HDCP protected content is connected to an input.

### *Parameters*

input  
Specifies the input to query.

pBProtected  
If TRUE, HDCP protected content is connected to an input.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetHDCPEnabled

```
unsigned long  
RGBInputGetHDCPEnabled (  
    unsigned long input,  
    signed long *pBEnabled)
```

Determines if an inputs HDCP decoder is enabled.

### *Parameters*

input  
Specifies the input to query.

pBEnabled  
If TRUE, the HDCP decoder is enabled.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputSetHDCPEnabled

```
unsigned long  
RGBInputSetHDCPEnabled (  
    unsigned long input,  
    signed long  enabled)
```

Enables and disables an inputs HDCP decoder.

### *Parameters*

input  
Specifies the input to query.

enabled  
TRUE to enable the HDCP decoder, FALSE to disable it.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBEnableInputSnapshot

```
unsigned long  
RGBEnableInputSnapshot (  
    signed long bEnable)
```

Enables and disables the current Input Snapshot.

### *Parameters*

enabled  
TRUE to enable the snapshot, FALSE to disable it.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBIsInputSnapshotEnabled

```
unsigned long  
RGBIsInputSnapshotEnabled (  
    signed long    * pBEnabled)
```

Determine if an Input Snapshot is enabled for the system.

### *Parameters*

pBEnabled  
If TRUE, an Input Snapshot is enabled.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetStatus

```
unsigned long  
RGBInputGetStatus (  
    unsigned long  input,  
    RGBINPUTSTATUS *pStatus)
```

Determine if an input is available to use.

### *Parameters*

input  
Specifies the input to query.

pStatus  
A RGBINPUTSTATUS value describing the availability if an input.

### *Return Values*

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBInputGetBitmapSnapshot

```
unsigned long  
RGBInputGetBitmapSnapshot (  
    unsigned long          input,  
    LPBITMAPINFOHEADER    pBitmapInfo,  
    void                  *pBitmapBits,  
    PRGBINPUTBITMAPSNAPSHOTFN pSnapshotFn,  
    ULONG_PTR             userData)
```

Returns a bitmap image snapshot of an input

### Parameters

input

Specifies the input.

pBitmapInfo

Pointer to a BITMAPINFOHEADER structure describing the format of the bitmap bits.

pBitmapBits

Pointer to the bitmap bits.

pSnapshotFn

Pointer to an application defined Input Snapshot callback function.

userData

Application-defined context to pass to the Input Snapshot callback function.

### Return Values

If the function succeeds, the return value is RGBERROR\_NO\_ERROR. If the function fails, the return value will be either an error code defined in RGBERROR.H or a standard windows error code.

## RGBAudioIsAudioSupported

```
unsigned long  
RGBAudioIsAudioSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determine if an input is audio capable.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports audio.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioIsHDMISupported

```
unsigned long  
RGBAudioIsHDMISupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determine HDMI audio capability.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports the capability.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioIsSDISupported

```
unsigned long  
RGBAudioIsSDISupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determine SDI audio capability.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports the capability.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioIsAnalogueSupported

```
unsigned long  
RGBAudioIsAnalogueSupported (  
    unsigned long  input,  
    signed long    *pBIsSupported )
```

Determine analogue audio capability.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports the capability

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetCapabilities

```
unsigned long  
RGBAudioGetCapabilities (  
    unsigned long  input,  
    unsigned long  index,  
    PAUDIOCAPS    pCaps )
```

Get the index specific capability parameters.

Use RGBAudioGetCapabilitiesCount to return the index total.

### *Parameters*

input

Specifies the input to query.

index

Index member for capability.

pCaps

Pointer to the structure to fill with the format information. The size field must be initialised before this function is called.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetCapabilitiesCount

```
unsigned long  
RGBAudioGetCapabilitiesCount (  
    unsigned long input,  
    unsigned long *pCount )
```

Get the count for available format capabilities including SampleRate.

### *Parameters*

input  
Specifies the input to query.

pCount  
Count of supported capabilities.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetCapabilities

```
unsigned long  
RGBAudioSetCapabilities (  
    unsigned long  input,  
    unsigned long  index )
```

Set the index specific capability parameters.

Use RGBAudioGetCapabilitiesCount to return the index total.Parameters

input  
Specifies the input to set.

index  
Index member for capability returned from RGBAudioGetCapabilities.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioOpenInput

```
unsigned long  
RGBAudioOpenInput (  
    PAUDIOCAPTUREDFN    pNotifyFn,  
    ULONG_PTR           pNotifyArg,  
    unsigned long       input,  
    PHAUDIO             phAudio )
```

Opens an Audio capture on the specified input.

### Parameters

`pNotifyFn`

Points to a user declared callback function, see Audio.H for further details.

`pNotifyArg`

User defined argument that can be passed into the `pNotifyFn` by the notify thread.

`input`

Specifies the input to open. The input must be a value in the range 0 to ( `numberOfInputs - 1` ). The number of inputs can be obtained by calling `RGBGetNumberOfInputs`.

`phAudio`

Pointer to a variable that receives the handle that identifies the Audio capture.

### Return Values

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioCloseInput

```
unsigned long  
RGBAudioCloseInput (  
    HAUDIO hAudio )
```

Closes an Audio capture.

### *Parameters*

hAudio

The Audio capture handle to be closed. hAudio is not valid after this call and must not be used again.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioChainOutputBuffer

```
unsigned long  
RGBAudioChainOutputBuffer (  
    HAUDIO          hAudio,  
    unsigned long   ulNumberBytes,  
    unsigned long   ulBufferPitch,  
    PVOID           lpBuffer )
```

This function adds a user managed buffer into the driver. Multiple buffers can be used with additional calls to this function. When a filled buffer is returned to through the `FrameCaptured` callback it must be reinserted with another call to this function for it to be refilled.

### Parameters

`hAudio`  
The Audio capture handle.

`ulNumberBytes`  
Size of the buffer in bytes.

`ulBufferPitch`  
Buffer pitch in bytes.

`lpBuffer`  
The buffer to be filled.

### Return Values

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioLoadOutputBuffer

```
unsigned long  
RGBAudioLoadOutputBuffer (  
    HAUDIO          hAudio,  
    unsigned long   ulNumberBytes,  
    unsigned long   ulBufferPitch,  
    unsigned long   *pBytesWritten,  
    PVOID           lpBuffer )
```

This function fills a user managed buffer with available data in the driver. Zero is inserted as data if no audio is available.

### Parameters

`hAudio`

The Audio capture handle.

`ulNumberBytes`

Size of the buffer in bytes.

`ulBufferPitch`

Buffer pitch in bytes.

`pBytesWritten`

For example, if `lpBuffer` is large this number is the limited byte count available in the drivers cyclic buffer for reading.

`lpBuffer`

The buffer to be filled.

### Return Values

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioReleaseOutputBuffers

```
unsigned long  
RGBAudioReleaseOutputBuffers (  
    HAUDIO    hAudio )
```

This function removes a user managed buffer in the driver. Multiple buffers lodged within the driver are removed. The buffer does not return via the `FrameCaptured` callback.

### *Parameters*

`hAudio`  
The Audio capture handle.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetState

```
unsigned long  
RGBAudioSetState (  
    HAUDIO          hAudio,  
    AUDIOCAPTURESTATE state)
```

Sets the current Audio capture state.

Purpose: Sets the current capture state listed within `AudioEasy.H`.

### *Parameters*

`hAudio`  
The Audio capture handle.

`state`  
Requested current state.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetState

```
unsigned long  
RGBAudioGetState (  
    HAUDIO                hAudio,  
    PAUDIOCAPTURESTATE   pState)
```

Gets the current Audio capture state.

Purpose: Determine the current capture state listed within `AudioEasy.H`.

### *Parameters*

`hAudio`  
The Audio capture handle.

`pState`  
Pointer to the current state.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetDigitalChannelPair

```
unsigned long  
RGBAudioSetDigitalChannelPair (  
    unsigned long uInput,  
    unsigned long channel )
```

Select which channel pair to use.

### *Parameters*

uInput  
Specifies the input to query.

channel  
The pair to select

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetDigitalChannelPair

```
unsigned long  
RGBAudioGetDigitalChannelPair (  
    unsigned long uInput,  
    unsigned long *pChannel )
```

Get the current channel pair.

### *Parameters*

uInput  
Specifies the input to query.

pChannel  
The current pair.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetDigitalMute

```
unsigned long  
RGBAudioSetDigitalMute (  
    unsigned long uInput,  
    unsigned long mute )
```

Toggle mute on/off.

### *Parameters*

uInput  
Specifies the input to query.

mute  
Boolean value indicating whether the audio is mute or not.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetDigitalMute

```
unsigned long  
RGBAudioGetDigitalMute (  
    unsigned long uInput,  
    unsigned long *pMute )
```

Determine whether audio is mute or not.

### *Parameters*

uInput  
Specifies the input to query.

pMute  
Pointer to a buffer that will receive the Boolean value.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetAnalogueBalancedGain

```
unsigned long  
RGBAudioSetAnalogueBalancedGain (  
    unsigned long uInput,  
    signed long gain )
```

Set the gain.

### *Parameters*

uInput  
Specifies the input to query.

gain  
value of gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedGain

```
unsigned long  
RGBAudioGetAnalogueBalancedGain (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the current gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the current gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedGainMinimum

```
unsigned long  
RGBAudioGetAnalogueBalancedGainMinimum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the minimum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the minimum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedGainMaximum

```
unsigned long  
RGBAudioGetAnalogueBalancedGainMaximum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the maximum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the maximum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedGainDefault

```
unsigned long  
RGBAudioGetAnalogueBalancedGainDefault (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the default gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the default gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedGainScale

```
unsigned long  
RGBAudioGetAnalogueBalancedGainScale (  
    unsigned long uInput,  
    unsigned long *pScale )
```

Get the gain scaling.

### *Parameters*

uInput  
Specifies the input to query.

pScale  
the gain scaling factor.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetAnalogueBalancedGainBoost

```
unsigned long  
RGBAudioSetAnalogueBalancedGainBoost (  
    unsigned long uInput,  
    unsigned long boost )
```

Toggle whether boost is on/off.

### *Parameters*

uInput  
Specifies the input to query.

boost  
Boolean value indicating whether boost should be on or not.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedGainBoost

```
unsigned long  
RGBAudioGetAnalogueBalancedGainBoost (  
    unsigned long uInput,  
    unsigned long *pBoost )
```

indicates whether boost is on or not.

### *Parameters*

uInput  
Specifies the input to query.

pBoost  
Boolean value indicating the current boost state

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetAnalogueBalancedMute

```
unsigned long  
RGBAudioSetAnalogueBalancedMute (  
    unsigned long uInput,  
    unsigned long mute )
```

Toggle mute on/off.

### *Parameters*

uInput  
Specifies the input to query.

mute  
Boolean value indicating whether the audio is mute or not.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueBalancedMute

```
unsigned long  
RGBAudioGetAnalogueBalancedMute (  
    unsigned long uInput,  
    unsigned long *pMute )
```

Determine whether audio is mute or not.

### *Parameters*

uInput  
Specifies the input to query.

pMute  
Pointer to a buffer that will receive the Boolean value.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetAnalogueUnbalancedGain

```
unsigned long  
RGBAudioSetAnalogueUnbalancedGain (  
    unsigned long uInput,  
    signed long gain )
```

Set the gain.

### *Parameters*

uInput  
Specifies the input to query.

gain  
value of gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueUnbalancedGain

```
unsigned long  
RGBAudioGetAnalogueUnbalancedGain (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the current gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the current gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueUnbalancedGainMinimum

```
unsigned long  
RGBAudioGetAnalogueUnbalancedGainMinimum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the minimum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the minimum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueUnbalancedGainMaximum

```
unsigned long  
RGBAudioGetAnalogueUnbalancedGainMaximum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the maximum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the maximum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueUnbalancedGainDefault

```
unsigned long  
RGBAudioGetAnalogueUnbalancedGainDefault (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the default gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the default gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueUnbalancedGainScale

```
unsigned long  
RGBAudioGetAnalogueUnbalancedGainScale (  
    unsigned long uInput,  
    unsigned long *pScale )
```

Get the gain scaling.

### *Parameters*

uInput  
Specifies the input to query.

pScale  
the gain scaling factor.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetAnalogueUnbalancedMute

```
unsigned long  
RGBAudioSetAnalogueUnbalancedMute (  
    unsigned long uInput,  
    unsigned long mute )
```

Toggle mute on/off.

### *Parameters*

uInput  
Specifies the input to query.

mute  
Boolean value indicating whether the audio is mute or not.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetAnalogueUnbalancedMute

```
unsigned long  
RGBAudioGetAnalogueUnbalancedMute (  
    unsigned long uInput,  
    unsigned long *pMute )
```

Determine whether audio is mute or not.

### *Parameters*

uInput  
Specifies the input to query.

pMute  
Pointer to a buffer that will receive the Boolean value.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetADCGain

```
unsigned long  
RGBAudioSetADCGain (  
    unsigned long uInput,  
    signed long gain )
```

Set the gain.

### *Parameters*

uInput  
Specifies the input to query.

gain  
value of gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetADCGain

```
unsigned long  
RGBAudioGetADCGain (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the current gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the current gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetADCGainMinimum

```
unsigned long  
RGBAudioGetADCGainMinimum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the minimum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the minimum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetADCGainMaximum

```
unsigned long  
RGBAudioGetADCGainMaximum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the maximum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the maximum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetADCGainDefault

```
unsigned long  
RGBAudioGetADCGainDefault (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the default gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the default gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetADCGainScale

```
unsigned long  
RGBAudioGetADCGainScale (  
    unsigned long uInput,  
    unsigned long *pScale )
```

Get the gain scaling.

### *Parameters*

uInput  
Specifies the input to query.

pScale  
the gain scaling factor.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetADCMute

```
unsigned long  
RGBAudioSetADCMute (  
    unsigned long uInput,  
    unsigned long mute )
```

Toggle mute on/off.

### *Parameters*

uInput  
Specifies the input to query.

mute  
Boolean value indicating whether the audio is mute or not.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetADCMute

```
unsigned long  
RGBAudioGetADCMute (  
    unsigned long uInput,  
    unsigned long *pMute )
```

Determine whether audio is mute or not.

### *Parameters*

uInput  
Specifies the input to query.

pMute  
Pointer to a buffer that will receive the Boolean value.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioIsLineOutSupported

```
unsigned long  
RGBAudioIsLineOutSupported (  
    unsigned long uInput,  
    signed long *pBIsSupported )
```

Determine line-out audio capability.

### *Parameters*

input  
Specifies the input to query.

pBIsSupported  
If TRUE, the input supports the capability

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetLineOutSource

```
unsigned long  
RGBAudioSetLineOutSource (  
    unsigned long uInput,  
    AUDIOCAPTURESOURCE source )
```

Set the line out source, see audio.h

### *Parameters*

uInput  
Specifies the input to query.

source  
the source to use.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutSource

```
unsigned long  
RGBAudioGetLineOutSource (  
    unsigned long uInput,  
    AUDIOCAPTURESOURCE *pSource )
```

Get the line out source, see audio.h

### *Parameters*

uInput  
Specifies the input to query.

pSource x  
The current source.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetLineOutGain

```
unsigned long  
RGBAudioSetLineOutGain (  
    unsigned long uInput,  
    signed long gain )
```

Set the gain.

### *Parameters*

uInput  
Specifies the input to query.

gain  
value of gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutGain

```
unsigned long  
RGBAudioGetLineOutGain (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the current gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the current gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutGainMinimum

```
unsigned long  
RGBAudioGetLineOutGainMinimum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the minimum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the minimum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutGainMaximum

```
unsigned long  
RGBAudioGetLineOutGainMaximum (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the maximum gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the maximum gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutGainDefault

```
unsigned long  
RGBAudioGetLineOutGainDefault (  
    unsigned long uInput,  
    signed long *pGain )
```

Get the default gain.

### *Parameters*

uInput  
Specifies the input to query.

pGain  
the default gain.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutGainScale

```
unsigned long  
RGBAudioGetLineOutGainScale (  
    unsigned long uInput,  
    unsigned long *pScale )
```

Get the gain scaling.

### *Parameters*

uInput  
Specifies the input to query.

pScale  
the gain scaling factor.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioSetLineOutMute

```
unsigned long  
RGBAudioSetLineOutMute (  
    unsigned long uInput,  
    unsigned long mute )
```

Toggle mute on/off.

### *Parameters*

uInput  
Specifies the input to query.

mute  
Boolean value indicating whether the audio is mute or not.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBAudioGetLineOutMute

```
unsigned long  
RGBAudioGetLineOutMute (  
    unsigned long uInput,  
    unsigned long *pMute )
```

Determine whether audio is mute or not.

### *Parameters*

uInput  
Specifies the input to query.

pMute  
Pointer to a buffer that will receive the Boolean value.

### *Return Values*

If the function succeeds, returns either zero if successful or an appropriate error value.

## RGBEasy Callback Functions

## RGBFRAMECAPTUREDFN

```
typedef void (RGBCBKAPI RGBFRAMECAPTUREDFN) (
    HWND          hWnd,
    HRGB          hRGB,
    LPBITMAPINFOHEADER pBitmapInfo,
    void          *pBitmapBits,
    ULONG_PTR     userData );
typedef RGBFRAMECAPTUREDFN *PRGBFRAMECAPTUREDFN;
```

The RGBFRAMECAPTUREDFN function is an application-defined callback function used with RGBSetFrameCapturedFn. The RGB SDK calls this function when a frame of RGB data has been captured. The pBitmapInfo and pBitmapBits pointers will be NULL when direct DMA has been enabled. This is because the RGB data is not available as it has been transferred by DMA directly from the capture card to the Display card.

### Parameters

`hWnd`

The window handle supplied to RGBSetWindow or NULL if not using a window.

`hRGB`

The handle returned by RGBOpenInput.

`pBitmapInfo`

Pointer to BITMAPINFOHEADER describing the bitmap or NULL if Direct DMA has been enabled.

`pBitmapBits`

Pointer to the bitmap bits or NULL if Direct DMA has been enabled.

`userData`

Application supplied context.

### Return Values

None.

## RGBFRAMECAPTUREDFNEX

```
typedef void (RGBCBKAPI RGBFRAMECAPTUREDFNEX) (  
    HWND          hWnd,  
    HRGB          hRGB,  
    PRGBFRAMEDATA pFrameData,  
    ULONG_PTR     userData );  
typedef RGBFRAMECAPTUREDFNEX *PRGBFRAMECAPTUREDFNEX;
```

The RGBFRAMECAPTUREDFNEX function is an application-defined callback function used with RGBSetFrameCapturedFnEx. The RGB SDK calls this function when a frame of RGB data has been captured.

### *Parameters*

*hWnd*

The window handle supplied to RGBSetWindow or NULL if not using a window.

*hRGB*

The handle returned by RGBOpenInput.

*pFrameData*

Pointer to a RGBFRAMEDATA structure describing the captured frame.

*userData*

Application supplied context.

### *Return Values*

None.

## RGBMEDIASAMPLECAPTUREDFN

```
typedef void (RGBCBKAPI RGBMEDIASAMPLECAPTUREDFN) (  
    HWND          hWnd,  
    HRGB          hRGB,  
    PDGCMEDIASAMPLE pMediaSample,  
    ULONG_PTR     userData );  
typedef RGBMEDIASAMPLECAPTUREDFN *PRGBMEDIASAMPLECAPTUREDFN;
```

The RGBMEDIASAMPLECAPTUREDFN function is an application-defined callback function used with RGBSetMediaSampleCapturedFn. The RGB SDK calls this function when a media sample has been captured.

### *Parameters*

`hWnd`

The window handle supplied to RGBSetWindow or NULL if not using a window.

`hRGB`

The handle returned by RGBOpenInput.

`pMediaSample`

Pointer to a DGCMEDIASAMPLE.

`userData`

Application supplied context.

### *Return Values*

None.

## RGBMODECHANGEDFN

```
typedef void (RGBCBKAPI RGBMODECHANGEDFN) (  
    HWND                hWnd,  
    HRGB                hRGB,  
    PRGBMODECHANGEDINFO pModeChangedInfo,  
    ULONG_PTR          userData );  
typedef RGBMODECHANGEDFN* PRGBMODECHANGEDFN;
```

The RGBMODECHANGEDFN function is an application-defined callback function used with RGBSetModeChangedFn. The RGB SDK calls this function when a new video mode has been detected.

### *Parameters*

`hWnd`

The window handle supplied to RGBSetWindow or NULL if not using a window.

`hRGB`

The handle returned by RGBOpenInput.

`pModeChangedInfo`

Pointer to a RGBMODECHANGEDINFO structure describing the new video mode.

`userData`

Application supplied context.

### *Return Values*

None.

## RGBNOSIGNALFN

```
typedef void (RGBCBKAPI RGBNOSIGNALFN) (  
    HWND                hWnd,  
    HRGB                hRGB,  
    ULONG_PTR          userData );  
typedef RGBNOSIGNALFN *PRGBNOSIGNALFN;
```

The RGBNOSIGNALFN function is an application-defined callback function used with RGBSetNoSignalFn. The RGB SDK calls this function when a video signal cannot be detected.

### *Parameters*

hWnd

The window handle supplied to RGBSetWindow or NULL if not using a window.

hRGB

The handle returned by RGBOpenInput.

userData

Application supplied context.

### *Return Values*

None.

## RGBDRAWNOSIGNALFN

```
typedef void (RGBCBKAPI RGBDRAWNOSIGNALFN) (  
    HWND          hWnd,  
    HRGB          hRGB,  
    HDC           hDC,  
    ULONG_PTR     userData );  
typedef RGBNOSIGNALFN *PRGBNOSIGNALFN;
```

The RGBDRAWNOSIGNALFN function is an application defined callback function used with RGBSetDrawNoSignalFn. The RGB SDK calls this function to draw the window when a video signal cannot be detected.

### *Parameters*

hWnd

The window handle supplied to RGBSetWindow or NULL if not using a window.

hRGB

The handle returned by RGBOpenInput.

hDC

The handle to the device context in which to draw the No Signal message.

userData

Application supplied context.

### *Return Values*

None.

## RGBINVALIDSIGNALFN

```
typedef void (RGBCBKAPI RGBINVALIDSIGNALFN) (  
    HWND            hWnd,  
    HRGB            hRGB,  
    unsigned long   horClock,  
    unsigned long   verClock,  
    ULONG_PTR       userData );  
typedef RGBINVALIDSIGNALFN *PRGBINVALIDSIGNALFN;
```

The RGBINVALIDSIGNALFN function is an application-defined callback function used with RGBSetInvalidSignalFn. The RGB SDK calls this function when a video signal beyond the capabilities of the hardware is detected.

### Parameters

hWnd

The window handle supplied to RGBSetWindow or NULL if not using a window.

hRGB

The handle returned by RGBOpenInput.

horClock

Horizontal refresh rate of the video source.

verClock

Vertical refresh rate of the video source.

userData

Application supplied context.

### Return Values

None.

## RGBDRAWINVALIDSIGNALFN

```
typedef void (RGBCBKAPI RGBDRAWINVALIDSIGNALFN) (  
    HWND          hWnd,  
    HRGB          hRGB,  
    HDC           hDC,  
    unsigned long horClock,  
    unsigned long verClock,  
    ULONG_PTR     userData );  
typedef RGBDRAWINVALIDSIGNALFN *PRGBDRAWINVALIDSIGNALFN;
```

The RGBDRAWINVALIDSIGNALFN function is an application defined callback function used with RGBSetDrawInvalidSignalFn. The RGB SDK calls this function to draw the window when a video signal is invalid.

### *Parameters*

hWnd

The window handle supplied to RGBSetWindow or NULL if not using a window.

hRGB

The handle returned by RGBOpenInput.

hDC

The handle to the device context in which to draw the Invalid Signal message.

horClock

Horizontal refresh rate of the video source.

verClock

Vertical refresh rate of the video source.

userData

Application supplied context.

### *Return Values*

None

## RGBERRORFN

```
typedef void (RGBCBKAPI RGBERRORFN) (  
    HWND            hWnd,  
    HRGB            hRGB,  
    unsigned long   error  
    ULONG_PTR       userData,  
    unsigned long   *pReserved );  
typedef RGBVALUECHANGEDFN *PRGBVALUECHANGEDFN;
```

The RGBERRORFN function is an application defined callback function used with RGBsetErrorFn. The RGB SDK calls this function when an unrecoverable error occurs.

### Parameters

hWnd

The window handle supplied to RGBSetWindow or NULL if not using a window.

hRGB

The handle returned by RGBOpenInput.

error

An error code.

userData

Application supplied context.

pReserved

Do not use.

### Return Values

None

## RGBVALUECHANGEDFN

```
typedef void (RGBCBKAPI RGBVALUECHANGEDFN) (  
    HWND                hWnd,  
    HRGB                hRGB,  
    PRGBVALUECHANGEDINFO pValueChangedInfo,  
    ULONG_PTR           userData );  
typedef RGBVALUECHANGEDFN *PRGBVALUECHANGEDFN;
```

The RGBVALUECHANGEDFN function is an application-defined callback function used with RGBSetModeValueChangedFn. The RGB SDK calls this function when a value new video mode has been detected.

### Parameters

`hWnd`

The window handle supplied to RGBSetWindow or NULL if not using a window.

`hRGB`

The handle returned by RGBOpenInput.

`pValueChangedInfo`

Pointer to a RGBVALUECHANGEDINFO structure describing the changes to the current video mode.

`userData`

Application supplied context.

### Return Values

None

## RGBOSDDRAWFN

```
typedef void (RGBCBKAPI RGBOSDDRAWFN) (  
    HWND          hWnd,  
    HRGB         hOSD,  
    HDC          hDC,  
    ULONG_PTR    userData );  
typedef RGBOSDDRAWFN *PRGBOSDDRAWFN;
```

The RGBOSDDRAWFN function is an application-defined callback function used with RGBSetOSDOwnerDrawnFn. The RGB SDK calls this function when the owner drawn OSD object must be redrawn.

### *Parameters*

hWnd

The window handle supplied to RGBSetWindow.

hOSD

The handle to the OSD object that needs to be redrawn.

hDC

The device context which can be used in for drawing the OSD.

userData

Application supplied context.

### *Return Values*

None

## RGBINPUTBITMAPSNAPSHOTFN

```
typedef void (RGBCBKAPI RGBINPUTBITMAPSNAPSHOTFN) (  
    unsigned long    input,  
    unsigned long    error,  
    PRGBFRAMEDATA   pFrameData,  
    ULONG_PTR       userData);  
typedef RGBINPUTBITMAPSNAPSHOTFN *PRGBINPUTBITMAPSNAPSHOTFN;
```

The `RGBINPUTBITMAPSNAPSHOTFN` function is an application-defined callback function used with `RGBInputGetBitmapSnapshot`.

### *Parameters*

`input`

The input number.

`error`

Zero if the snapshot was successful or an appropriate error value.

`pFrameData`

Pointer to a `RGBFRAMEDATA` structure describing the snapshot frame.

`userData`

Application supplied context.

### *Return Values*

None

## RGBEasy Enumerations

## CAPTURECARD

```
typedef enum _CAPTURECARD
{
    RGB_CAPTURECARD_DGC103,
    RGB_CAPTURECARD_DGC133,
} CAPTURECARD, *PCAPTURECARD;
```

List of capture card types returned by RGBGetCaptureCard.

RGB\_CAPTURECARD\_DGC103  
PCI type capture card.

RGB\_CAPTURECARD\_DGC133  
PCIe type capture card.

## RGBDEVICETYPE

```
typedef enum _RGBDEVICETYPE
{
    DEVICETYPE133,
    DEVICETYPE139,
    DEVICETYPE144,
    DEVICETYPE150,
    DEVICETYPE151,
    DEVICETYPE139S,
    DEVICETYPE150S,
    DEVICETYPE151S,
    DEVICETYPE153,
    DEVICETYPE154,
    DEVICETYPE159,
    DEVICETYPE161,
    DEVICETYPE165,
    DEVICETYPE167,
    DEVICETYPE168,
    DEVICETYPE179,
    DEVICETYPE182,
    DEVICETYPE184,
    DEVICETYPE186,
    DEVICETYPE199,
    DEVICETYPE200,
    DEVICETYPE201,
    DEVICETYPE204,
    DEVICETYPE205,
    DEVICETYPE211,
} RGBDEVICETYPE, *PRGBDEVICETYPE;
```

List of device types returned by RGBInputGetDeviceType.

## CAPTURESTATE

```
typedef enum _CAPTURESTATE
{
    RGB_STATE_CAPTURING,
    RGB_STATE_NOSIGNAL,
    RGB_STATE_INVALIDSIGNAL,
    RGB_STATE_PAUSED,
    RGB_STATE_ERROR,
} CAPTURESTATE, *PCAPTURESTATE;
```

List of the capture states returned by RGBGetCaptureState.

## PIXELGAMUT

```
typedef enum _PIXELGAMUT
{
    RGB_PIXELGAMUT_709,
    RGB_PIXELGAMUT_601,
    RGB_PIXELGAMUT_2020,
} PIXELGAMUT, *PPIXELGAMUT;
```

The colour gamut applied prior to the DMA.

RGB\_PIXELGAMUT\_709  
BT.709 colour gamut.

RGB\_PIXELGAMUT\_601  
BT.601 colour gamut.

RGB\_PIXELGAMUT\_2020  
BT.2020 colour gamut.

## PIXELRANGE

```
typedef enum _PIXELRANGE
{
    RGB_PIXELRANGE_FULL,
    RGB_PIXELRANGE_LIMITED,
} PIXELRANGE, *PPIXELRANGE;
```

The colour range expansion or compression applied prior to the DMA.

RGB\_PIXELRANGE\_FULL  
Full range.

RGB\_PIXELRANGE\_LIMITED  
Limited/Studio range.

## PIXELFORMAT

```
typedef enum _PIXELFORMAT
{
    RGB_PIXELFORMAT_AUTO,
    RGB_PIXELFORMAT_555,
    RGB_PIXELFORMAT_565,
    RGB_PIXELFORMAT_888,
    RGB_PIXELFORMAT_GREY,
    RGB_PIXELFORMAT_RGB24,
    RGB_PIXELFORMAT_YUY2,
    RGB_PIXELFORMAT_YVYU,
    RGB_PIXELFORMAT_UYVY,
    RGB_PIXELFORMAT_NV12,
    RGB_PIXELFORMAT_YV12,
    RGB_PIXELFORMAT_RGB10,
    RGB_PIXELFORMAT_Y410,
} PIXELFORMAT, *PPIXELFORMAT;
```

The PIXELFORMAT enumerated type contains a list of possible pixel formats for the Dma. Please note that a capture card may only support a subset of these pixel formats.

RGB\_PIXELFORMAT\_AUTO  
Deprecated, do not use.

RGB\_PIXELFORMAT\_555  
16bit RGB 555 format

RGB\_PIXELFORMAT\_565  
16bit RGB 565 format

RGB\_PIXELFORMAT\_888  
32bit RGB X888 format

RGB\_PIXELFORMAT\_GREY  
8bit YYYY grey/Luma format

RGB\_PIXELFORMAT\_RGB24  
24bit RGB format

RGB\_PIXELFORMAT\_YUY2  
16bit YUV 4:2:2 format

RGB\_PIXELFORMAT\_YVYU  
16bit YUV 4:2:2 format

RGB\_PIXELFORMAT\_UYVY  
16bit YUV 4:2:2 format

RGB\_PIXELFORMAT\_NV12  
12bit YUV 4:2:0 planar format

RGB\_PIXELFORMAT\_YV12

12bit YUV 4:2:0 planar format

RGB\_PIXELFORMAT\_RGB10  
32bit A2R10G10B10 format

RGB\_PIXELFORMAT\_Y410  
32bit YUV A2V10Y10U10 format

## DEINTERLACE

```
typedef enum _DEINTERLACE
{
    RGB_DEINTERLACE_WEAVE,
    RGB_DEINTERLACE_BOB,
    RGB_DEINTERLACE_FIELD_0,
    RGB_DEINTERLACE_FIELD_1,
} DEINTERLACE, *PDEINTERLACE;
```

The DEINTERLACE enumerated type contains a list of possible de-interlace options for interlaced capture sources. Please note that the capture card in conjunction with the display card may only support a subset of these de-interlace options

RGB\_DEINTERLACE\_WEAVE

Both fields are line interleaved into a single Dma frame

RGB\_DEINTERLACE\_BOB

Alternate fields that will be displayed bob de-interlaced either by the capture card prior to Dma or on a supported graphics card after Dma during drawing.

RGB\_DEINTERLACE\_FIELD\_0

Dma Field 0 only.

RGB\_DEINTERLACE\_FIELD\_1

Dma Field 1 only.

## SIGNALTYPE

```
typedef enum _SIGNALTYPE
{
    RGB_SIGNALTYPE_NOSIGNAL,
    RGB_SIGNALTYPE_VGA,
    RGB_SIGNALTYPE_DVI,
    RGB_SIGNALTYPE_YPRPB,
    RGB_SIGNALTYPE_COMPOSITE,
    RGB_SIGNALTYPE_SVIDEO,
    RGB_SIGNALTYPE_OFRANGE,
    RGB_SIGNALTYPE_SDI,
    RGB_SIGNALTYPE_DLDVI,
    RGB_SIGNALTYPE_DISPLAYPORT,
} SIGNALTYPE, *PSIGNALTYPE;
```

The **SIGNALTYPE** enumerated type contains a list of possible signal types returned by **RGBGetInputSignalType**.

**RGB\_SIGNALTYPE\_NOSIGNAL**  
No Signal has been detected

**RGB\_SIGNALTYPE\_VGA**  
RGBHV or RGBS VGA signal type

**RGB\_SIGNALTYPE\_DVI**  
DVI signal type

**RGB\_SIGNALTYPE\_YPRPB**  
SOG / SOY VGA signal type

**RGB\_SIGNALTYPE\_COMPOSITE**  
Composite signal type

**RGB\_SIGNALTYPE\_SVIDEO**  
S-Video signal type

**RGB\_SIGNALTYPE\_OFRANGE**  
The detected signal is out of range of the capture card

**RGB\_SIGNALTYPE\_SDI**  
SDI signal type

**RGB\_SIGNALTYPE\_DLDVI**  
Dual Link DVI signal type

**RGB\_SIGNALTYPE\_DISPLAYPORT**  
Display Port Signal Type

## BUFFERTYPE

```
typedef enum _BUFFERTYPE
{
    RGB_BUFFERTYPE_MAPPED,
    RGB_BUFFERTYPE_DIRECTGMA,
    RGB_BUFFERTYPE_GPUDIRECT ,
} BUFFERTYPE, *PBUFFERTYPE;
```

Describes the type of buffer being passed into RGBChainBufferEx function.

RGB\_BUFFERTYPE\_MAPPED

System memory buffer

RGB\_BUFFERTYPE\_DIRECTGMA

An AMD DirectGMA buffer.

RGB\_BUFFERTYPE\_GPUDIRECT

A nVidia GPUDirect buffer

## RGB\_FRAME\_FLAGS

```
/* Frame or Field (default is frame) */  
#define RGB_FRAME_FLAG_FIELD_0      0x0000L  
#define RGB_FRAME_FLAG_FIELD_1      0x0001L
```

## ANALOG\_INPUT\_TYPE

```
typedef enum _ANALOG_INPUT_TYPE
{
    RGB_TYPE_VGA,
    RGB_TYPE_VIDEO,
} ANALOG_INPUT_TYPE, *PANALOG_INPUT_TYPE;
```

Defines the type of analogue signal.

RGB\_TYPE\_VGA

VGA type analogue signal.

RGB\_TYPE\_VIDEO

Composite or S-Video type analogue signal.

## DIGITAL\_INPUT\_TYPE

```
typedef enum _DIGITAL_INPUT_TYPE
{
    RGB_TYPE_DVI,
    RGB_TYPE_SDI,
    RGB_TYPE_DISPLAYPORT,
} DIGITAL_INPUT_TYPE, *PDIGITAL_INPUT_TYPE;
```

Defines the type of digital signal.

RGB\_TYPE\_DVI

Single or Dual link DVI type digital signal.

RGB\_TYPE\_SDI

SDI type digital signal.

RGB\_TYPE\_DISPLAYPORT

DisplayPort type digital signal.

## VGA\_INPUT\_FLAGS

```
typedef enum _VGA_INPUT_FLAGS
{
    VGA_TYPE_5WIRE,
    VGA_TYPE_4WIRE,
    VGA_TYPE_SOG_SOY,
} VGA_INPUT_FLAGS, *PVGA_INPUT_FLAGS;
```

Defines the type of VGA signal.

VGA\_TYPE\_5WIRE

RGBHV type VGA signal.

VGA\_TYPE\_4WIRE

RGBS type VGA signal.

VGA\_TYPE\_SOG\_SOY

Sync on Green / Sync on Luma type VGA signal.

## SDI\_INPUT\_FLAGS

```
typedef enum _SDI_INPUT_FLAGS
{
    SDI_TYPE_SD,
    SDI_TYPE_HD,
    SDI_TYPE_HD_DL,
    SDI_TYPE_3GA,
    SDI_TYPE_3GB_DL,
    SDI_TYPE_3GB_DS,
    SDI_TYPE_3GB_STEREO,
} SDI_INPUT_FLAGS, *PSDI_INPUT_FLAGS;
```

Defines the type of SDI signal.

SDI\_TYPE\_SD

Standard Definition SDI signal.

SDI\_TYPE\_HD

High Definition SDI signal.

SDI\_TYPE\_HD\_DL

Dual Link High Definition SDI signal.

SDI\_TYPE\_3GA

3G Level A SDI signal.

SDI\_TYPE\_3GB\_DL

3G Level B Dual Link SDI signal.

SDI\_TYPE\_3GB\_DS

3G Level B Dual Stream SDI signal.

SDI\_TYPE\_3GB\_STEREO

3G Level B 3D SDI signal.

## DISPLAYPORT\_INPUT\_FLAGS

```
typedef enum _DISPLAYPORT_INPUT_FLAGS
{
    DISPLAYPORT_TYPE_STEREO_FS = 1,
} DISPLAYPORT_INPUT_FLAGS, *PDISPLAYPORT_INPUT_FLAGS;
```

Defines the type of DisplayPort signal.

DISPLAYPORT\_TYPE\_STEREO\_FS  
3D Frame Sequential stereo DisplayPort Signal.

## RGBOSD\_TYPE

```
typedef enum
{
    RGBOSD_TYPE_DISABLED,
    RGBOSD_TYPE_TEXT,
    RGBOSD_TYPE_OWNERDRAWN,
    RGBOSD_TYPE_BITMAP,
} RGBOSD_TYPE;
```

Defines the type of OSD. attached

RGBOSD\_TYPE\_DISABLED  
The attached OSD is disabled.

RGBOSD\_TYPE\_TEXT  
The attached OSD is text.

RGBOSD\_TYPE\_OWNERDRAWN  
The attached OSD is owner drawn

RGBOSD\_TYPE\_BITMAP  
The attached OSD is a bitmap

## ROTATIONANGLE

```
typedef enum _ROTATIONANGLE
{
    RGB_ROTATIONANGLE_0,
    RGB_ROTATIONANGLE_90,
    RGB_ROTATIONANGLE_180,
    RGB_ROTATIONANGLE_270,
} ROTATIONANGLE, *PROTATIONANGLE;
```

The ROTATIONANGLE enumerated type contains a list of rotation angles. Values in this enumeration must correspond exactly to those in D3D\_ROTATION in d3dapi.

RGB\_ROTATIONANGLE\_0

No Rotation

RGB\_ROTATIONANGLE\_90

90 degrees clockwise

RGB\_ROTATIONANGLE\_180

180 degrees clockwise

RGB\_ROTATIONANGLE\_270

270 degrees clockwise

## RGBOSD flags

```
#define RGBOSD_HOR_LEFT      0x0000
#define RGBOSD_HOR_CENTRE   0x0001
#define RGBOSD_HOR_RIGHT    0x0002

#define RGBOSD_VER_TOP      0x0000
#define RGBOSD_VER_CENTRE  0x0001
#define RGBOSD_VER_BOTTOM  0x0002
```

Defines the horizontal and vertical alignment of the on screen display

## GRAPHICSHARDWARE

```
typedef enum _GRAPHICSHARDWARE
{
    GPU_AMD,
    GPU_NVIDIA,
} GRAPHICSHARDWARE, *PGRAPHICSHARDWARE;
```

Specifies the type of graphics card technology to use for accelerated capture. GPU\_AMD specifies an AMD graphics card that supports DirectGMA and GPU\_NVIDIA specifies an NVIDIA graphics card that supports GPUDirect.

## NVIDIAOP

```
typedef enum _NVIDIAOP
{
    NVIDIA_GPU_COPY,
    NVIDIA_GPU_WAIT,
    NVIDIA_GPU_END,
} NVIDIAOP, *PNVIDIAOP;
```

Specifies an NVIDIA specific GPUDirect operation.

NVIDIA\_GPU\_COPY

Starts a DMA operation from the system memory to the GPU.

NVIDIA\_GPU\_WAIT

Waits until the DMA operation has finished and blocks any DMA operation.

NVIDIA\_GPU\_END

Ends a DMA operation from the system memory to the GPU.

## SIGNALDETECT

```
typedef enum _SIGNALDETECT
{
    RGB_SIGNALDETECT_AUTOMATIC,
    RGB_SIGNALDETECT_ANALOG,
    RGB_SIGNALDETECT_DVI,
} SIGNALDETECT, *PSIGNALDETECT;
```

Inputs that are capable of capturing both analogue and DVI signals the default strategy is to check for the presence of an analogue signal before the digital signal. This order can be changed using the RGBSetSignalDetection function.

RGB\_SIGNALDETECT\_AUTOMATIC  
Use the default signal detection strategy

RGB\_SIGNALDETECT\_ANALOG  
Check for an analogue signal only

RGB\_SIGNALDETECT\_DVI  
Check for DVI signal only.

## COLOURDOMAINDETECT

```
typedef enum _COLOURDOMAINDETECT
{
    RGB_COLOURDOMAINDETECT_RGB709_FULL,
    RGB_COLOURDOMAINDETECT_YUV709_FULL,
    RGB_COLOURDOMAINDETECT_YUV601_FULL,
    RGB_COLOURDOMAINDETECT_YUV709_STUDIO,
    RGB_COLOURDOMAINDETECT_YUV601_STUDIO,
    RGB_COLOURDOMAINDETECT_AUTO,
    RGB_COLOURDOMAINDETECT_RGB709_STUDIO,
    RGB_COLOURDOMAINDETECT_YUV2020_FULL,
    RGB_COLOURDOMAINDETECT_YUV2020_STUDIO,
    RGB_COLOURDOMAINDETECT_RGB601_FULL,
    RGB_COLOURDOMAINDETECT_RGB601_STUDIO,
    RGB_COLOURDOMAINDETECT_RGB2020_FULL,
    RGB_COLOURDOMAINDETECT_RGB2020_STUDIO,

    // Backwards compatibility
    RGB_COLOURDOMAINDETECT_RGB = RGB_COLOURDOMAINDETECT_RGB709_FULL,
    RGB_COLOURDOMAINDETECT_HDYPrPb = RGB_COLOURDOMAINDETECT_YUV709_FULL,
    RGB_COLOURDOMAINDETECT_YPrPb = RGB_COLOURDOMAINDETECT_YUV601_FULL,
    RGB_COLOURDOMAINDETECT_HDYCrCb = RGB_COLOURDOMAINDETECT_YUV709_STUDIO,
    RGB_COLOURDOMAINDETECT_YCrCb = RGB_COLOURDOMAINDETECT_YUV601_STUDIO,
} COLOURDOMAINDETECT, *PCOLOURDOMAINDETECT;
```

Describes the colour gamut and range of the input signal.

RGB\_COLOURDOMAINDETECT\_RGB709\_FULL  
RGB BT.709 Full Range

RGB\_COLOURDOMAINDETECT\_YUV709\_FULL  
YUV BT.709 Full Range

RGB\_COLOURDOMAINDETECT\_YUV601\_FULL  
YUV BT.601 Full Range

RGB\_COLOURDOMAINDETECT\_YUV709\_STUDIO  
YUV BT.709 Studio Range

RGB\_COLOURDOMAINDETECT\_YUV601\_STUDIO  
YUV BT.601 Studio Range

RGB\_COLOURDOMAINDETECT\_AUTO  
Special value indicating that the default automatic colour domain has been applied.  
Use RGBGetColourDomainDetect to get the real on-the-wire colour domain

RGB\_COLOURDOMAINDETECT\_RGB709\_STUDIO  
RGB BT.709 Studio Range

RGB\_COLOURDOMAINDETECT\_YUV2020\_FULL  
YUV BT.2020 Full Range

RGB\_COLOURDOMAINDETECT\_YUV2020\_STUDIO

## YUV BT.2020 Studio Range

RGB\_COLOURDOMAINDETECT\_RGB601\_FULL  
RGB BT.601 Full Range

RGB\_COLOURDOMAINDETECT\_RGB601\_STUDIO  
RGB BT.601 Studio Range

RGB\_COLOURDOMAINDETECT\_RGB2020\_FULL  
RGB BT.2020 Full Range

RGB\_COLOURDOMAINDETECT\_RGB2020\_STUDIO  
RGB BT.2020 Studio Range

For backwards compatibility the original COLOURDOMAINDETECT values have been mapped to the new values:

RGB\_COLOURDOMAINDETECT\_RGB  
RGB BT.709 Full Range

RGB\_COLOURDOMAINDETECT\_HDYPrPb  
YUV BT.709 Full Range

RGB\_COLOURDOMAINDETECT\_YPrPb  
YUV BT.601 Full

RGB\_COLOURDOMAINDETECT\_HDYCrCb  
YUV BT.709 Studio Range

RGB\_COLOURDOMAINDETECT\_YCrCb  
YUV BT.601 Studio Range

## LIVESTREAM

```
typedef enum _LIVESTREAM{  
    LIVESTREAM_OFF,  
    LIVESTREAM_1,  
} LIVESTREAM, *PLIVESTREAM;
```

The Live stream status of an input.

LIVESTREAM\_OFF

Live stream is not enabled

LIVESTREAM\_1

Live stream is enabled for the first client of each Dma engine.

## LIVESTREAM\_SUPPORT

```
typedef enum _LIVESTREAM_SUPPORT
{
    RGB_LIVESTREAM_UNSUPPORTED,
    RGB_LIVESTREAM_SELECTABLE,
    RGB_LIVESTREAM_ALWAYS_ON,
} LIVESTREAM_SUPPORT, *PLIVESTREAM_SUPPORT;
```

The type of LiveStream support for an input.

RGB\_LIVESTREAM\_UNSUPPORTED

LiveStream is not supported

RGB\_LIVESTREAM\_SELECTABLE

LiveStream can be enabled or disabled.

RGB\_LIVESTREAM\_ALWAYS\_ON

LiveStream is always on and is not configurable.

## LINKRATE

```
typedef enum _LINKRATE {  
    RGB_LINKRATE_RBR,  
    RGB_LINKRATE_HBR,  
    RGB_LINKRATE_HBR2,  
} LINKRATE, *PLINKRATE;
```

The link rate value of the DisplayPort link.

RGB\_LINKRATE\_RBR

1.62 GBits/s Reduced bit rate.

RGB\_LINKRATE\_HBR

2.7 GBits/s High Bit Rate.

RGB\_LINKRATE\_HBR2

5.4 GBits/s High Bit Rate 2

## RGBGANG\_TYPE

```
typedef enum _RGBGANG_TYPE
{
    RGBGANG_TYPE_DISABLED,
    RGBGANG_TYPE_2x1,
    RGBGANG_TYPE_1x2,
    RGBGANG_TYPE_3x1,
    RGBGANG_TYPE_1x3,
    RGBGANG_TYPE_4x1,
    RGBGANG_TYPE_1x4,
    RGBGANG_TYPE_2x2,
} RGBGANG_TYPE, *PRBGANG_TYPE;
```

Describes how the gen-locked inputs on a supported capture card are ganged together to form a single capture on the first input of a supported card.

RGBGANG\_TYPE\_DISABLED

Input Ganging is disabled

RGBGANG\_TYPE\_2x1

Inputs one and two are ganged together horizontally to form a single input that is twice the horizontal resolution.

RGBGANG\_TYPE\_1x2

Inputs one and two are ganged together horizontally to form a single input that is twice the vertical resolution.

RGBGANG\_TYPE\_3x1

Inputs one, two and three are ganged together horizontally to form a single input that is three times the horizontal resolution.

RGBGANG\_TYPE\_1x3

Inputs one, two and three are ganged together horizontally to form a single input that is three times the vertical resolution.

RGBGANG\_TYPE\_4x1

Inputs one, two, three and four are ganged together horizontally to form a single input that is four times the horizontal resolution.

RGBGANG\_TYPE\_1x4

Inputs one, two, three and four are ganged together horizontally to form a single input that is four times the vertical resolution.

RGBGANG\_TYPE\_2x2

Inputs one, two, three and four are ganged together horizontally to form a single input that is twice the horizontal resolution and twice the vertical resolution.

## DGCMEDIASAMPLETYPE

```
typedef enum tagDGCMEDIASAMPLETYPE
{
    DGCMEDIASAMPLETYPE_UNKNOWN
    DGCMEDIASAMPLETYPE_RAW_VIDEO
    DGCMEDIASAMPLETYPE_ENC_VIDEO
} DGCMEDIASAMPLETYPE, *PDGCMEDIASAMPLETYPE;
```

Describes the Major Type of the Media Sample.

DGCMEDIASAMPLETYPE\_RAW\_VIDEO

Raw uncompressed video

DGCMEDIASAMPLETYPE\_ENC\_VIDEO

Compressed video.

## DGCMEDIASAMPLESUBTYPE

```
typedef enum tagDGCMEDIASAMPLESUBTYPE
{
    DGCMEDIASAMPLESUBTYPE_UNKNOWN = 0,          // UNKNOWN

    // Uncompressed Video Types
    DGCRAWVIDEOSAMPLESUBTYPE_UNKNOWN,
    DGCRAWVIDEOSAMPLESUBTYPE_RGB555,
    DGCRAWVIDEOSAMPLESUBTYPE_RGB565,
    DGCRAWVIDEOSAMPLESUBTYPE_RGB888,
    DGCRAWVIDEOSAMPLESUBTYPE_YYYY,
    DGCRAWVIDEOSAMPLESUBTYPE_RGB24,
    DGCRAWVIDEOSAMPLESUBTYPE_YUY2,
    DGCRAWVIDEOSAMPLESUBTYPE_YVYU,
    DGCRAWVIDEOSAMPLESUBTYPE_UYVY,
    DGCRAWVIDEOSAMPLESUBTYPE_NV12,
    DGCRAWVIDEOSAMPLESUBTYPE_YV12,
    DGCRAWVIDEOSAMPLESUBTYPE_I420,
    DGCRAWVIDEOSAMPLESUBTYPE_Y42B,
    DGCRAWVIDEOSAMPLESUBTYPE_Y444,
    DGCRAWVIDEOSAMPLESUBTYPE_RGB10,
    DGCRAWVIDEOSAMPLESUBTYPE_Y410,

    DGCENCVIDEOSAMPLESUBTYPE_H264,
} DGCMEDIASAMPLESUBTYPE, *PDGCMEDIASAMPLESUBTYPE;
```

Describes the SubType of the Media Sample.

For DGCMEDIASAMPLETYPE\_RAW\_VIDEO types the following subtypes are defined.

DGCRAWVIDEOSAMPLESUBTYPE\_RGB555  
16bit RGB 555 pixel format.

DGCRAWVIDEOSAMPLESUBTYPE\_RGB565  
16bit RGB 565 pixel format.

DGCRAWVIDEOSAMPLESUBTYPE\_RGB888  
32bit RGB pixel format.

DGCRAWVIDEOSAMPLESUBTYPE\_YYYY  
8bit Luma pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_RGB24  
24bit RGB pixel format.

DGCRAWVIDEOSAMPLESUBTYPE\_YUY2  
16bit YUV 4:2:2 packed pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_YVYU  
16bit YUV 4:2:2 packed pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_UYVY  
16bit YUV 4:2:2 packed pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_NV12  
12bit YUV 4:2:0 planar format

DGCRAWVIDEOSAMPLESUBTYPE\_YV12  
12bit YUV 4:2:0 planar format

DGCRAWVIDEOSAMPLESUBTYPE\_I420  
12bit YUV 4:2:0 planar format

DGCRAWVIDEOSAMPLESUBTYPE\_Y42B  
16bit YUV 4:2:2 packed pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_Y444  
24bit YUV 4:4:4 packed pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_RGB10,  
32bit RGB A2R10G10B10 pixel format

DGCRAWVIDEOSAMPLESUBTYPE\_Y410,  
32bit YUV A2V10Y10U10 packed pixel format

For DGCEDIASAMPLETYPE\_ENC\_VIDEO types the following subtypes are defined.

DGCENCVIDEOSAMPLESUBTYPE\_H264  
H264 Encoded video.

## RGBEasy Structures

## RGBFRAMEDATA

```
typedef struct tagRGBFrameData
{
    unsigned long          Size;
    LPBITMAPINFOHEADER    PBitmapInfo;
    void                  *PBitmapBits;
    unsigned long          FrameFlags;
    ULONGLONG             TimeStamp;
} RGBFRAMEDATA, *PRGBFRAMEDATA;
```

The RGBFRAMEDATA structure describes the the buffer received by a PRGBFRAMECAPTUREDFNEX frame captured call-back function.

Size

The size of the structure in bytes.

PBitmapInfo

Pointer to a BITMAPINFOHEADER structure describing the buffer

PBitmapBits

Pointer to the bitmap bits

FrameFlags

RGB\_FRAME\_XXX flags

TimeStamp

Captured frame timestamp in 100ns units

## RGBMODECHANGEDINFO

```
typedef struct
{
    unsigned long    Size;
    unsigned long    RefreshRate;
    unsigned long    LineRate;
    unsigned long    TotalNumberOfLines;
    long             BInterlaced;
    long             BDVI;
    union
    {
        ANALOG_INPUT_TYPE    AnalogType;
        DIGITAL_INPUT_TYPE    DigitalType;
    };
    unsigned long    VideoStandard;
    union
    {
        VGA_INPUT_FLAGS    VGAFlags;
        SDI_INPUT_FLAGS    SDIFlags;
        DISPLAYPORT_INPUT_FLAGS    DisplayPortFlags
    };
} RGBMODECHANGEDINFO, *PRGBMODECHANGEDINFO;
```

The RGBMODECHANGEDINFO structure describes the newly detected video mode.

### Size

The size of this structure.

### RefreshRate

The horizontal refresh rate in Hertz.

### LineRate

The vertical line rate in Hertz.

### TotalNumberOfLines

The total number of lines.

### BInterlaced

A boolean indicating an interlaced mode has been detected.

### BDVI

A boolean indicating a digital mode has been detected.

### AnalogType

A variable indicating the type of analogue input detected. Only valid if BDVI is 0.

### DigitalType

A variable indicating the type of digital mode. Only valid if BDVI is 1.

### VideoStandard

A variable indicating the video standard of the detected mode. Only valid if BDVI is 0 and AnalogType is RGB\_TYPE\_VIDEO.

VGAFlags

A variable indicating the type of VGA signal detected. Only valid if BDVI is 0 and AnalogType is RGB\_TYPE\_VGA.

SDIFlags

A variable indicating the type of SDI signal detected. Only valid if BDVI is 1 and DigitalType is RGB\_TYPE\_SDI.

DisplayPortFlags

A variable indicating the type of DisplayPort signal detected. Only valid if BDVI is 1 and DigitalType is RGB\_TYPE\_DISPLAYPORT.

## RGBMODEINFO

```
typedef struct
{
    unsigned long    Size;
    CAPTURESTATE    State;
    unsigned long    RefreshRate;
    unsigned long    LineRate;
    unsigned long    TotalNumberOfLines;
    long             BInterlaced;
    long             BDVI;
    union
    {
        {
            ANALOG_INPUT_TYPE    AnalogType;
            DIGITAL_INPUT_TYPE    DigitalType;
        };
        unsigned long    VideoStandard;
        union
        {
            {
                VGA_INPUT_FLAGS    VGAFlags;
                SDI_INPUT_FLAGS    SDIFlags;
                DISPLAYPORT_INPUT_FLAGS    DisplayPortFlags
            };
        };
    };
} RGBMODECHANGEDINFO, *PRGBMODECHANGEDINFO;
```

The RGBMODEINFO describes the detected video mode. If State is RGB\_STATE\_NOSIGNAL then none of the other values contain valid information. If State is RGB\_STATE\_INVALIDSIGNAL then RefreshRate and LineRate contain the out of range and BDVI indicates whether the signal is analogue or digital. None of the other values are valid.

### Size

The size of this structure.

### State

The state of the capture.

### RefreshRate

The horizontal refresh rate in Hertz.

### LineRate

The vertical line rate in Hertz.

### TotalNumberOfLines

The total number of lines.

### BInterlaced

A boolean indicating an interlaced mode has been detected.

### BDVI

A boolean indicating a digital mode has been detected.

### AnalogType

A variable indicating the type of analogue input detected. Only valid if BDVI is 0.

`DigitalType`

A variable indicating the type of digital mode. Only valid if BDVI is one.

`VideoStandard`

A variable indicating the video standard of the detected mode.

Only valid if BDVI is 0 and AnalogType is RGB\_TYPE\_VIDEO.

`VGAFlags`

A variable indicating the type of VGA signal detected. Only valid if BDVI is 0 and AnalogType is RGB\_TYPE\_VGA.

`SDIFlags`

A variable indicating the type of SDI signal detected. Only valid if BDVI is 1 and DigitalType is RGB\_TYPE\_SDI.

`DisplayPortFlags`

A variable indicating the type of DisplayPort signal detected. Only valid if BDVI is 1 and DigitalType is RGB\_TYPE\_DISPLAYPORT.

## RGBVALUECHANGEDINFO

```
typedef struct
{
    unsigned long    Size;
    SIGNEDVALUE     HorPosition;
    UNSIGNEDVALUE   HorScale;
    UNSIGNEDVALUE   VerPosition;
    SIGNEDVALUE     CaptureWidth;
    UNSIGNEDVALUE   CaptureHeight;
    SIGNEDVALUE     Brightness;
    SIGNEDVALUE     Contrast;
    SIGNEDVALUE     BlackLevel;
    SIGNEDVALUE     Phase;
    UNSIGNEDVALUE   RedGain;
    UNSIGNEDVALUE   GreenGain;
    UNSIGNEDVALUE   BlueGain;
    UNSIGNEDVALUE   RedOffset;
    UNSIGNEDVALUE   GreenOffset;
    UNSIGNEDVALUE   BlueOffset;
    UNSIGNEDVALUE   Saturation;
    UNSIGNEDVALUE   Hue;
    UNSIGNEDVALUE   VideoStandard;
    UNSIGNEDVALUE   Equalisation;
    UNSIGNEDVALUE   ColourDomain;
    UNSIGNEDVALUE   LinkRate;
} RGBVALUECHANGEDINFO, *PRGBVALUECHANGEDINFO;
```

The RGBVALUECHANGEDINFO structure describes changes in the capture parameters of the current video mode instigated by another capture on the same RGB input. Each capture parameter is represented by a structure that contains a flag that indicates if this parameter has been changed and what the new value is.

### Size

The size of this structure in bytes.

### HorPosition

Horizontal Position.

### HorScale

Horizontal Scale.

### VerPosition

Vertical Position.

### CaptureWidth

Capture Width.

### CaptureHeight

Capture Height.

### Brightness

Brightness.

Contrast

**Contrast.**

BlackLevel

**Black Level.**

Phase

**Phase.**

GreenGain

**Colour Balance Green Gain.**

BlueGain

**Colour Balance Blue Gain.**

RedOffset

**Colour Balance Red Offset.**

GreenOffset

**Colour Balance Green Offset.**

BlueOffset

**Colour Balance Blue Offset.**

Saturation

**Saturation.**

Hue

**Hue.**

VideoStandard

**Video Standard.**

Equalisation

**DVI equalisation.**

ColourDomain

**Colour domain.**

LinkRate

**DisplayPort Link Rate.**

## SIGNEDVALUE

```
typedef struct  
{  
    long          BChanged;  
    signed long Value;  
} SIGNEDVALUE;
```

BChanged

Flag to indicate that the value has changed.

Value

The new signed value.

## UNSIGNEDVALUE

```
typedef struct
{
    long          BChanged;
    unsigned long Value;
} UNSIGNEDVALUE;
```

BChanged

Flag to indicate that the value has changed.

Value

The new unsigned value.

## RGBDRIVERVER

```
typedef struct tagDriverVer
{
    unsigned long Major;
    unsigned long Minor;
    unsigned long Micro;
    unsigned long Revision;
} RGBDRIVERVER, *PRGBDRIVERVER;
```

This structure provides the version of the driver which has been installed against the input. The version is defined as [Major].[Minor].[Micro].[Revision]

## RGBLOCATION

```
typedef struct tagLocation
{
    unsigned long Bus;
    unsigned long Device;
    unsigned long Function;
} RGBLOCATION, *PRGBLOCATION;
```

This structure details the PCI location of the input in terms of [Bus].[Device].[Function].

## RGBCHASSIS

```
typedef struct tagChassis
{
    unsigned long Index;
    unsigned long Slot;
} RGBCHASSIS, *PRGBCHASSIS;
```

This structure describes the physical location of the card when it is installed in compatible systems. If the driver is unable to retrieve this information then the Chassis Index and Slot will both be set to zero.

### Index

This is the index of the chassis in which the card is installed. Chassis 0 is the host, chassis 1 is the first expansion chassis etc.

### Slot

This is the physical PCIe slot in which the card has been installed.

## RGBINPUTINFO

```
typedef struct tagRGBDevInfoA
{
    unsigned long    Size;

    RGBDRIVERVER    Driver;
    RGBLOCATION       Location;
    unsigned long    FirmWare;
    unsigned long    VHDL;
    unsigned long    Identifier[2];
    CHAR             DeviceName[256];
    RGBCHASSIS       Chassis;
} RGBINPUTINFOA, *PRGBINPUTINFOA

typedef struct tagRGBDevInfoW
{
    unsigned long    Size;
    RGBDRIVERVER    Driver;
    RGBLOCATION       Location;
    unsigned long    FirmWare;
    unsigned long    VHDL;
    unsigned long    Identifier[2];
    WCHAR           DeviceName[256];
    RGBCHASSIS       Chassis;
} RGBINPUTINFOW, *PRGBINPUTINFOW
```

The RGBINPUTINFO structure describes the characteristics of the physical input including, for instance, the revision of firmware running on the input and it's device identifier.

- For ANSI builds, RGBINPUTINFO is defined as RGBINPUTINFOA.
- For UNICODE builds, RGBINPUTINFO is defined as RGBINPUTINFOW.

### Size

The size of this structure in bytes.

### Driver

The version of driver installed against the input.

### Location

The PCI Bus address of the input.

### Firmware

The firmware revision running on the input.

### VHDL

The VHDL revision programmed into the input.

### Identifier

The unique hardware identifier.

### DeviceName

The name of the device.

Chassis

The chassis and slot number that the card has been installed in.

## DGCMEDIASAMPLE

```
typedef struct tagDGCMEDIASAMPLE
{
    uint32_t                Size;
    DGCMEDIASAMPLETYPE     MajorType;
    DGCMEDIASAMPLESUBTYPE  SubType;
    void                    *PFormatHeader;
    DGCBUFFERHEADERTYPE    BufferHeaderType;
    void                    *PBufferHeader;
} DGCMEDIASAMPLE, *PDGCMEDIASAMPLE;
```

The DGCMEDIASAMPLE is used to describes both planar and non-planar buffers for use by RGBChainMediaSample and its RGBMEDIASAMPLECAPTUREDFN call-back function

**Size**

The size of this structure in bytes.

**MajorType**

Major Type of media sample.

**SubType**

Sub Type of media sample.

**PFormatHeader**

Sub Type of media sample. Currently only DGCVIDEOHEADER is supported.

**BufferHeaderType**

The buffer header type. Currently only DGCBUFFERHEADERTYPE\_MEMORY is supported

**PBufferHeader**

Pointer to the media sample buffer. Currently only DGCMEMORYBUFFERHEADER is supported.

## DGCVIDEOHEADER

```
typedef struct tagDGCVIDEOHEADER
{
    uint32_t          Size;
    uint32_t          Flags;
    int32_t           Width;
    int32_t           Height;
    int32_t           FrameRate;
} DGCVIDEOHEADER, *PDGCVIDEOHEADER;
```

Describes the overall dimensions of the video to be captured.

Size

The size of the structure in bytes.

Flags

Flags. There are no flags currently defined.

Width

Image width in pixels.

Height

Image height in lines.

FrameRate

Required frame rate in mHz. Currently not supported.

## DGCMEMORYBUFFERHEADER

```
typedef struct tagDGCMEMORYBUFFERHEADER
{
    uint32_t          Size;
    uint32_t          Flags;
    uint64_t          StartTime;
    uint64_t          EndTime;
    uint32_t          NumberOfPlanes;
    DGCMEMORYBUFFER Planes[3];
} DGCMEMORYBUFFERHEADER, *PDGCMEMORYBUFFERHEADER;
```

### Size

Size of this structure in bytes.

### Flags

Flags. .There are no flags currently defined

### StartTime

Media sample start time in 100ns units.

### EndTime

Media sample end time in 100ns units.

### NumberOfPlanes

Number of video planes defined in this structure.

### Planes

Array of video planes.

## DGCMEMORYBUFFER

```
typedef struct tagDGCMEMORYBUFFER
{
    uint32_t Size;
    uint32_t Flags;
    void      *PBuffer;
    uint32_t Length;
    uint32_t Pitch;
    uint32_t OffsetX;
    uint32_t OffsetY;
    uint32_t ActualLength;
} DGCMEMORYBUFFER, *PDGCMEMORYBUFFER;
```

Size

Size of this structure in bytes.

Flags

Flags. . There are no flags currently defined.

PBuffer

Pointer to the start of the buffer.

Length

Buffer length in bytes.

Pitch

Buffer pitch in bytes.

OffsetX

Start offset into the buffer in pixels.

OffsetY

Start offset into the buffer in lines.

ActualLength

Amount of actual data within the buffer in bytes.

## GPTRANSFERDESCRIPTOR

```
typedef struct
{
    unsigned int          Size;
    unsigned int          **Buffer;
    unsigned long         Width;
    unsigned long         Height;
    unsigned int          OglByteFormat;
    unsigned int          OglColourFormat;
    unsigned int          FormatSize;
    unsigned int          *OglObject;
    unsigned int          NumBuffers;
    GRAPHICSHARDWARE     GpuBrand;
    unsigned int          BufferSize;
} GPTRANSFERDESCRIPTOR, *PGPTRANSFERDESCRIPTOR;
```

The GPTRANSFERDESCRIPTOR structure describes the parameters for the communication between the capture card and the graphics card.

### Size

The size of this structure in bytes.

### Buffer

A pointer to an array of buffer handles allocated by the driver and used to identify the buffer in the frame captured callback function.

### Width

The width of the capture.

### Height

The height of the capture.

### OglByteFormat

OpenGL data type of the pixel data.

### OglColourFormat

OpenGL format of the pixel data.

### FormatSize

Number of bytes for the OpenGL colour format.

### OglObject

Pointer to an OpenGL array of object names (AMD) or an array of OpenGL texture names (NVIDIA).

### NumBuffers

Number of buffers to be used.

### GpuBrand

A GRAPHICSHARDWARE variable which specifies the brand of the graphics card (AMD or NVIDIA).

### BufferSize

The size of each buffer, this will be calculated by the capture driver.