



NI-IMAQ™ Function Reference Help

June 2008, 370161J-01

The *NI-IMAQ Function Reference Help* is for NI-IMAQ driver software users. NI-IMAQ provides a powerful application programming interface (API) between image acquisition applications and National Instruments image acquisition devices. The NI-IMAQ Function Library, a series of C functions for using LabWindows™/CVI™ and other C compilers with NI image acquisition devices, is included with the NI-IMAQ driver software.

For more information about this help file, refer to the following topics:

[Using Help](#)

[Related Documentation](#)

[Important Information](#)

[Technical Support and Professional Services](#)

To comment on National Instruments documentation, refer to the [National Instruments Web site](#).

© 1998–2008 National Instruments Corporation. All rights reserved.

Related Documentation

Some NI-IMAQ manuals also are available as PDFs. You must have Adobe Reader with Search and Accessibility 5.0.5 or later installed to view the PDFs. Refer to the [Adobe Systems Incorporated Web site](#) at www.adobe.com to download Adobe Reader. Refer to the [National Instruments Product Manuals Library](#) at ni.com/manuals for updated documentation resources.

The following documents contain information you may find helpful as you use this help file:

- *NI Vision Acquisition Software Release Notes*—Contains information about new functionality, minimum system requirements, and installation instructions for NI-IMAQ driver software.
- *Measurement & Automation Explorer Help for NI-IMAQ*—Describes how to configure NI-IMAQ driver software, NI image acquisition devices, and cameras using Measurement & Automation Explorer.
- NI Developer Zone—For more information about developing your image acquisition application, visit the [NI Developer Zone](#) at ni.com/zone. NI Developer Zone contains example programs, tutorials, technical presentations, the Instrument Driver Network, a measurement glossary, an online magazine, and a product advisor, as well as a community area where you can share ideas, questions, and source code with developers around the world.

Using Help

[Conventions](#)

[Navigating Help](#)

[Searching Help](#)

[Printing Help File Topics](#)

Conventions

This help file uses the following conventions:

- < > Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, AO <0..3>.
- » The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
- * The * symbol indicates that the following text applies only to a specific product, a specific operating system, or a specific software version.
-  This icon denotes a tip, which alerts you to advisory information.
-  This icon denotes a note, which alerts you to important information.
- bold** Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
- green Underlined text in this color denotes a link to a help topic, help file, or Web address.
- italic* Italic text denotes variables, emphasis, cross references, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.
- monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

Navigating Help (Windows Only)

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window or use the following toolbar buttons located above the tabs:

- **Hide**—Hides the navigation pane from view.
- **Locate**—Locates the currently displayed topic in the **Contents** tab, allowing you to view related topics.
- **Back**—Displays the previously viewed topic.
- **Forward**—Displays the topic you viewed before clicking the **Back** button.
- **Options**—Displays a list of commands and viewing options for the help file.

Searching Help (Windows Only)

Use the **Search** tab to the left of this window to locate content in this help file. If you want to search for words in a certain order, such as "related documentation," add quotation marks around the search words as shown in the example. Searching for terms on the **Search** tab allows you to quickly locate specific information and information in topics that are not included on the **Contents** tab.

Wildcards

You also can search using asterisk (*) or question mark (?) wildcards. Use the asterisk wildcard to return topics that contain a certain string. For example, a search for "prog**" lists topics that contain the words "program," "programmatically," "progress," and so on.

Use the question mark wildcard as a substitute for a single character in a search term. For example, "?ext" lists topics that contain the words "next," "text," and so on.

Nested Expressions

Use nested expressions to combine searches to further refine a search. You can use Boolean expressions and wildcards in a nested expression. For example, "example AND (program OR VI)" lists topics that contain "example program" or "example VI." You cannot nest expressions more than five levels.

Boolean Expressions

Click the  button to add Boolean expressions to a search. The following Boolean operators are available:

- **AND** (default)—Returns topics that contain both search terms. You do not need to specify this operator unless you are using nested expressions.
- **OR**—Returns topics that contain either the first or second term.
- **NOT**—Returns topics that contain the first term without the second term.
- **NEAR**—Returns topics that contain both terms within eight words of each other.

Search Options

Use the following checkboxes on the **Search** tab to customize a search:

- **Search previous results**—Narrows the results from a search that returned too many topics. You must remove the checkmark from this checkbox to search all topics.
- **Match similar words**—Broadens a search to return topics that contain words similar to the search terms. For example, a search for "program" lists topics that include the words "programs," "programming," and so on.
- **Search titles only**—Searches only in the titles of topics.

Printing Help File Topics (Windows Only)

Complete the following steps to print an entire book from the **Contents** tab:

1. Right-click the book.
2. Select **Print** from the shortcut menu to display the **Print Topics** dialog box.
3. Select the **Print the selected heading and all subtopics** option.
 **Note** Select **Print the selected topic** if you want to print the single topic you have selected in the **Contents** tab.
4. Click the **OK** button.

Printing PDF Documents

This help file may contain links to PDF documents. To print PDF documents, click the print button located on the Adobe Acrobat Viewer toolbar.

NI-IMAQ Functions

Interface Functions

[imgInterfaceOpen](#)

[imgSessionOpen](#)

[imgClose](#)

[imgInterfaceQueryNames](#)

[imgInterfaceReset](#)

High-Level Functions

Snap Functions

[imgSnap](#)

[imgSnapArea](#)

Grab Functions

[imgGrabArea](#)

[imgGrab](#)

[imgGrabSetup](#)

Ring and Sequence Functions

[imgRingSetup](#)

[imgSequenceSetup](#)

[imgSessionStartAcquisition](#)

[imgSessionStopAcquisition](#)

Signal I/O Functions

[imgSessionTriggerConfigure2](#)

[imgSessionLineTrigSource2](#)

[imgSessionTriggerClear](#)

[imgSessionTriggerDrive2](#)

[imgSessionTriggerRoute2](#)

[imgSessionTriggerRead2](#)

[imgSessionWaitSignal2](#)

[imgSessionWaitSignalAsync2](#)

[imgEncoderResetPosition](#)

[imgPulseCreate2](#)

[imgPulseDispose](#)

[imgPulseRate](#)

[imgPulseStart](#)

[imgPulseStop](#)

[Miscellaneous Functions](#)

[imgSessionStatus](#)

[imgSessionGetROI](#)

[imgSessionConfigureROI](#)

[imgSessionFitROI](#)

[imgSessionGetBufferSize](#)

[Low-Level Functions](#)

[Acquisition Functions](#)

[imgSessionAbort](#)

[imgSessionAcquire](#)

[imgSessionConfigure](#)

[imgSessionCopyArea](#)

[imgSessionCopyBuffer](#)

[imgSessionExamineBuffer](#)

[imgSessionReleaseBuffer](#)

[Attribute Functions](#)

[imgGetAttribute](#)

[imgSetAttribute](#)

[imgGetCameraAttributeNumeric](#)

[imgGetCameraAttributeString](#)

[imgSetCameraAttributeNumeric](#)

[imgSetCameraAttributeString](#)

[imgSessionSetUserLUT8bit](#)

[imgSessionSetUserLUT16bit](#)

Buffer Management Functions

[imgCreateBuffer](#)

[imgCreateBufList](#)

[imgDisposeBuffer](#)

[imgDisposeBufList](#)

[imgGetBufferElement](#)

[imgSetBufferElement](#)

Utility Functions

[imgPlot](#)

[imgPlotDC](#)

[imgSessionSaveBufferEx](#)

[imgShowError](#)

[imgBayerColorDecode](#)

[imgCalculateBayerColorLUT](#)

Serial Communication Functions

[imgSessionSerialWrite](#)

[imgSessionSerialRead](#)

[imgSessionSerialReadBytes](#)

[imgSessionSerialFlush](#)

Interface Functions

Use interface functions to set up and close the interface and session and to perform operations specific to an interface. All interface functions require a valid INTERFACE_ID.

[imgInterfaceOpen](#)

[imgSessionOpen](#)

[imgClose](#)

[imgInterfaceQueryNames](#)

[imgInterfaceReset](#)

imgInterfaceOpen

Usage

```
rval imgInterfaceOpen(const char* interfaceName, INTERFACE_ID* pifid);
```

Purpose

Opens an interface by name as specified in Measurement & Automation Explorer (MAX). If it is successful, this function returns an INTERFACE_ID.

Parameters

Name	Type	Direction
interfaceName	const char*	input
pifid	INTERFACE_ID*	output
rval	Int32	output

Parameter Discussion

interfaceName: name of the interface to open, such as img0, img1, and so on.



Note The interface name always identifies a single port of an image acquisition device. A port identifies a single independent data stream from a camera. All NI image acquisition devices support at least one port. Devices that support multiple ports can sustain independent and asynchronous acquisitions from the cameras on each port.

The port number may be explicitly identified by using the :: operator to append the port number suffix to the interface name. Port numbers are zero-based. For example, img0::1 opens port number 1 of the image acquisition device identified by img0. Interface names that do not have a port number suffix default to port 0. img0::0 and img0 are equivalent in meaning.

pifid: pointer to an INTERFACE_ID variable. If the function succeeds, the variable is populated with a valid INTERFACE_ID that you can use in subsequent functions.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note You can use [imgInterfaceQueryNames](#) to retrieve a valid list of interface names.

imgSessionOpen

Usage

```
rval imgSessionOpen(INTERFACE_ID ifid, SESSION_ID* psid);
```

Purpose

Opens a session and returns a session ID. This function inherits all data associated with the given interface.

Parameters

Name	Type	Direction
ifid	INTERFACE_ID	input
psid	SESSION_ID*	output
rval	Int32	output

Parameter Discussion

ifid: valid INTERFACE_ID.

psid: pointer to a SESSION_ID variable. If the function succeeds, the variable is populated with a valid SESSION_ID that you can use in subsequent functions.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgClose

Usage

```
rval imgClose(uInt32 void_id, uInt32 freeResources);
```

Purpose

Closes a session or interface and releases all associated resources.

Parameters

Name	Type	Direction
void_id	uInt32	input
freeResources	uInt32	input
rval	Int32	output

Parameter Discussion

void_id: valid SESSION_ID or INTERFACE_ID

freeResources: if **freeResources** is non-zero, the function releases all buffers and buffer lists associated with the session or interface. If **freeResources** is 0, the function performs no buffer cleanup.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgInterfaceQueryNames

Usage

```
rval imgInterfaceQueryNames(UInt32 index, char* name);
```

Purpose

Returns the interface name identified by the **index** parameter. To obtain a list of all the available interface names, call this function repeatedly until the function returns an error. Make the first call with **index** initialized to zero. Each successive call increments the value of **index** by one.



Note This function will not enumerate the individual port numbers for interfaces that support multiple ports. For example, an interface img0 that supports two ports will still be enumerated as img0. To query the number of ports that an interface supports, open port 0 of the interface and query the attribute `IMG_ATTR_NUM_PORTS`.

Parameters

Name	Type	Direction
------	------	-----------

index	UInt32	input
--------------	--------	-------

name	char*	output
-------------	-------	--------

rval	Int32	output
-------------	-------	--------

Parameter Discussion

index: interface number to obtain.

name: pointer to a character array to receive the interface name. The array must contain at least 256 elements.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgInterfaceReset

Usage

```
rval imgInterfaceReset(INTERFACE_ID ifid);
```

Purpose

Performs a reset on the interface. This function sets the hardware associated with the interface to its initial state.

Parameters

Name	Type	Direction
ifid	INTERFACE_ID	input
rval	Int32	input

Parameter Discussion

ifid: valid INTERFACE_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

High-Level Functions

Use high-level functions to easily perform basic image acquisition functions without having an advanced understanding of the image acquisition process. For example, high-level functions include acquiring images in single (snap) or continuous (grab) mode.

[Snap Functions](#)

[Grab Functions](#)

[Ring and Sequence Functions](#)

[Signal I/O Functions](#)

[Miscellaneous Functions](#)

Snap Functions

Use the snap functions to acquire a single image after using [imgInterfaceOpen](#) and [imgSessionOpen](#) to obtain a valid SESSION_ID.

[imgSnap](#)

[imgSnapArea](#)

imgSnap

Usage

```
rval imgSnap(SESSION_ID sid, void** bufAddr);
```

Purpose

Acquires a single image.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufAddr	void**	input/output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufAddr: pointer to a pointer to an area of memory in which to store the image. If **bufAddr** points to a NULL pointer, this call allocates a buffer, acquires an image, and returns the buffer address in the location specified by **bufAddr**. If **bufAddr** points to a non-NULL pointer, this function acquires into that buffer.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSnapArea

Usage

```
rval imgSnapArea(SESSION_ID sid, void** bufAddr, uInt32 top, uInt32 left,  
uInt32 height, uInt32 width, uInt32 rowPixels);
```

Purpose

Acquires a single image. This function operates the same as [imgSnap](#), but is used to acquire only a portion of the image.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufAddr	void**	input/output
top	ulInt32	input
left	ulInt32	input
height	ulInt32	input
width	ulInt32	input
rowPixels	ulInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufAddr: pointer to a pointer to an area of memory in which to store the image. If the parameter points to a NULL pointer, this call allocates a buffer, acquires an image, and returns the buffer address in the location specified by **bufAddr**. If the parameter points to a non-NULL pointer, this function acquires into that buffer.

top: top offset of the first pixel to acquire.

left: left offset of the first pixel to acquire.

height: height of the area to acquire.

width: width of the area to acquire.

rowPixels: total number of pixels in each image line. Passing a zero for this parameter causes the function to ignore the parameter and use the IMG_ATTR_ROWPIXELS attribute instead. Use this parameter for byte alignment or if the image buffer contains a border for image processing.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information, call [imgShowError](#).

Grab Functions

Use the grab functions to perform a continuous acquisition into a single buffer.

To use the grab functions, call `imgGrabSetup` to configure the session for grabbing and optionally start the acquisition process. If you do not start the acquisition using `imgGrabSetup`, you must start it by calling [`imgSessionStartAcquisition`](#) prior to calling the `imgGrab` and `imgGrabArea` functions.

After the acquisition has started, obtain an image by calling `imgGrab` and `imgGrabArea`. To stop the acquisition, call [`imgSessionStopAcquisition`](#).

[imgGrabSetup](#)

[imgGrab](#)

[imgGrabArea](#)

imgGrabSetup

Usage

```
rval imgGrabSetup(SESSION_ID sid, uInt32 startNow);
```

Purpose

Configures and optionally starts a continuous acquisition.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
startNow	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

startNow: non-zero value specifies that the continuous acquisition should start immediately. If the value is zero, you must manually start the acquisition with [imgSessionStartAcquisition](#).

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information about the error code, call [imgShowError](#).

imgGrab

Usage

```
rval imgGrab(SESSION_ID sid, void** bufAddr, uInt32 waitForNext);
```

Purpose

Acquires the most current frame into the specified buffer. Call this function only after calling [imgGrabSetup](#).

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufAddr	void**	input/output
waitForNext	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufAddr: pointer to a pointer to an area of memory in which to store the image. If **bufAddr** points to a NULL pointer, this call allocates a buffer, acquires an image, and returns the buffer address in the location specified by **bufAddr**. If **bufAddr** points to a non-NULL pointer, this function acquires into that buffer.

waitForNext: if zero, the function returns the most recently acquired image. If non-zero, the function waits for and returns the next acquired image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information about the error code, call [imgShowError](#).

imgGrabArea

Usage

```
rval imgGrabArea(SESSION_ID sid, void** bufAddr, uInt32 waitForNext,  
uInt32 top, uInt32 left, uInt32 height, uInt32 width, uInt32 rowPixels);
```

Purpose

Performs a transfer from a continuous acquisition. This function operates similarly to [imgGrab](#), but imgGrabArea only acquires a portion of the image. Call this function only after calling [imgGrabSetup](#).

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufAddr	void**	input/output
waitForNext	UInt32	input
top	UInt32	input
left	UInt32	input
height	UInt32	input
width	UInt32	input
rowPixels	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufAddr: pointer to a pointer to an area of memory in which to store the image. If **bufAddr** points to a NULL pointer, this call allocates a buffer, acquires an image, and returns the buffer address in the location specified by **bufAddr**. If **bufAddr** points to a non-NULL pointer, this function acquires into that buffer.

waitForNext: if zero, the function returns the most recently acquired image. If non-zero, the function waits for and returns the next acquired image.

top: top offset of the first pixel to acquire.

left: left offset of the first pixel to acquire.

height: height of the area to acquire.

width: width of the area to acquire.

rowPixels: total number of pixels in each image line. Passing a zero for this parameter causes the function to ignore the parameter and use the IMG_ATTR_ROWPIXELS attribute instead. Use this parameter for byte alignment or if the image buffer contains a border for image processing.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information about the error code, call [imgShowError](#).

Ring and Sequence Functions

Use the ring and sequence functions to perform a multibuffered acquisition that stops after all buffers are filled (sequence) or continually loops through the buffers (ring).

To use the ring and sequence functions, you must first call `imgRingSetup` or `imgSequenceSetup` to configure the session and optionally start the acquisition process. If you do not start the acquisition using `imgRingSetup` or `imgSequenceSetup`, you must call `imgSessionStartAcquisition` to start the acquisition.

[imgRingSetup](#)

[imgSequenceSetup](#)

[imgSessionStartAcquisition](#)

[imgSessionStopAcquisition](#)

imgRingSetup

Usage

```
rval imgRingSetup(SESSION_ID sid, uInt32 numberOfBuffers, void*  
bufferList[ ], uInt32 skipCount, uInt32 startNow);
```

Purpose

Prepares a session for acquiring continuously and looping into a buffer list.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
numberOfBuffers	UInt32	input
bufferList	void*[]	input/output
skipCount	UInt32	input
startNow	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID

numberOfBuffers: number of buffers in the buffer list.

bufferList[]: array of buffer pointers. For each element in the buffer list that is initialized to NULL, **bufferList[]** allocates a buffer and returns this buffer address in the array element. This function acquires into the buffer for each element that is not NULL.

skipCount: number of images to skip before acquiring each buffer. This number is the same for all acquisitions.

-  **Note** skipCount is not supported for line scan acquisitions. Refer to [imgSessionLineTrigSource2](#) for information about triggering line scan skip triggers.
-  **Note** skipCount is not supported on the NI 1427, NI 1429, or NI 1430.

startNow: non-zero value specifies that the continuous acquisition should start immediately. If the value is zero, you must manually start the acquisition with [imgSessionStartAcquisition](#).

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSequenceSetup

Usage

```
rval imgSequenceSetup(SESSION_ID sid, uInt32 numberOfBuffers, void*
bufferList[ ], uInt32 skipCount[ ], uInt32 startNow, uInt32 async);
```

Purpose

Prepares a session for acquiring a sequence into the buffer list.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
numberOfBuffers	UInt32	input
bufferList	void*[]	input/output
skipCount	UInt32[]	input
startNow	UInt32	input
async	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID

numberOfBuffers: number of buffers in the buffer list.

bufferList[]: array of buffer pointers. For each element in the buffer list that is initialized to NULL, **bufferList[]** allocates a buffer and returns this buffer address in the array element. This function acquires into the buffer for each element that is not NULL.

skipCount[]: array containing the number of images to skip before each acquisition.

-  **Note** skipCount is not supported for line scan acquisitions. Refer to [imgSessionLineTrigSource2](#) for information about triggering line scan skip triggers.
-  **Note** skipCount is not supported on the NI 1427, NI 1429, or NI 1430.

startNow: non-zero value specifies that the continuous acquisition should start immediately. If the value is zero, you must manually start the acquisition with [imgSessionStartAcquisition](#).

async: if **async** is zero and **startNow** is non-zero, this function does not return until the acquisition completes. Otherwise, the function returns immediately.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionStartAcquisition

Usage

```
rval imgSessionStartAcquisition(SESSION_ID sid);
```

Purpose

Starts an acquisition asynchronously in the session identified by **sid**.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionStopAcquisition

Usage

```
rval imgSessionStopAcquisition(SESSION_ID sid);
```

Purpose

Stops the acquisition in the session identified by **sid**. This function performs the least amount of work necessary to stop the acquisition, and also maintains configuration information. The configuration time of subsequent acquisitions is minimized as long as no acquisition parameters change. The buffer list configuration is also preserved. To disassociate the buffer list and clear the acquisition configuration, use [imgSessionAbort](#).

Parameters

Name	Type	Direction
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

Signal I/O Functions

Use the signal I/O functions to control the trigger lines on image acquisition devices. You can use these functions to start an acquisition based on a trigger, output status signals on a trigger line, wait for a specified signal to occur, or output pulses on the trigger lines.

[imgSessionTriggerConfigure2](#)

[imgSessionLineTrigSource2](#)

[imgSessionTriggerClear](#)

[imgSessionTriggerDrive2](#)

[imgSessionTriggerRoute2](#)

[imgSessionTriggerRead2](#)

[imgSessionWaitSignal2](#)

[imgSessionWaitSignalAsync2](#)

[imgEncoderResetPosition](#)

[imgPulseCreate2](#)

[imgPulseDispose](#)

[imgPulseRate](#)

[imgPulseStart](#)

[imgPulseStop](#)

imgSessionTriggerConfigure2

Usage

```
rval imgSessionTriggerConfigure2(SESSION_ID sid, IMG_SIGNAL_TYPE  
triggerType, uInt32 triggerNumber, uInt32 polarity, uInt32 timeout, uInt32  
action);
```

Purpose

Configures an acquisition to start based on an external trigger.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
triggerType	IMG_SIGNAL_TYPE	input
triggerNumber	UInt32	input
polarity	UInt32	input
timeout	UInt32	input
action	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

triggerType: type of trigger line to use. **triggerType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

triggerNumber: number of the trigger line to use.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEV` Triggers on a falling edge.

`IMG_TRIG_POLAR_ACTIVEH` Triggers on a rising edge.

timeout: time, in milliseconds, to wait for the trigger to occur.

action: action to take when the trigger edge occurs. The following are valid values for **action**:

`IMG_TRIG_ACTION_NONE` Clears the trigger.

`IMG_TRIG_ACTION_CAPTURE` Starts acquiring.

`IMG_TRIG_ACTION_BUFLIST` Acquires the buffer list once.

`IMG_TRIG_ACTION_BUFFER` Acquires a single image.

`IMG_TRIG_ACTION_STOP` Stops the acquisition.

-  **Note** When using `IMG_TRIG_ACTION_STOP`, the device will continue acquiring a variable number of post-trigger buffers before stopping the acquisition. Use the `IMG_ATTR_NUM_POST_TRIGGER_BUFFERS` attribute to set the number of post-trigger buffers.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionLineTrigSource2

Usage

```
rval imgSessionLineTrigSource2(SESSION_ID sid, IMG_SIGNAL_TYPE  
triggerType, uInt32 triggerNumber, uInt32 polarity, uInt32 skipCount);
```

Purpose

Configures triggering per line for acquisition from a line scan camera. Use this function to require a trigger to start the acquisition of each line from a line scan camera.



Note This function requires a camera that has the functionality to externally control the line rate. To implement this functionality, the external line rate signal must be cabled to one of the image acquisition control lines.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
triggerType	IMG_SIGNAL_TYPE	input
triggerNumber	UInt32	input
polarity	UInt32	input
skipCount	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

triggerType: type of trigger line to use. **triggerType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

[IMG_SIGNAL_SCALED_ENCODER](#)

triggerNumber: number of the trigger line to use.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEV` Triggers on a falling edge.

`IMG_TRIG_POLAR_ACTIVEH` Triggers on a rising edge.

skipCount: number of triggers to skip before acquiring a new line. For example, if you are using an encoder to trigger lines and it outputs 1,000 ticks per revolution, but you want to acquire only 10 lines per revolution, set this parameter to 99. Set this parameter to 0 to acquire a line on every trigger.

-  **Note** Typically, this parameter is unnecessary when the trigger type is [IMG_SIGNAL_SCALED_ENCODER](#) because the scaled encoder signal can implement a skip count by using the `IMG_ATTR_ENCODER_DIVIDE_FACTOR` attribute.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionTriggerClear

Usage

```
rval imgSessionTriggerClear(SESSION_ID sid);
```

Purpose

Disables all triggers on the session.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionTriggerDrive2

Usage

```
rval imgSessionTriggerDrive2(SESSION_ID sid, IMG_SIGNAL_TYPE  
triggerType, uInt32 triggerNumber, uInt32 polarity, uInt32 source);
```

Purpose

Configures the specified trigger line to drive a signal out.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
triggerType	IMG_SIGNAL_TYPE	input
triggerNumber	UInt32	input
polarity	UInt32	input
source	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

triggerType: type of trigger line to use. **triggerType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_OUT](#)

triggerNumber: number of the trigger line to use.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEV` Drives the line low when the signal is true.

`IMG_TRIG_POLAR_ACTIVEH` Drives the line high when the signal is true.

source: specifies the signal that drives the trigger line as specified by the following constants:

`IMG_TRIG_DRIVE_DISABLED`

`IMG_TRIG_DRIVE_AQ_IN_PROGRESS`

`IMG_TRIG_DRIVE_AQ_DONE`

`IMG_TRIG_DRIVE_ASSERTED`

`IMG_TRIG_DRIVE_UNASSERTED`

`IMG_TRIG_DRIVE_HSYNC`

`IMG_TRIG_DRIVE_VSYNC`

`IMG_TRIG_DRIVE_FRAME_START`

`IMG_TRIG_DRIVE_FRAME_DONE`

IMG_TRIG_DRIVE_SCALED_ENCODER

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionTriggerRead2

Usage

```
rval imgSessionTriggerRead2(SESSION_ID sid, IMG_SIGNAL_TYPE  
triggerType, uInt32 triggerNumber, uInt32 polarity, uInt32* status);
```

Purpose

Reads the current value of the specified trigger line.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
triggerType	IMG_SIGNAL_TYPE	input
triggerNumber	UInt32	input
polarity	UInt32	input
status	UInt32*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

triggerType: type of trigger line to use. **triggerType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

[IMG_SIGNAL_ISO_OUT](#)

triggerNumber: number of the trigger line to use.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEL` Asserts the trigger line when the signal is low.

`IMG_TRIG_POLAR_ACTIVEH` Asserts the trigger line when the signal is high.

status: pointer to a variable to receive the state of the trigger. Upon return, the function sets **status** to a non-zero value if the trigger is asserted and to a zero value if the trigger is unasserted.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionTriggerRoute2

Usage

```
rval imgSessionTriggerRoute2(SESSION_ID sid, IMG_SIGNAL_TYPE  
srcTrigType, uInt32 srcTrigNumber, IMG_SIGNAL_TYPE dstTrigType, uInt32  
dstTrigNumber);
```

Purpose

Drives the destination trigger line with the signal on the source trigger line.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
srcTrigType	IMG_SIGNAL_TYPE	input
srcTrigNumber	UInt32	input
dstTrigType	IMG_SIGNAL_TYPE	input
dstTrigNumber	UInt32	input

Parameter Discussion

sid: valid SESSION_ID.

srcTrigType: type of source trigger line. To stop driving, set the **srcTrigType** to IMG_SIGNAL_NONE. **srcTrigType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_NONE](#)

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

srcTrigNumber: number of the source trigger line to use.

dstTrigType: type of destination trigger line. **dstTrigType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in MAX.

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_OUT](#)

dstTrigNumber: number of the destination trigger to use.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionWaitSignal2

Usage

```
rval imgSessionWaitSignal2(SESSION_ID sid, IMG_SIGNAL_TYPE  
signalType, uInt32 signalIdentifier, uInt32 state, uInt32 timeout);
```

Purpose

Waits for a signal to be in a given state. This function returns when either the specified signal is in a given state or the wait times out.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
signalType	IMG_SIGNAL_TYPE	input
signalIdentifier	UInt32	input
state	UInt32	input
timeout	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

signalType: type of signal to wait for. **signalType** can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

[IMG_SIGNAL_ISO_OUT](#)

[IMG_SIGNAL_STATUS](#)

signalIdentifier: is the identifier of the signal to wait for. If the signal type is one of the triggers, then this value specifies which trigger line. If the signal type is IMG_SIGNAL_STATUS, **signalIdentifier** must be one of the following constants:

[IMG_AQ_DONE](#)

[IMG_AQ_IN_PROGRESS](#)

[IMG_BUF_COMPLETE](#)

[IMG_FRAME_DONE](#)

[IMG_FRAME_START](#)

state: state of the signal to wait for. **state** can be one of the following constants:

IMG_SIGNAL_STATE_RISING Waits for a rising edge.

IMG_SIGNAL_STATE_FALLING Waits for a falling edge.

IMG_SIGNAL_STATE_HIGH Returns immediately if the signal is high. Otherwise, waits for a rising edge.

IMG_SIGNAL_STATE_LOW Returns immediately if the signal is

low. Otherwise, waits for a falling edge.

timeout: time, in milliseconds, to wait for the appropriate state. If the appropriate state does not occur within the timeout period, the function returns IMG_ERR_TIMEOUT.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionWaitSignalAsync2

Usage

```
rval imgSessionWaitSignalAsync2(SESSION_ID sid, IMG_SIGNAL_TYPE  
signalType, uInt32 signalIdentifier, uInt32 state, CALL_BACK_PTR2 function,  
void* data);
```

Purpose

Waits for a signal to be in a given state and when the signal is in that state, calls a user-defined function.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
signalType	IMG_SIGNAL_TYPE	input
signalIdentifier	UInt32	input
state	UInt32	input
function	CALL_BACK_PTR	input
data	void*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

signalType: type of signal to wait for. The signal can be one of the following constants:

-  **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

[IMG_SIGNAL_ISO_OUT](#)

[IMG_SIGNAL_STATUS](#)

signalIdentifier: is the signal identifier. If the signal type is one of the triggers, then this value specifies which trigger line. If the signal type is IMG_SIGNAL_STATUS, **signalIdentifier** must be one of the following constants:

[IMG_AQ_DONE](#)

[IMG_AQ_IN_PROGRESS](#)

[IMG_BUF_COMPLETE](#)

[IMG_FRAME_DONE](#)

[IMG_FRAME_START](#)

state: state of the signal to wait for. **state** can be one of the following constants:

IMG_SIGNAL_STATE_RISING Waits for a rising edge.

IMG_SIGNAL_STATE_FALLING Waits for a falling edge.

IMG_SIGNAL_STATE_HIGH Returns immediately if the signal is high. Otherwise, waits for a rising edge.

IMG_SIGNAL_STATE_LOW Returns immediately if the signal is

low. Otherwise, waits for a falling edge.

function: pointer to the callback function. Your function should match the following prototype:

```
uInt32(*function)(SESSION_ID sid, IMG_ERR err, IMG_SIGNAL_TYPE  
signalType, uInt32 signalIdentifier, void* userdata)
```

 **Note** The return value of the callback function determines the behavior of the driver for subsequent signal assertions. Return zero to disregard future signal assertions. Return a non-zero value to continue to receive callbacks.

data value that is passed to the callback function. The value can be a pointer to user data.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgEncoderResetPosition

Usage

```
rval imgEncoderResetPosition(SESSION_ID sid);
```

Purpose

Resets the absolute encoder position counter to 0. Read this position counter value by querying the attribute **IMG_ATTR_ENCODER_POSITION**.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPulseCreate2

Usage

```
rval imgPulseCreate2(UInt32 timebase, UInt32 delay, UInt32 width,  
IMG_SIGNAL_TYPE signalType, UInt32 signalIdentifier, UInt32 signalPolarity,  
IMG_SIGNAL_TYPE outputType, UInt32 outputNumber, UInt32 outputPolarity,  
UInt32 pulseMode, PULSE_ID* plsID);
```

Purpose

Configures the attributes of a pulse. A single pulse consists of a delay phase (phase 1), followed by a pulse phase (phase 2), and then a return to the phase 1 level.

Only the NI PCI/PXI-1409, NI PCI-1410, NI PCI/PXI-1422, NI PCI-1424, NI PCI-1426, NI PCIe-1427, NI PCI/PXI-1428, NI PCIe-1429, and NI PCIe-1430 can generate pulses.

- The NI 1409, NI 1410, NI 1422, NI 1424, NI 1426, and NI 1428 can generate a maximum of 2 pulses.
- The NI 1427 and NI 1429 can generate a maximum of 3 pulses.
- The NI 1430 can generate a maximum of 6 pulses.

Parameters

Name	Type	Direction
timebase	ulInt32	input
delay	ulInt32	input
width	ulInt32	input
signalType	ulInt32	input
signalIdentifier	ulInt32	input
signalPolarity	ulInt32	input
outputType	ulInt32	input
outputNumber	ulInt32	input
outputPolarity	ulInt32	input
pulseMode	ulInt32	input
plsID	PULSE_ID*	output
rval	Int32	output

Parameter Discussion

timebase: timebase of the counter. **timebase** has the following possible values:

PULSE_TIMEBASE_PIXELCLK	Specifies the incoming pixel clock from the camera to use as a timebase for pulse generation.
PULSE_TIMEBASE_100KHZ	Specifies a 100 kHz timebase to use for pulse generation.
PULSE_TIMEBASE_50MHZ	Specifies a 50 MHz timebase to use for pulse generation.
PULSE_TIMEBASE_SCALED_ENCODER	Specifies scaled encoder counts as units for pulse generation.

delay: duration of the first phase of the pulse. Use the following formula to determine the actual time period that delay represents:

$$\text{delay} \times (\text{timebase resolution})$$

width: duration of the second phase of the pulse, phase 2. The unit is cycles of the timebase. Use the following formula to determine the actual time period that width represents:

$$\text{width} \times (\text{timebase resolution})$$

signalType: type of signal that will initiate the pulse generation.

signalType can be one of the following constants:



Note To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in Measurement & Automation Explorer (MAX).

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_IN](#)

IMG_SIGNAL_STATUS

signalIdentifier: is the identifier of the signal that will initiate the pulse generation. If the signal type is one of the triggers, then this value specifies which trigger line. If the signal type is IMG_SIGNAL_STATUS, **signalIdentifier** can be one of the following constants:

[IMG_AQ_DONE](#)

[IMG_AQ_IN_PROGRESS](#)

[IMG_FRAME_DONE](#)

[IMG_FRAME_START](#)

[IMG_FRAME_VALID](#)

[IMG_IMMEDIATE](#)

[IMG_LINE_VALID](#)

signalPolarity: polarity of the signal input as defined by the following constants:

IMG_TRIG_POLAR_ACTIVEV Triggers on a falling edge

IMG_TRIG_POLAR_ACTIVEH Triggers on a rising edge

outputType: type of trigger line on which the pulse is generated.

outputType can be one of the following constants:

 **Note** To use the ISO_IN or RS422_IN signals on the NI PCI-1426, select **External** as the **Trigger Type** for your function and choose **ISO In** or **RS-422 In** as the **Signal Level** for the trigger line in MAX.

[IMG_SIGNAL_EXTERNAL](#)

[IMG_SIGNAL_RTSI](#)

[IMG_SIGNAL_ISO_OUT](#)

outputNumber: number of the trigger line on which the pulse is generated.

outputPolarity: polarity of the pulse output as defined by the following constants:

IMG_PULSE_POLAR_ACTIVEV Drives the line high during the delay

phase, and drives the line low during the pulse phase

`IMG_PULSE_POLAR_ACTIVEH` Drives the line low during the delay phase, and drives the line high during the pulse phase

pulseMode: value that indicates if the pulse is generated once or continuously. **pulseMode** can be one of the following constants:

`PULSE_MODE_TRAIN`

Pulse is generated continuously after the trigger is asserted. Choose this option to generate a continuous pulse train that is inactive for the number of cycles specified in the **delay** parameter, and active for the number of cycles specified in the **width** parameter. When the pulse train is started, it continues periodically until you call `imgPulseStop`, `imgPulseDispose`, or `imgClose`.

`PULSE_MODE_SINGLE`

Pulse occurs one time when the first trigger occurs. Choose this option to generate a single pulse. On the first occurrence of `signal_source`, the output line stays inactive for the number of cycles specified in the **delay** parameter, and becomes active for the number of cycles specified in the **width** parameter. Future occurrences of `signal_source` do not affect the output line.

`PULSE_MODE_SINGLE_REARM`

Pulse occurs one time on each trigger occurrence. Choose this option to generate a rearmed single shot pulse. On every occurrence of `signal_source`, the output line stays inactive for the number of cycles

specified in the **delay** parameter, and becomes active for the number of cycles specified in the **width** parameter. When the pulse is started, output toggles for each occurrence of signal_source until you call imgPulseStop, imgPulseDispose, or imgClose.

plsID: pointer to a variable to receive the pulse ID. If the function succeeds, the variable is populated with a valid PULSE_ID that can be used in subsequent functions.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPulseDispose

Usage

```
rval imgPulseDispose(PULSE_ID plsID);
```

Purpose

Disposes a pulse. Disposing a pulse stops and disarms the pulse.

Parameters

Name	Type	Direction
plID	PULSE_ID	input
rval	Int32	output

Parameter Discussion

pIsID: ID of the pulse to dispose.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPulseRate

Usage

```
rval imgPulseRate(double delayTime, double widthTime, uInt32* delay, uInt32*  
width, uInt32* timebase);
```

Purpose

Converts delay time and width time into delay, width, and timebase values used by [imgPulseCreate](#).

Parameters

Name	Type	Direction
delayTime	double	input
widthTime	double	input
delay	ulInt32*	output
width	ulInt32*	output
timebase	ulInt32*	output
rval	Int32	output

Parameter Discussion

delayTime: duration of the first phase of the pulse, in seconds.

widthTime: duration of the second phase of the pulse, in seconds.

delay: on return, the function populates the variable with the number of cycles of the first phase of the pulse.

width: on return, the function populates the variable with the number of cycles of timebase of the second phase of the pulse.

timebase: on return, the function populates the variable with the code that represents the timebase on the device that the counter uses to produce the pulse.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPulseStart

Usage

```
rval imgPulseStart(PULSE_ID pid, SESSION_ID sid);
```

Purpose

Arms a pulse.

Parameters

Name	Type	Direction
pid	PULSE_ID	input
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

pid: ID of the pulse to arm.

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPulseStop

Usage

```
rval imgPulseStop(PULSE_ID pid);
```

Purpose

Stops and disarms a pulse.

Parameters

Name	Type	Direction
pid	PULSE_ID	input
rval	Int32	output

Parameter Discussion

pid: ID of the pulse to stop.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

Miscellaneous Functions

Use miscellaneous functions to obtain status information for a session, get and set a region of interest (ROI), and get the buffer size required for a session based on current attributes.

[imgSessionStatus](#)

[imgSessionGetROI](#)

[imgSessionConfigureROI](#)

[imgSessionFitROI](#)

[imgSessionGetBufferSize](#)

imgSessionStatus

Usage

```
rval imgSessionStatus(SESSION_ID sid, uInt32* status, uInt32* bufferIndex);
```

Purpose

Gets the current session status.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
status	UInt32*	output
bufferIndex	UInt32*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

status: on return, the function populates **status** with zero if the session is not acquiring or a non-zero value if the session is acquiring.

bufferIndex: on return, the function populates **bufferIndex** with the last available buffer list index.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionGetROI

Usage

```
rval imgSessionGetROI(SESSION_ID sid, uInt32* top, uInt32* left, uInt32*  
height, uInt32* width);
```

Purpose

Gets the acquisition region of interest (ROI).

Parameters

Name	Type	Direction
sid	SESSION_ID	input
top	ulnt32*	output
left	ulnt32*	output
height	ulnt32*	output
width	ulnt32*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

top: on return, the function sets **top** to the current value of IMG_ATTR_ROI_TOP.

left: on return, the function sets **left** to the current value of IMG_ATTR_ROI_LEFT.

height: on return, the function sets **height** to the current value of IMG_ATTR_ROI_HEIGHT.

width: on return, the function sets **width** to the current value of IMG_ATTR_ROI_WIDTH.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionConfigureROI

Usage

```
rval imgSessionConfigureROI(SESSION_ID sid, uInt32 top, uInt32 left, uInt32  
height, uInt32 width);
```

Purpose

Sets the acquisition region of interest (ROI). This function modifies the following attributes:

IMG_ATTR_ROI_TOP

IMG_ATTR_ROI_LEFT

IMG_ATTR_ROI_HEIGHT

IMG_ATTR_ROI_WIDTH

To use imgSessionConfigureROI, the ROI you set must be properly aligned. Use [imgSessionFitROI](#) function to ensure that the ROI is properly aligned.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
top	ulnt32	input
left	ulnt32	input
height	ulnt32	input
width	ulnt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

top: top offset of the first pixel to acquire.

left: left offset of the first pixel to acquire.

height: height of the area to acquire.

width: width of the area to acquire.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionFitROI

Usage

```
rval imgSessionFitROI(SESSION_ID boardid, IMG_ROI_FIT_MODE fitMode,  
uInt32 top, uInt32 left, uInt32 height, uInt32 width, uInt32* fittedTop, uInt32*  
fittedLeft, uInt32* fittedHeight, uInt32* fittedWidth);
```

Purpose

Evaluates the region of interest (ROI) you specify, and returns valid left, top, height, and width values. Returns an ROI that NI-IMAQ is guaranteed to be able to acquire and that best accommodates the ROI you specify.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
fitMode	IMG_ROI_FIT_MODE	input
top	UInt32	input
left	UInt32	input
height	UInt32	input
width	UInt32	input
fittedTop	UInt32*	output
fittedLeft	UInt32*	output
fittedHeight	UInt32*	output
fittedWidth	UInt32*	output

Parameter Discussion

sid: valid SESSION_ID.

fitMode: next smallest or next largest acquirable ROI. The following are valid values for this parameter:

[IMG_ROI_FIT_LARGER](#)

[IMG_ROI_FIT_SMALLER](#)

top: top coordinate to fit.

left: left coordinate to fit.

height: height coordinate to fit.

width: width coordinate to fit.

fittedTop: fitted ROI top coordinate.

fittedLeft: fitted ROI left coordinate.

fittedHeight: fitted ROI height coordinate.

fittedWidth: fitted ROI width coordinate.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionGetBufferSize

Usage

```
rval imgSessionGetBufferSize(SESSION_ID sid, uInt32* sizeNeeded);
```

Purpose

Gets the size, in bytes, of an image.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
sizeNeeded	UInt32*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

sizeNeeded: size, in bytes, of an image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note If you change the ROI after calling this function, the return value is no longer accurate.

Low-Level Functions

Use low-level functions to perform in-depth tasks that require more control of the image acquisition hardware.

[Acquisition Functions](#)

[Attribute Functions](#)

[Buffer Management Functions](#)

[Utility Functions](#)

[Serial Communication Functions](#)

Acquisition Functions

Use acquisition functions to configure, start, and abort an image acquisition. These functions also let you examine a buffer during acquisition.

[imgSessionAbort](#)
[imgSessionAcquire](#)
[imgSessionConfigure](#)
[imgSessionCopyArea](#)
[imgSessionCopyBuffer](#)
[imgSessionExamineBuffer](#)
[imgSessionReleaseBuffer](#)

imgSessionAbort

Usage

```
rval imgSessionAbort(SESSION_ID sid, uInt32* bufferIndex);
```

Purpose

Stops an acquisition immediately. This function clears all acquisition configuration, disassociates the session from the buffer list, and unregisters all acquisition triggers. Prior to starting a new acquisition, the buffer list and any acquisition triggers need to be reassociated with the session via [imgSessionConfigure](#) and [imgSessionTriggerConfigure2](#), respectively. Use [imgSessionStopAcquisition](#) to stop an acquisition and maintain the current session configuration.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufferIndex	UInt32*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufferIndex: on return, the function populates **bufferIndex** with the last available buffer list index.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionAcquire

Usage

```
rval imgSessionAcquire(SESSION_ID sid, uInt32 async, CALL_BACK_PTR  
callback);
```

Purpose

Starts an acquisition, synchronously or asynchronously, to the buffers in the associated session buffer list.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
async	UInt32	input
callback	CALLBACK_PTR	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

async: asynchronous flag. If **async** is zero, this function does not return until the acquisition completes.

callback: pointer to the callback function. If **async** is non-zero, this callback function is called under one of the following two conditions:

- If the acquisition is non-continuous, the callback is called when all buffers are acquired.
- If the acquisition is continuous, the callback is called after each buffer becomes available.

Your function must match the following prototype:

```
uInt32(*function)(SESSION_ID sid, IMG_ERR err, uInt32 signal, void*  
userdata)
```



Note For non-continuous acquisitions, the callback function must return zero. For continuous acquisitions, the return value of the callback function determines the behavior of the driver for subsequent buffer completions. Return zero to disregard future buffer complete notifications. Return a non-zero value to continue to receive callbacks.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionConfigure

Usage

```
rval imgSessionConfigure(SESSION_ID sid, BUFLIST_ID bulist);
```

Purpose

Configures the hardware in preparation for an acquisition using the given buffer list. You must pass a valid BUFLIST_ID. Upon successful completion of this call, you can call [imgSessionAcquire](#).

Parameters

Name	Type	Direction
sid	SESSION_ID	input
buflist	BUFLIST_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

buflist: valid BUFLIST_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionCopyArea

Usage

```
rval imgSessionCopyArea (SESSION_ID sid, uInt32 bufferIndex, uInt32 top,  
uInt32 left, uInt32 height, uInt32 width, void* buffer, uInt32 rowPixels, uInt32  
waitForNext);
```

Purpose

Copies an area of a session buffer to a user-specified buffer.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufferIndex	UInt32	input
top	UInt32	input
left	UInt32	input
width	UInt32	input
height	UInt32	input
buffer	void*	input
rowPixels	UInt32	input
vsync	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufferIndex: valid buffer list index from which to copy.

top: top coordinate of the area to copy.

left: left coordinate of the area to copy.

height: height of the area to copy.

width: width of the area to copy.

buffer: buffer that receives the image.

rowPixels: number of pixels in each image line of the destination buffer.

Passing a zero for this parameter causes the function to ignore the parameter and use the IMG_ATTR_ROWPIXELS attribute instead. Use this parameter for byte alignment or if the image buffer contains a border for image processing.

waitForNext: if zero, the function copies immediately. If non-zero, the function waits until the current acquisition is complete before copying.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionCopyBuffer

Usage

```
rval imgSessionCopyBuffer(SESSION_ID sid, uInt32 bufferIndex, void* buffer,  
uInt32 waitForNext);
```

Purpose

Copies session image data to a user buffer.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
bufferIndex	UInt32	input
buffer	void*	input
waitForNext	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

bufferIndex: valid buffer list index from which to copy.

buffer: points to an area of memory to receive the copy.

waitForNext: if zero, the function copies immediately. If non-zero, the function waits until the current acquisition is complete before copying.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionExamineBuffer2

Usage

```
rval imgSessionExamineBuffer2(SESSION_ID sid, uInt32 whichBuffer, void*  
bufferNumber, void** bufferAddr);
```

Purpose

Extracts an image from a live acquisition. This function lets you lock an image out of a continuous loop sequence for processing when you are using a ring (continuous) sequence. If the requested image has been acquired and exists in memory, the function returns that image immediately. If the requested image has not yet been acquired, the function does not return until the image has been acquired or the timeout period has expired. If the requested image has already been overwritten, the function returns the most current image. If the buffer remains extracted long enough that the acquisition hardware wraps around the buffer list and encounters the extracted buffer again, the acquisition will stall, increment the lost frame count, and the extracted buffer will not be overwritten.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
whichBuffer	ulInt32	input
bufferNumber	void*	input
bufferAddr	void**	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

whichBuffer: cumulative image number to extract. Pass IMG_CURRENT_BUFFER to get the buffer that is currently being acquired.

bufferNumber: on return, the function populates this parameter with the cumulative number of the returned image.

bufferAddr: on return, the function populates this parameter with a pointer to the locked image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note Use [imgSessionReleaseBuffer](#) to release the image being held with imgSessionExamineBuffer.

imgSessionReleaseBuffer

Usage

```
rval imgSessionReleaseBuffer(SESSION_ID sid);
```

Purpose

Releases an image that was previously held with [imgSessionExamineBuffer](#). This function has the effect of re-entering an image into a continuous ring buffer pool after analysis.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionCopyAreaByNumber

Usage

```
rval imgSessionCopyAreaByNumber (SESSION_ID boardid, UInt32  
bufNumber, UInt32 top, UInt32 left, UInt32 height, UInt32 width, void*  
userBuffer, UInt32 rowPixels, IMG_OVERWRITE_MODE overwriteMode,  
UInt32* copiedNumber, UInt32* copiedIndex);
```

Purpose

Copies an area of a session buffer to a user-specified buffer.

Parameters

Name	Type	Direction
boardid	SESSION_ID	input
bufNumber	ulInt32	input
top	ulInt32	input
left	ulInt32	input
height	ulInt32	input
width	ulInt32	input
userBuffer	void*	input
rowPixels	ulInt32	input
overwriteMode	IMG_OVERWRITE_MODE	input
copiedNumber	ulInt32*	output
copiedIndex	ulInt32*	output

Parameter Discussion

boardid: valid SESSION_ID.

bufNumber: cumulative buffer number.

top: top coordinate of the area to copy.

left: left coordinate of the area to copy.

height: height of the area to copy.

width: width of the area to copy.

userBuffer: image to plot.

rowPixels: number of pixels in each image line of the destination buffer.

Passing a zero for this parameter causes the function to ignore the parameter and use the IMG_ATTR_ROWPIXELS attribute instead. Use this parameter for byte alignment or if the image buffer contains a border for image processing.

overwriteMode: gets/sets the overwrite mode, used to determine acquisition when an image transfer cannot be completed due to an overwritten internal buffer.

copiedNumber: actual cumulative number provided.

copiedIndex: actual cumulative index provided.

Return Value

This function returns zero on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionCopyBufferByNumber

Usage

```
rval imgSessionCopyArea (SESSION_ID boardid, uInt32 bufNumber, void*
userBuffer, IMG_OVERWRITE_MODE overwriteMode, uInt32*
copiedNumber, uInt32* copiedIndex);
```

Purpose

Copies an area of a session buffer to a user-specified buffer.

Parameters

Name	Type	Direction
boardid	SESSION_ID	input
bufNumber	ulInt32	input
userBuffer	void*	input
overwriteMode	IMG_OVERWRITE_MODE	input
copiedNumber	ulInt32*	input
copiedIndex	ulInt32*	input

Parameter Discussion

boardid: valid SESSION_ID.

bufNumber: cumulative buffer number.

userBuffer: image to plot.

overwriteMode: gets/sets the overwrite mode, used to determine acquisition when an image transfer cannot be completed due to an overwritten internal buffer.

copiedNumber: actual cumulative number provided.

copiedIndex: actual cumulative index provided.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

Attribute Functions

Use attribute functions to examine and change NI-IMAQ and camera attributes.

You can change some attributes while an acquisition is in progress. However, most attributes require that you call [imgSessionConfigure](#) to reconfigure the driver.

NI-IMAQ does not let you make any attribute changes that would have a detrimental effect on any acquisition in progress. If NI-IMAQ lets you change an attribute during a live acquisition, the effects are immediate. If NI-IMAQ does not let you change an attribute during a live acquisition, stop the acquisition, change the attribute, and restart the acquisition.

[imgGetAttribute](#)

[imgSetAttribute](#)

[imgGetCameraAttributeString](#)

[imgSetCameraAttributeString](#)

[imgGetCameraAttributeNumeric](#)

[imgSetCameraAttributeNumeric](#)

[imgSessionSetUserLUT8bit](#)

[imgSessionSetUserLUT16bit](#)

imgGetAttribute

Usage

```
rval imgGetAttribute(UInt32 void_id, UInt32 attr, void* value);
```

Purpose

Returns an attribute value.

Parameters

Name	Type	Direction
------	------	-----------

void_id	ulnt32	input
----------------	--------	-------

attr	ulnt32	input
-------------	--------	-------

value	void*	output
--------------	-------	--------

rval	Int32	output
-------------	-------	--------

Parameter Discussion

void_id: valid SESSION_ID or INTERFACE_ID.

attr: attribute to set. Refer to [Attributes](#) for valid types.

value: pointer to the variable that receives the attribute.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSetAttribute2

Usage

```
rval imgSetAttribute2(UInt32 void_id, UInt32 type, ...);
```

Purpose

Sets an attribute value.

Parameters

Name	Type	Direction
void_id:	ulnt32	input
attr	ulnt32	input
...	variable argument	Data is passed by value. The data type should match the expected attribute type.
rval	Int32	output

Parameter Discussion

void_id valid SESSION_ID or INTERFACE_ID.

attr: attribute to set. Refer to [Attributes](#) for valid types.

Refer to [Attributes](#) for the attribute type.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgGetCameraAttributeString

Usage

```
rval imgGetCameraAttributeString (SESSION_ID sid, const char*  
attributeString, char* currentValue, uInt32 sizeofcurrentValue);
```

Purpose

Gets the value of camera attributes. Camera attributes vary according to which camera you are using. Refer to Measurement & Automation Explorer (MAX) for information about valid camera attributes for your camera and image acquisition device. Use this function to get the value of string or numeric attribute types.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
attributeString	const char*	input
currentValue	char*	output
sizeofcurrentValue	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

attributeString: attribute name as described in the Camera Attributes tab in MAX.

currentValue: pointer to the array to receive the attribute. Upon success, the function populates the variable with the current value of the attribute.

sizeofcurrentValue: size of the **currentValue** that has been passed in. If the size of the string is not large enough to hold the attribute, the attribute is truncated.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information about the error code, call [imgShowError](#).

imgSetCameraAttributeString

Usage

```
rval imgSetCameraAttributeString(SESSION_ID sid, const char*
attributeString, const char* newValue);
```

Purpose

Sets the value of camera attributes. Camera attributes vary according to which camera you are using. Refer to Measurement & Automation Explorer (MAX) for information about valid camera attributes for your camera and image acquisition device. Use this function to set the value of string or numeric attribute types. If the attribute is a numeric type (integer or float), this function converts the string input into a numeric value.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
attributeString	const char*	input
newValue	const char*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

attributeString: attribute name as described in the Camera Attributes tab in MAX.

newValue: new value of the attribute.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgGetCameraAttributeNumeric

Usage

```
rval imgGetCameraAttributeNumeric (SESSION_ID sid, const char*  
attributeString, double* currentValue);
```

Purpose

Gets the value of numeric camera attributes. Camera attributes vary according to which camera you are using. Refer to Measurement & Automation Explorer (MAX) for information about valid camera attributes for your camera and image acquisition device.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
attributeString	const char*	input
currentValue	double*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

attributeString: attribute name as described in the Camera Attributes tab in MAX.

currentValue: pointer to a variable to receive the attribute. Upon success, the function populates the variable with the current value of the attribute.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information about the error code, call [imgShowError](#).

imgSetCameraAttributeNumeric

Usage

```
rval imgSetCameraAttributeNumeric (SESSION_ID sid, const char*  
attributeString, double newValue);
```

Purpose

Sets the value of numeric camera attributes. Camera attributes vary according to which camera you are using. Refer to Measurement & Automation Explorer (MAX) for information about valid camera attributes for your camera and image acquisition device.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
attributeString	const char*	input
newValue	double	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

attributeString: attribute name as described in the Camera Attributes tab in MAX.

newValue: new value of the attribute.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionSetUserLUT8bit

Usage

```
rval imgSessionSetUserLUT8bit(SESSION_ID sid, uInt32 lutType, uInt8* lut);
```

Purpose

Downloads a custom 8-bit lookup table (LUT) to the image acquisition device. Call this function at least once with the default constant (IMG_LUT_TYPE_DEFAULT) to initialize all LUTs, or make successive calls using different constants for each LUT. You also can override the default LUT on a per-tap basis. This function works with analog or digital image acquisition devices acquiring from an 8-bit camera.



Note User LUTs are not supported on the NI 1405, NI 1426, NI 1427, NI 1429, or the NI 1430.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
lutType	UInt32	input
lut	UInt8*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID

lutType: type of LUT to be written. The following are valid values for this parameter:

IMG_LUT_TYPE_DEFAULT	Default LUT used to initialize all LUTs.
IMG_LUT_TYPE_RED	Red channel LUT for RGB digital cameras or the PCI/PXI-1411.
IMG_LUT_TYPE_GREEN	Green channel LUT for RGB digital cameras or the PCI/PXI-1411.
IMG_LUT_TYPE_BLUE	Blue channel LUT for RGB digital cameras or the PCI/PXI-1411.
IMG_LUT_TYPE_TAP0	Tap 0 LUT for digital devices.
IMG_LUT_TYPE_TAP1	Tap 1 LUT for digital devices.
IMG_LUT_TYPE_TAP2	Tap 2 LUT for digital devices.
IMG_LUT_TYPE_TAP3	Tap 3 LUT for digital devices.

 **Note** IMG_LUT_TYPE_RED, IMG_LUT_TYPE_GREEN, and IMG_LUT_TYPE_BLUE are valid only for color cameras.

lut: array that contains the LUT to be downloaded to the image acquisition device. This array must contain 256 elements.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionSetUserLUT16bit

Usage

```
rval imgSessionSetUserLUT16bit(SESSION_ID sid, uInt32 lutType, uInt16*  
lut);
```

Purpose

Downloads a custom 16-bit lookup table (LUT) to the image acquisition device. Call this function at least once with the default constant (IMG_LUT_TYPE_DEFAULT) to initialize all LUTs, or make successive calls using different constants for each LUT. You also can override the default LUT on a per-tap basis. This function works with analog and digital cameras featuring 10-, 12-, 14-, or 16-bit pixel depths.



Note User LUTs are not supported on the NI 1405, NI 1426, NI 1427, NI 1429, or the NI 1430.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
lutType	UInt32	input
lut	UInt16*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

lutType: type of LUT to be written. The following are valid values for this parameter:

IMG_LUT_TYPE_DEFAULT Default LUT used to initialize all LUTs.

IMG_LUT_TYPE_TAP0 Tap 0 LUT for digital devices.

IMG_LUT_TYPE_TAP1 Tap 1 LUT for digital devices.

lut: array that contains the LUT to be downloaded to the image acquisition device. This array must contain 2^n elements where n is the bit depth of the camera (1,024 for 10-bit cameras, 2,048 for 12-bit cameras, and so on).

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

Buffer Management Functions

Use buffer management functions to set up objects such as buffer lists and buffers. When changing buffer list elements, make sure no other sessions depend on that buffer list to be in a known state.

[imgCreateBuffer](#)

[imgCreateBufList](#)

[imgDisposeBuffer](#)

[imgDisposeBufList](#)

[imgGetBufferElement](#)

[imgSetBufferElement](#)

imgCreateBuffer

Usage

```
rval imgCreateBuffer(SESSION_ID sid, uInt32 where, uInt32 bufferSize, void**  
bufPtrAddr);
```

Purpose

Creates a frame buffer based on the region of interest (ROI) in the associated session. Passing NULL or zero for the SESSION_ID is valid. In this case, you must pass a buffer size. If **bufferSize** is zero, the buffer size is computed as follows:

[ROI height] x [rowPixels] x [number of bytes per pixel]

The function returns an error if the buffer size is smaller than the minimum buffer size required for the session.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
where	ulnt32	input
bufferSize	ulnt32	input
bufPtrAddr	void**	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID

where: value that indicates if the buffer should be stored in system memory or in onboard memory on the image acquisition device, as specified by the following constants:

IMG_HOST_FRAME Specifies the new buffer is created in the host computer memory.

IMG_DEVICE_FRAME Specifies the new buffer is created in onboard memory. This feature is not available on the NI PCI/PXI-1407, NI PCIe-1427, NI PCIe-1429, and NI PCIe-1430 devices.

bufferSize: size of the buffer you want to create, in bytes.

bufPtrAddr: pointer to an area of memory that stores the new buffer address.



Note If you use IMG_DEVICE_FRAME, do not access the returned value of **bufPtrAddr**.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgCreateBufList

Usage

```
rval imgCreateBufList(UInt32 numElements, BUFLIST_ID* bid);
```

Purpose

Creates a buffer list. You must initialize the buffer list before calling imgSessionConfigure. Use [imgSetBufferElement](#) to initialize the buffer list.

Parameters

Name	Type	Direction
numElements	UInt32	input
bid	BUFLIST_ID*	output
rval	Int32	output

Parameter Discussion

numElements: number of elements the buffer list should contain.

bid: pointer to an area of memory that contains a BUFLIST_ID variable.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgDisposeBuffer

Usage

```
rval imgDisposeBuffer(void* buffPtrAddr);
```

Purpose

Disposes of a frame buffer created by [imgCreateBuffer](#).

Parameters

Name	Type	Direction
------	------	-----------

bufPtrAddr	void*	input
-------------------	-------	-------

rval	Int32	output
-------------	-------	--------

Parameter Discussion

bufPtrAddr: pointer to buffer created by [imgCreateBuffer](#).

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note Before calling imgDisposeBuffer(), ensure that this buffer is not included in any active buffer lists.

imgDisposeBufList

Usage

```
rval imgDisposeBufList(BUFLIST_ID bid, uInt32 freeResources);
```

Purpose

Disposes either a buffer created by imgCreateBufList() and the buffer list, or disposes only the buffer list.

Parameters

Name	Type	Direction
bid	BUFLIST_ID	input
freeResources	UInt32	input
rval	Int32	output

Parameter Discussion

bid: valid BUFLIST_ID

freeResources: value that determines whether both the buffers and the buffer list or only the buffer list will be disposed. If **freeResources** is non-zero, the function disposes all of the driver-allocated buffers assigned to this list in addition to the buffer list. If **freeResources** is zero, the function disposes only the buffer list.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgGetBufferElement

Usage

```
rval imgGetBufferElement(BUFLIST_ID bid, uInt32 element, uInt32 itemType,  
void* itemValue);
```

Purpose

Gets the value for a specified **itemType** for a buffer in a buffer list.

Parameters

Name	Type	Direction
bid	BUFLIST_ID	input
element	UInt32	input
itemType	UInt32	input
itemValue	void*	output
rval	Int32	output

Parameter Discussion

bid: valid BUFLIST_ID variable.

element: index of the buffer list item to examine.

itemType: describes the type of element to get as specified by the following constants:

IMG_BUFF_ACTUALHEIGHT	Returns the actual height, in lines, of a buffer acquired in VHA mode.
IMG_BUFF_ADDRESS	Specifies the buffer address portion of a buffer list element.
IMG_BUFF_CHANNEL	Specifies the channel from which to acquire an image.
IMG_BUFF_COMMAND	Specifies the command portion of a buffer list element.
IMG_BUFF_SIZE	Specifies the size portion of a buffer list element (the buffer size). Required for user-allocated buffers.
IMG_BUFF_SKIPCOUNT	Specifies the skip count portion of a buffer list element.

itemValue: passes a pointer to an area of memory reserved for the return type (32 bits).

Return Value

This function returns 0 on success. On failure, this function returns an error code. For more information about the error code, call [imgShowError](#).



Note Refer to [Constants](#) for valid element and command types.

imgSetBufferElement2

Usage

```
rval imgSetBufferElement2(BUFLIST_ID bid, uInt32 element, uInt32 itemType,  
...);
```

Purpose

Sets the value for a specified **itemType** for a buffer in a buffer list.

Parameters

Name	Type	Direction
bid	BUFLIST_ID	input
element	ulnt32	input
itemType	ulnt32	input
...	variable argument	Data is passed by value. The data type should match the expected item type.
rval	Int32	output

Parameter Discussion

bid: valid BUFLIST_ID variable.

element: element number of the buffer list item to modify.

itemType: describes the parameter of the element to set, as specified by the following constants:

IMG_BUFF_ADDRESS	Specifies the buffer address portion of a buffer list element.
IMG_BUFF_CHANNEL	Specifies the channel from which to acquire an image.
IMG_BUFF_COMMAND	Specifies the command portion of a buffer list element.
IMG_BUFF_SIZE	Specifies the size portion of a buffer list element (the buffer size). Required for user-allocated buffers.
IMG_BUFF_SKIPCOUNT	Specifies the skip count portion of a buffer list element.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note Refer to [Constants](#) for valid element and command types.

Utility Functions

Use the utility functions to display an image in a window, save an image to a file, or get detailed error information.

[imgPlot](#)

[imgPlotDC](#)

[imgSessionSaveBufferEx](#)

[imgShowError](#)

[imgBayerColorDecode](#)

[imgCalculateBayerColorLUT](#)

imgPlot2

Usage

```
rval imgPlot2(void* hwnd, void* buffer, uInt32 leftBufOffset, uInt32  
topBufOffset, uInt32 xsize, uInt32 ysize, uInt32 xpos, uInt32 ypos, uInt32 flags);
```

Purpose

Plots a buffer to a window given a native Windows handle. Use this function to display a buffer after it is acquired.

Parameters

Name	Type	Direction
hwnd	void*	input
buffer	void*	input
leftBufOffset	ulInt32	input
topBufOffset	ulInt32	input
xsize	ulInt32	input
ysize	ulInt32	input
xpos	ulInt32	input
ypos	ulInt32	input
flags	ulInt32	input
rval	Int32	output

Parameter Discussion

hwnd: native Windows handle designating the window in which to plot.

buffer: image to plot.

leftBufOffset: left offset into the image to start plotting.

topBufOffset: top offset into the image to start plotting.

xsize: width of the image, in pixels.

ysize: number of lines in the image.

xpos: left position to start plotting in the window.

ypos: top position to start plotting in the window.

flags: sets the display property. **flags** is used with the following constants:

IMGPLOT_COLOR_RGB32	Specifies a 32-bit color RGB image.
IMGPLOT_COLOR_HSL32	Specifies a 32-bit color HSL image.
IMGPLOT_INVERT	Specifies to invert the image when plotted.
IMGPLOT_MONO_8	Specifies an 8-bit monochrome image.
IMGPLOT_MONO_10	Specifies a 10-bit monochrome image.
IMGPLOT_MONO_12	Specifies a 12-bit monochrome image.
IMGPLOT_MONO_14	Specifies a 14-bit monochrome image.
IMGPLOT_MONO_16	Specifies a 16-bit monochrome image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPlotDC2

Usage

```
rval imgPlotDC2 (void* hdc, void* buffer, uInt32 xbuffoff, uInt32 ybuffoff,  
uInt32 xsize, uInt32 ysize, uInt32 xscreen, uInt32 yscreen, uInt32 flags);
```

Purpose

Plots a buffer to a device context given a device context handle.

Parameters

Name	Type	Direction
hdc	void*	input
buffer	void*	input
xbuffoff	ulnt32	input
ybuffoff	ulnt32	input
xsize	ulnt32	input
ysize	ulnt32	input
xpos	ulnt32	input
ypos	ulnt32	input
flags	ulnt32	input
rval	Int32	output

Parameter Discussion

hdc: native Windows device context in which to draw.

buffer: image to plot.

xbuffoff: left offset into the image to start plotting.

ybuffoff: top offset into the buffer to start plotting.

xsize: width of the image, in pixels.

ysize: number of lines in the image.

xpos: left position to start plotting in the window.

ypos: top position to start plotting in the window.

flags: sets the display property. **flags** is used with the following constants:

IMGPLOT_INVERT	Specifies to invert the image when plotted.
IMGPLOT_MONO_8	Specifies a 8-bit monochrome image.
IMGPLOT_MONO_10	Specifies a 10-bit monochrome image.
IMGPLOT_MONO_12	Specifies a 12-bit monochrome image.
IMGPLOT_MONO_14	Specifies a 14-bit monochrome image.
IMGPLOT_MONO_16	Specifies a 16-bit monochrome image.
IMGPLOT_COLOR_HSL32	Specifies a 32-bit color HSL image.
IMGPLOT_COLOR_RGB32	Specifies a 32-bit color RGB image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionSaveBufferEx

Usage

```
rval imgSessionSaveBufferEx(SESSION_ID sid, const void* buffer, const char*  
fileName);
```

Purpose

Saves a buffer to disk in BMP, TIFF, or PNG format.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
buffer	const void*	input
fileName	const char*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

buffer: image to save.

-  **Note** The function takes information, such as bits per pixel and buffer size, from the current session settings. Therefore, this buffer must be associated with the current session.

fileName: filename used to save the image. If the filename has the extension .bmp, the function saves the image as a bitmap file. Bitmap files support 8-bit monochrome and 32-bit RGB color images. If the filename extension is .tif, the function saves the image as a TIFF file. TIFF files support 8-bit monochrome and 32-bit RGB color images. If the filename extension is .png, the function saves the image as a portable network graphics (PNG) file. PNG files support all image types.

-  **Note** If you are saving a 10-, 12-, 14-, or 16-bit monochrome image, use the PNG file format.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgShowError

Usage

```
rval imgShowError(IMG_ERR error, char* text);
```

Purpose

Returns a null-terminated string that describes the error code.

Parameters

Name	Type	Direction
error	IMG_ERR	input
text	char*	output
rval	Int32	output

Parameter Discussion

error: valid NI-IMAQ error code.

text: pointer to an array of at least 256 bytes.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgBayerColorDecode

Usage

```
rval imgBayerColorDecode(void* dst, const void* src, uInt32 rows, uInt32 cols,  
uInt32 dstRowPixels, uInt32 srcRowPixels, uInt32* redLUT, uInt32* greenLUT,  
uInt32* blueLUT, uInt8 bayerPattern, uInt8 bitDepth, uInt32 reserved)
```

Purpose

Decodes a Bayer-encoded image to produce an RGB representation of the image.

Parameters

Name	Type	Direction
dst	void*	output
src	const void*	input
rows	ulInt32	input
cols	ulInt32	input
dstRowPixels	ulInt32	input
srcRowPixels	ulInt32	input
redLUT	ulInt32*	input
greenLUT	ulInt32*	input
blueLUT	ulInt32*	input
bayerPattern	ulInt8	input
bitDepth	ulInt8	input
reserved	ulInt32	input

Parameter Discussion

dst: pointer to an area of memory that will contain the decoded image on return.

src: pointer to the source image.

rows: number of rows in the source image.

cols: number of columns in the source image.

dstRowPixels: number of pixels in each row of the destination image, including the borders.

srcRowPixels: number of pixels in each row of the source image, including the borders.

redLUT: pointer to an area of memory that contains the lookup table to be applied to the red pixels. This array must be allocated by the user and contain 256 elements for 8-bit images and 65,536 elements for 16-bit images. Generate the **redLUT** using [imgCalculateBayerColorLUT](#).

greenLUT: pointer to an area of memory that contains the lookup table to be applied to the green pixels. This array must be allocated by the user and contain 256 elements for 8-bit images and 65,536 elements for 16-bit images. Generate the **greenLUT** using [imgCalculateBayerColorLUT](#).

blueLUT: pointer to an area of memory that contains the lookup table to be applied to the blue pixels. This array be must allocated by the user and contain 256 elements for 8-bit images and 65,536 elements for 16-bit images. Generate the **blueLUT** using [imgCalculateBayerColorLUT](#).

bayerPattern: Bayer encoding pattern. Valid patterns include the following values:

IMG_BAYER_PATTERN_GBGB_RGRG Specifies the following Bayer pattern:
GBGB
RGRG

IMG_BAYER_PATTERN_GRGR_BGBG Specifies the following Bayer pattern:
GRGR
BGBG

IMG_BAYER_PATTERN_BGBG_GRGR Specifies the following Bayer

pattern:
BGBG
GRGR

IMG_BAYER_PATTERN_RGRG_GBGB Specifies the following Bayer pattern:
RGRG
GBGB

bitDepth: number of bits per pixel of the source image.

algorithm: reserved for future use. This value must be 0.



Tip For information about Bayer decoding, refer to the White Balancing Utility located at **Start»All Programs»National Instruments»Vision»White Balancing Utility**.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgCalculateBayerColorLUT

Usage

```
rval imgCalculateBayerColorLUT(double redGain, double greenGain, double  
blueGain, UInt32* redLUT, UInt32* greenLUT, UInt32* blueLUT, UInt32  
bitDepth)
```

Purpose

Calculates the red, green, and blue lookup tables (LUTs) based on the input gain values. Call this function before you enter the acquisition loop. You can then pass the LUTs to [imgBayerColorDecode](#) to quickly perform the color decode.

Parameters

Name	Type	Direction
redGain	double	input
greenGain	double	input
blueGain	double	input
redLUT	ulnt32*	output
greenLUT	ulnt32*	output
blueLUT	ulnt32*	output
bitDepth	ulnt32	input

Parameter Discussion

redGain: gain used to generate the **redLUT**. The value must be between 0 and 3.999.

greenGain: gain used to generate the **greenLUT**. The value must be between 0 and 3.999.

blueGain: gain used to generate the **blueLUT**. The value must be between 0 and 3.999.



Tip To find the appropriate values for the gains, use the White Balancing Utility located at **Start»All Programs»National Instruments»Vision»White Balancing Utility**.

redLUT: pointer to an area of memory that will contain the red lookup table on return. For 8-bit images this array must contain 256 elements. For images with bit depths greater than 8, this array must contain 65,536 elements. You must allocate this array before passing it to the function.

greenLUT: pointer to an area of memory that will contain the green lookup table on return. For 8-bit images this array must contain 256 elements. For images with bit depths greater than 8, this array must contain 65,536 elements. You must allocate this array before passing it to the function.

blueLUT: pointer to an area of memory that will contain the blue lookup table on return. For 8-bit images this array must contain 256 elements. For images with bit depths greater than 8, this array must contain 65,536 elements. You must allocate this array before passing it to the function.

bitDepth: bits per pixel of the monochrome source image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

Serial Communication Functions

Use the following serial communication functions with devices that support serial communication.

[imgSessionSerialWrite](#)

[imgSessionSerialRead](#)

[imgSessionSerialReadBytes](#)

[imgSessionSerialFlush](#)

imgSessionSerialWrite

Usage

```
rval imgSessionSerialWrite(SESSION_ID sid, const void* buffer, UInt32*  
bufSize, UInt32 timeout);
```

Purpose

Writes data to the serial port. Serial communication parameters, such as baud rate, are set in the camera file associated with the session. You can adjust these communication parameters directly in the camera file.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
buffer	const void*	input
bufSize	UInt32*	input/output
timeout	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

buffer: data to send.

bufSize: on input, the number of bytes to send. On return, the number of bytes written.

timeout: time, in milliseconds, to wait for the data to be written.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionSerialRead

Usage

```
rval imgSessionSerialRead(SESSION_ID sid, void* buffer, uInt32* bufSize,  
uInt32 timeout);
```

Purpose

Reads in data from the serial port on devices that support serial communication. This function fills the buffer with characters received from the serial port until either a termination character has been received or the timeout period has elapsed. The termination character is defined in the camera file associated with the session.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
buffer	void*	input
bufSize	UInt32*	input/output
timeout	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

buffer: buffer to receive the data read from the serial port.

bufSize: on input, the size of the buffer. On output, the number of bytes read into the buffer.

timeout: time, in milliseconds, to wait for the data to be read.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionSerialReadBytes

Usage

```
rval imgSessionSerialReadBytes(SESSION_ID sid, void* buffer, uInt32*  
bufSize, uInt32 timeout);
```

Purpose

Reads in an expected number of bytes from the serial port on image acquisition devices that support serial communication. This function fills the buffer with characters received from the serial port until either the buffer is full or the timeout period has elapsed. When you use this function, the serial termination string attribute is ignored.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
buffer	void*	input
bufSize	UInt32*	input/output
timeout	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

buffer: buffer to receive the data read from the serial port.

bufSize: on input, this is the number of bytes you expect to read. The buffer must be large enough to hold this number of bytes. On output, this is the number of bytes read into the buffer.

timeout: time, in milliseconds, to wait for the data to be read.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSessionSerialFlush

Usage

```
rval imgSessionSerialFlush(SESSION_ID sid);
```

Purpose

Clears the internal serial buffer. In a serial write/read sequence, call `imgSessionSerialFlush` before calling [`imgSessionSerialWrite`](#) to clear the internal serial buffer for the next read.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

Attributes

Attributes are divided into groups. Click the links for a description, the possible values or range, and the data type of the properties within each group.

- [Analog](#)
- [Color](#)
- [Device Information](#)
- [Image](#)
- [Session Information](#)
- [Status Information](#)

Analog

Analog attributes allow you to set analog device parameters such as antichrominance filter and input range.

Readable	Writable	Devices
Indicates when the attribute is readable: <ul style="list-style-type: none">• Always—Attribute is readable both during acquisition and during configuration.• Running—Attribute is readable only during acquisition.• Configuration—Attribute is readable only during configuration.• NotReadable—Attribute is never readable.	Indicates when the attribute is writable: <ul style="list-style-type: none">• Always—Attribute is writable both during acquisition and during configuration.• Running—Attribute is writable only at run time.• Configuration—Attribute is writable only during configuration.• NotWritable—Attribute is never writable.	Indicates the devices that the attribute applies to.

IMG_ATTR_CHROMA_FILTER

Gets/sets the antichrominance filter used by the image acquisition device. This property is valid only on devices with an antichrominance filter.

Values

The following list includes possible values:

- **IMG_FILTER_NONE**—specifies no video filter
- **IMG_FILTER_NTSC**—specifies NTSC video filter
- **IMG_FILTER_PAL**—specifies PAL video filter

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI/PXI-1409• NI PCI-1410

IMG_ATTR_BLACK_REF_VOLT

Gets/sets the black reference level, in volts, of the channel associated with this session. Refer to the following tables for value ranges that are valid for each device.



Note The black reference level must be less than the white reference level.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device	Range (volts)
Double	Always	Configuration	NI 1405	0
Double	Always	Always	NI 1407 (PCI Rev A through D)	0 to 1.26
Double	Always	Always	NI 1407 (PCI Rev E or later)	0 to 0.5
Double	Always	Always	NI 1407 (PXI All Revs)	0 to 1.26
Double	Always	Configuration	NI 1409	0 to 1.4
Double	Always	Configuration	NI 1410	0 to 1.4
Double	Always	Configuration	NI 1411	0

IMG_ATTR_WHITE_REF_VOLT

Gets/sets the white reference level, in volts, of the channel associated with this session. Refer to the following tables for value ranges that are valid for each device.



Note The white reference level must be greater than the black reference level.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device	Range (volts)
Double	Always	Configuration	1405	0 to 1.26
Double	Always	Always	1407 (PCI Rev A through E or later)	0 to 1.26
Double	Always	Always	1407 (PXI All Revs)	0 to 1.26
Double	Always	Configuration	1409	0 to 1.4
Double	Always	Configuration	1410	0 to 1.4
Double	Always	Configuration	1411 (PXI and PCI Rev A)	0 to 0.08
Double	Always	Configuration	1411 (PXI and PCI Rev B or later)	0 to 1.26

Color

Color attributes set parameters associated with a color acquisition. You can only use the NI PCI-1405 and NI PCI/PXI-1411 for color acquisitions.

Readable	Writable	Devices
Indicates when the property is readable: <ul style="list-style-type: none">• Always—Property is readable both during acquisition and during configuration.• Running—Property is readable only during acquisition.• Configuration—Property is readable only during configuration.• NotReadable—Property is never readable.	Indicates when the property is writable: <ul style="list-style-type: none">• Always—Property is writable both during acquisition and during configuration.• Running—Property is writable only during acquisition.• Configuration—Property is writable only during configuration.• NotWritable—Property is never writable.	Indicates the devices that the property applies to.

IMG_ATTR_COLOR_CHROMA_BANDWIDTH

Specifies the resulting bandwidth of the chroma information of the image.

Values

The following list includes possible constant values:

- `IMG_COLOR_CHROMA_BANDWIDTH_HIGH`
- `IMG_COLOR_CHROMA_BANDWIDTH_LOW`

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_BRIGHTNESS

Adjusts the brightness of an image—the amount of white light added to or subtracted from each image pixel.

Values

The range is –50 to +50 IRE in steps of 1. IRE is the percentage of the white level. The default is 0 IRE.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
double	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_CHROMA_COMB

Selects the type of comb filter used in the chroma path.

Values

The following list includes possible constant values:

- `IMG_COLOR_COMB_OFF`—comb filter disabled (default in S-Video (Y/C) mode)
- `IMG_COLOR_COMB_1LINE`—comb filtering using one delayed line
- `IMG_COLOR_COMB_2LINES`—comb filtering using two delayed lines

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_CHROMA_PROCESS

Specifies the processing applied to the chroma signal.

Values

The following list includes possible constant values:

- `IMG_COLOR_CHROMA_PROCESS_ALWAYS_OFF`
- `IMG_COLOR_CHROMA_PROCESS_ALWAYS_ON`
- `IMG_COLOR_CHROMA_PROCESS_AUTODETECT`

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_CHROMA_TRAP

Enables the chroma trap filter in the luma signal path. Make sure this attribute is always disabled in S-Video (Y/C) mode.

Values

The following list includes possible constant values:

- FALSE—Chroma trap filter disabled (default in S-Video (Y/C) mode)
- TRUE—Chroma trap filter enabled, if the chroma trap filter is needed in composite mode

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_CONTRAST

Adjusts the contrast of the image. The value is a scaling factor applied to every pixel. The contrast adjustment is centered around the median pixel value.

Values

The range is 0.4 to 1.5. The default is 1.0

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_GAIN_BLUE

Gets/sets the gain applied to the blue color plane of the RGB image. This gain also affects the blue data used to calculate the hue, saturation, and luminance planes.

Values

The range is 0.8 to 1.198. The default value is 1.0.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
double	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_GAIN_GREEN

Gets/sets the gain applied to the green color plane of the RGB image. This gain also affects the green data used to calculate the hue, saturation, and luminance planes.

Values

The range is 0.8 to 1.198. The default value is 1.0.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
double	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_GAIN_RED

Gets/sets the gain applied to the red color plane of the RGB image. This gain also affects the red data used to calculate the hue, saturation, and luminance planes.

Values

The range is 0.8 to 1.198. The default value is 1.0.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
double	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_HSL_CORING_LEVEL

Gets/sets the HSL coring level when Image Representation is set to HSL. In HSL mode, if the saturation value (S) of any image pixel is lower than the specified value, the Hue value (H) of the pixel is set to the Hue Replace Value.

Values

The range is 0 to 255 LSB, with a default of 0 LSB.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	NI PCI/PXI-1411

IMG_ATTR_COLOR_HUE_OFFSET_ANGLE

Rotates the Hue plane with a specified offset angle. The hue value of a pixel is defined as an angle in the normal color plane. You can offset this angle to move the discontinuity point (at 0 modulo 360°) to another angle value.

Values

The range is -180° to +180°

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	NI PCI/PXI-1411

IMG_ATTR_COLOR_HUE_REPLACE_VALUE

Gets/sets the value used to replace the hue when it is below
IMG_ATTR_COLOR_HSL_CORING_LEVEL. You use this attribute only
when the Image Representation is set to HSL.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	NI PCI/PXI-1411

IMG_ATTR_COLOR_IMAGE REP

Specifies the type of image data returned when a color image is acquired.

Values

The following list includes possible constant values:

- **IMG_COLOR REP BLUE8**—Specifies 8-bit Blue color output.
- **IMG_COLOR REP GREEN8**—Specifies 8-bit Green color output.
- **IMG_COLOR REP HSI32**—Specifies a color image encoded in 32 bits—8 bits unused and 8 bits each for the Hue, Saturation, and Intensity planes.
- **IMG_COLOR REP HSL32**—Specifies a color image encoded in 32 bits—8 bits unused and 8 bits each for the Hue, Saturation, and Luminance planes.
- **IMG_COLOR REP HUE16**—Specifies 16-bit Hue color output.
- **IMG_COLOR REP HUE8**—Specifies 8-bit Hue color output.
- **IMG_COLOR REP INT16**—Specifies 16-bit Intensity color output
- **IMG_COLOR REP INT8**—Specifies 8-bit Intensity color output.
- **IMG_COLOR REP LUM16**—Specifies 16-bit Luminance color output.
- **IMG_COLOR REP LUM8**—Specifies 8-bit Luminance color output.
- **IMG_COLOR REP RED8**—Specifies 8-bit Red color output.
- **IMG_COLOR REP RGB16**—Specifies 16-bit RGB color output.
- **IMG_COLOR REP RGB24**—Specifies 24-bit RGB color output (default).
- **IMG_COLOR REP RGB32**—Specifies 32-bit RGB color output.
- **IMG_COLOR REP RGB48**—Specifies 48-bit RGB color output
- **IMG_COLOR REP SAT16**—Specifies 16-bit Saturation color output.
- **IMG_COLOR REP SAT8**—Specifies 8-bit Saturation color output.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_COLOR_LUMA_BANDWIDTH

Selects different bandwidths for the luminance signal.

Values

The following list includes possible constant values:

- **IMG_COLOR_LUMA_BANDWIDTH_FULL**—All filters, including decimation filter, are disabled. Default value in CCIR or RS-170 mode.
- **IMG_COLOR_LUMA_BANDWIDTH_HIGH**—Highest available bandwidth with decimation filter enabled. Default value for PAL or NTSC mode.
- **IMG_COLOR_LUMA_BANDWIDTH_LOW**—Decimation filter enabled, lowest bandwidth.
- **IMG_COLOR_LUMA_BANDWIDTH_MEDIUM**—Decimation filter enabled, medium bandwidth.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_LUMA_COMB

Selects the type of comb filter used in the luma path.

Values

The following list includes possible constant values:

- **IMG_COLOR_COMB_OFF**—Comb filter disabled (default in S-Video (Y/C) mode).
- **IMG_COLOR_COMB_1LINE**—Comb 1 LineComb filtering using 1 delayed line.
- **IMG_COLOR_COMB_2LINES**—Comb 2 LinesComb filtering using 2 delayed lines

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_PEAKING_ENABLE

Enables the peaking filter in the luma path.

Values

The following list includes possible constant values:

- FALSE—Peaking filter disabled (default)
- TRUE—Peaking filter enabled

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_RGB_CORING_LEVEL

Selects among four different coring levels. On any image pixel, if the color saturation of the pixel is lower than the specified value, the saturation is set to zero, which results in a monochrome pixel.

Values

The following list includes possible constant values:

- **IMG_COLOR_RGB_CORING_LEVEL_NOCORING**—Coring not activated.
- **IMG_COLOR_RGB_CORING_LEVEL_C1**—Coring activated for saturation equal or below 1 LSB.
- **IMG_COLOR_RGB_CORING_LEVEL_C3**—Coring activated for saturation equal or below 3 LSB.
- **IMG_COLOR_RGB_CORING_LEVEL_C7**—Coring activated for saturation equal or below 7 LSB.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_SATURATION

Adjusts the saturation of the image—a factor multiplied to the chroma information of the image.

Values

The range is 0.5 to 1.5. The default is 1.00.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
double	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

IMG_ATTR_COLOR_TINT

Gets/sets the tint of the image. Tint is specified in degrees and corresponds to the rotation of the UV color plane. The default value is 0.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
double	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1411

Device Information

Device information attributes return information concerning the image acquisition device.

Readable	Writable	Devices
Indicates when the property is readable: <ul style="list-style-type: none">• Always—Property is readable both during acquisition and during configuration.• Running—Property is readable only during acquisition.• Configuration—Property is readable only during configuration.• NotReadable—Property is never readable.	Indicates when the property is writable: <ul style="list-style-type: none">• Always—Property is writable both during acquisition and during configuration.• Running—Property is writable only during acquisition.• Configuration—Property is writable only during configuration.• NotWritable—Property is never writable.	Indicates the devices that the property applies to.

IMG_ATTR_CLOCK_FREQ

Returns the maximum pixel clock frequency of the device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_COLOR

Returns TRUE if the interface device supports color processing.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_GETSERIAL

Returns the serial number of the device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_HASRAM

Returns TRUE if the interface device has onboard memory.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_INTERFACE_TYPE

Returns the type of the interface in hex. For example, this attribute returns 0x1424 for the NI PCI-1424.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_LINESCAN

Returns TRUE if the camera attached to the interface is a line scan camera.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_NUM_EXT_LINES

Returns the number of External trigger lines available to the device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_NUM_ISO_IN_LINES

Returns the number of Iso In trigger lines available to the device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_NUM_ISO_OUT_LINES

Returns the number of Iso Out trigger lines available to the device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_NUM_RTSI_LINES

Returns the number of RTSI trigger lines available to the device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_RAMSIZE

Returns the size of the RAM on the interface device.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_NUM_PORTS

Returns the number of ports the device supports.



Note A port identifies a single independent data stream from a camera. All NI image acquisition devices support at least one port. Devices that support multiple ports can sustain independent and asynchronous acquisitions from the cameras on each port.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_CURRENT_PORT_NUM

Returns the current port number that the opened interface is accessing.



Note A port identifies a single independent data stream from a camera. All NI image acquisition devices support at least one port. Devices that support multiple ports can sustain independent and asynchronous acquisitions from the cameras on each port.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

Image

Image attributes define parameters that affect an image acquisition, such as region of interest.

Readable	Writable	Devices
Indicates when the property is readable: <ul style="list-style-type: none">• Always—Property is readable both during acquisition and during configuration.• Running—Property is readable only during acquisition.• Configuration—Property is readable only during configuration.• NotReadable—Property is never readable.	Indicates when the property is writable: <ul style="list-style-type: none">• Always—Property is writable both during acquisition and during configuration.• Running—Property is writable only during acquisition.• Configuration—Property is writable only during configuration.• NotWritable—Property is never writable.	Indicates the devices to which the property applies.

IMG_ATTR_ACQUIRE_FIELD

Gets/sets the field acquired when IMG_ATTR_FRAME_FIELD is set to FIELD_MODE. When you are using FRAME_MODE, this attribute is the first field that is acquired in time.

Values

The following list includes possible constant values:

- **IMG_ACQUIRE_ALL**—Acquires all fields.
- **IMG_ACQUIRE_EVEN**—Acquires only even fields.
- **IMG_ACQUIRE_ODD**—Acquires only odd fields.
- **IMG_ACQUIRE_ALTERNATING**—Acquires all fields and uses `IMG_ATTR_START_FIELD` to determine the first field to acquire.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_BIN_THRESHOLD_HIGH

The upper limit for the binary threshold LUT.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_BIN_THRESHOLD_LOW

The lower limit for the binary threshold LUT.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_BITSPERPIXEL

Returns the bits per pixel value of the camera associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_BYTESPERPIXEL

Returns the bytes per pixel value of the camera/channel associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_CHANNEL

Programs the current channel selected on the interface (0–3).

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI/PXI-1409• NI PCI-1410

IMG_ATTR_FRAMEWAIT_MSEC

Gets/sets the timeout value for a frame, in milliseconds.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_FRAME_FIELD

Gets/sets the current mode of the interlace (Frame or Field).

Values

The following list includes possible values:

- **IMG_FIELD_MODE**—Set this attribute to IMG_FIELD_MODE to acquire a single, non-interlaced field per image buffer.
- **IMG_FRAME_MODE**—Set this attribute to IMG_FRAME_MODE to acquire two interlaced fields per image buffer.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_HSCALE

Gets/sets the horizontal hardware scaling factor. The horizontal resolution of the acquired image data will be reduced by this factor. This attribute is deprecated and is not supported by all NI image acquisition devices.

Values

The following list includes possible constant values:

- **IMG_SCALE_NONE**—Performs no horizontal scaling.
- **IMG_SCALE_DIV2**—The horizontal resolution is scaled down by a factor of 2.
- **IMG_SCALE_DIV4**—The horizontal resolution is scaled down by a factor of 4.
- **IMG_SCALE_DIV8**—The horizontal resolution is scaled down by a factor of 8.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCI/PXI-1428

IMG_ATTR_INVERT

Sets/gets the invert image mode. If this property is set to invert, the image will be upside down in memory.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_LUT

Programs the look up table (LUT) for the given interface. Pass a constant to indicate the LUT you want to use.

Values

The following list includes possible constant values:

- `IMG_LUT_NORMAL`
- `IMG_LUT_INVERSE`
- `IMG_LUT_LOG`
- `IMG_LUT_INVERSE_LOG`
- `IMG_LUT_BINARY`
- `IMG_LUT_INVERSE_BINARY`



Note `IMG_LUT_NORMAL`, `IMG_LUT_BINARY`, and `IMG_LUT_INVERSE_BINARY` are the only valid constant values for the NI PCI-1426, NI PCIe-1427, NI PCIe-1429, and NI PCIe-1430.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ROI_HEIGHT

Gets/sets the region of interest (ROI) height of the camera/channel associated with this session.



Note This attribute requires an aligned ROI. Use [imgSessionFitROI](#) to calculate an aligned ROI.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ROI_LEFT

Gets/sets the region of interest left of the camera/channel associated with this session.



Note This attribute requires an aligned ROI. Use [imgSessionFitROI](#) to calculate an aligned ROI.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ROI_TOP

Gets/sets the region of interest top of the camera/channel associated with this session.



Note This attribute requires an aligned ROI. Use [imgSessionFitROI](#) to calculate an aligned ROI.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ROI_WIDTH

Gets/sets the region of interest left of the camera/channel associated with this session.



Note This attribute requires an aligned ROI. Use [imgSessionFitROI](#) to calculate an aligned ROI.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ROWPIXELS

Gets/sets the number of pixels in a row of an image. This attribute value may be larger than the width of the image. For example, if you are acquiring an 640x480 image into a 1280x960 buffer, and you want to place the image at the 640x480 position in the buffer, you must set **IMG_ATTR_ROWPIXELS** to 1280 so that the acquisition moves 1280 pixels from the end of first row of the acquired image before it copies the next row of pixels into the buffer.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_START_FIELD

For interlaced acquisitions, this attribute specifies the field that occupies line 0 of the image buffer.

Values

The following list includes possible constant values:

- `IMG_FIELD_ODD`
- `IMG_FIELD_EVEN`

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_VSCALE

Gets/sets the vertical hardware scaling factor. The vertical resolution of the acquired image data will be reduced by this factor. This attribute is deprecated and is not supported by all NI image acquisition devices.

Values

The following list includes possible constant values:

- **IMG_SCALE_NONE**—Performs no vertical scaling.
- **IMG_SCALE_DIV2**—The vertical resolution is scaled down by a factor of 2.
- **IMG_SCALE_DIV4**—The vertical resolution is scaled down by a factor of 4.
- **IMG_SCALE_DIV8**—The vertical resolution is scaled down by a factor of 8.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCI/PXI-1428

Session Information

Session information attributes set information about the maximum possible image size for an acquisition.

Readable	Writable	Devices
Indicates when the property is readable: <ul style="list-style-type: none">• Always—Property is readable both during acquisition and during configuration.• Running—Property is readable only during acquisition.• Configuration—Property is readable only during configuration.• NotReadable—Property is never readable.	Indicates when the property is writable: <ul style="list-style-type: none">• Always—Property is writable both during acquisition and during configuration.• Running—Property is writable only during acquisition.• Configuration—Property is writable only during configuration.• NotWritable—Property is never writable.	Indicates the devices to which the property applies.

IMG_ATTR_ACQWINDOW_HEIGHT

Gets/sets the acquisition window height of the camera/channel associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ACQWINDOW_LEFT

Gets/sets the acquisition window left offset of the camera/channel associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ACQWINDOW_TOP

Gets/sets the acquisition window top offset of the camera/channel associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ACQWINDOW_WIDTH

Gets/sets the acquisition window width of the camera/channel associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ENCODER_DIVIDE_FACTOR

Gets/sets the divide factor to use to derive the scaled encoder signal.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Always	<ul style="list-style-type: none">• NI PCI-1426• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ENCODER_FILTER

Enables/disables the noise filter for encoder phase A and phase B signals.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Always	<ul style="list-style-type: none">• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_ENCODER_PHASE_A_POLARITY

Gets/sets the encoder phase A signal polarity.

Values

The following list includes possible constant values:

- **IMG_TRIG_POLAR_ACTIVEH**—Sets the phase A signal polarity to active high.
- **IMG_TRIG_POLAR_ACTIVEL**—Sets the phase A signal polarity to active low.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Always	<ul style="list-style-type: none">• NI PCI-1426• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430



Note The NI 1426 only supports active high.

IMG_ATTR_ENCODER_PHASE_B_POLARITY

Gets/sets the encoder phase B signal polarity.

Values

The following list includes possible constant values:

- **IMG_TRIG_POLAR_ACTIVEH**—Sets the phase B signal polarity to active high.
- **IMG_TRIG_POLAR_ACTIVEL**—Sets the phase B signal polarity to active low.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Always	<ul style="list-style-type: none">• NI PCI-1426• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430



Note The NI 1426 only supports active high.

IMG_ATTR_ENCODER_POSITION

Returns the current absolute encoder position.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt64	Always	Never	<ul style="list-style-type: none">• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430



Note The NI PCI-1426 encoder hardware does not support reading the absolute encoder position.

IMG_ATTR_EXT_TRIG_LINE_FILTER

Enables/disables the noise filter for external trigger lines.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Always	<ul style="list-style-type: none">• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_NUM_POST_TRIGGER_BUFFERS

Gets/sets the number of buffers to be acquired after the assertion edge of the stop trigger. This attribute is only used when a stop trigger has been registered for the acquisition.



Note Use [imgSessionTriggerConfigure2](#) to register a stop trigger. Choose IMG_TRIG_ACTION_STOP as the trigger action.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_RTSI_LINE_FILTER

Enables/disables the noise filter for RTSI trigger lines.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Always	<ul style="list-style-type: none">• NI PCIe-1427• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_VHA_MODE

Enables Variable Height Acquisition (VHA) Mode.

Values

The following list includes possible constant values:

- **FALSE**—VHA Mode disabled
- **TRUE**—VHA Mode enabled

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	Configuration	<ul style="list-style-type: none">• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

Status Information

Status information attributes return status information about an acquisition.

Readable	Writable	Devices
Indicates when the property is readable: <ul style="list-style-type: none">• Always—Property is readable both during acquisition and during configuration.• Running—Property is readable only during acquisition.• Configuration—Property is readable only during configuration.• NotReadable—Property is never readable.	Indicates when the property is writable: <ul style="list-style-type: none">• Always—Property is writable both during acquisition and during configuration.• Running—Property is writable only during acquisition.• Configuration—Property is writable only during configuration.• NotWritable—Property is never writable.	Indicates the devices that the property applies to.

IMG_ATTR_ACQ_IN_PROGRESS

Returns TRUE if an acquisition is in progress on the camera associated with this session.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_FRAME_COUNT

Returns the number of frames acquired since the start of an acquisition.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_LAST_VALID_BUFFER

Returns the last available buffer list index.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_LAST_VALID_FRAME

Returns the last available cumulative buffer number.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Running	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

IMG_ATTR_LOST_FRAMES

Returns the total number of lost frames in a continuous acquisition. The value increments one time for each lost frame.

Remarks

The following table shows when the property is readable and writable and what devices it applies to.

Datatype	Readable	Writable	Device
UInt32	Always	NotWritable	<ul style="list-style-type: none">• NI PCI-1405• NI PCI/PXI-1407• NI PCI/PXI-1409• NI PCI-1410• NI PCI/PXI-1411• NI PCI/PXI-1422• NI PCI-1424• NI PCI-1426• NI PCIe-1427• NI PCI/PXI-1428• NI PCIe-1429• NI PCIe-1430

Reference

[Constants](#)

[Error Codes](#)

[Quadrature Encoder Overview](#)

[Scaled Encoder Signal](#)

[Variable Data Types](#)

[Primary Data Types](#)

[Programming Language Considerations](#)

[LabVIEW](#)

[LabWindows/CVI](#)

[Other Programming Environments](#)

[Code Examples](#)

Constants

Constants help clearly define specific function parameter values. These constants are included in the niimaq.h header file.

The following table lists the constant name, the function to which the constant applies, and a general description.

Constant	Use With
IMG_ACQUIRE_ALL	imgGetAttribute imgSetAttribute
IMG_ACQUIRE_EVEN	imgGetAttribute imgSetAttribute
IMG_ACQUIRE_ODD	imgGetAttribute imgSetAttribute
IMG_AQ_DONE	imgSessionWaitSignal2 imgSessionWaitSignalAsyn imgPulseCreate2
IMG_AQ_IN_PROGRESS	imgSessionWaitSignal2 imgSessionWaitSignalAsyn imgPulseCreate2
IMG_BAYER_PATTERN_GBGB_RGRG	imgBayerColorDecode
IMG_BAYER_PATTERN_GRGR_BGBG	imgBayerColorDecode
IMG_BAYER_PATTERN_BGBG_GRGR	imgBayerColorDecode
IMG_BAYER_PATTERN_RGRG_GBGB	imgBayerColorDecode

IMG_BUF_COMPLETE	imgSessionWaitSignal2 imgSessionWaitSignalAsy imgPulseCreate2
IMG_BUFF_ACTUAL_HEIGHT	imgGetBufferElement
IMG_BUFF_ADDRESS	imgGetBufferElement imgSetBufferElement
IMG_BUFF_CHANNEL	imgGetBufferElement imgSetBufferElement
IMG_BUFF_COMMAND	imgGetBufferElement imgSetBufferElement
IMG_BUFF_SIZE	imgGetBufferElement imgSetBufferElement
IMG_BUFF_SKIPCOUNT	imgGetBufferElement imgSetBufferElement
IMG_CMD_LOOP	imgGetBufferElement imgSetBufferElement (with IMG_BUFF_COMMA constant)
IMG_CMD_NEXT	imgGetBufferElement imgSetBufferElement (with IMG_BUFF_COMMA

constant)

IMG_CMD_STOP

[imgGetBufferElement](#)
[imgSetBufferElement](#)
(with IMG_BUFF_COMMA
constant)

IMG_COLOR_CHROMA_
BANDWIDTH_HIGH

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_CHROMA_
BANDWIDTH_LOW

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_CHROMA_
PROCESS_ALWAYS_OFF

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_CHROMA_
PROCESS_ALWAYS_ON

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_CHROMA_
PROCESS_AUTODETECT

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_COMB_1LINE

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_COMB_2LINES

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_COMB_OFF

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_LUMA_
BANDWIDTH_FULL

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_LUMA_
BANDWIDTH_HIGH

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_LUMA_
BANDWIDTH_LOW

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR_LUMA_
BANDWIDTH_MEDIUM

[imgSetAttribute](#)
[imgGetAttribute](#)

IMG_COLOR REP_BLUE8

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_GREEN8

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_HSI32

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_HSL32

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_HUE16

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_HUE8

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_INT16

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_INT8

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_COLOR REP_LUM16

[imgGetAttribute](#)

	imgSetAttribute
IMG_COLOR REP LUM8	imgGetAttribute imgSetAttribute
IMG_COLOR REP RED8	imgGetAttribute imgSetAttribute
IMG_COLOR REP RGB16	imgGetAttribute imgSetAttribute
IMG_COLOR REP RGB24	imgGetAttribute imgSetAttribute
IMG_COLOR REP RGB32	imgGetAttribute imgSetAttribute
IMG_COLOR REP RGB48	imgGetAttribute imgSetAttribute
IMG_COLOR REP SAT16	imgGetAttribute imgSetAttribute
IMG_COLOR REP SAT8	imgGetAttribute imgSetAttribute
IMG_COLOR_RGB_CORING_ LEVEL_NOCORING	imgSetAttribute imgGetAttribute
IMG_COLOR_RGB_CORING_ LEVEL_C1	imgSetAttribute imgGetAttribute
IMG_COLOR_RGB_CORING_ LEVEL_C3	imgSetAttribute imgGetAttribute
IMG_COLOR_RGB_CORING_ LEVEL_C7	imgSetAttribute imgGetAttribute
IMG_CURRENT_BUFFER	imgSessionExamineBuffer
IMG_DEVICE_FRAME	imgCreateBuffer

IMG_FIELD_EVEN	imgGetAttribute imgSetAttribute
IMG_FIELD_MODE	imgGetAttribute imgSetAttribute
IMG_FIELD_ODD	imgGetAttribute imgSetAttribute
IMG_FILTER_NONE	imgGetAttribute imgSetAttribute
IMG_FILTER_NTSC	imgGetAttribute imgSetAttribute
IMG_FILTER_PAL	imgGetAttribute imgSetAttribute
IMG_FRAME_DONE	imgSessionWaitSignal2 imgSessionWaitSignalAsy imgPulseCreate2
IMG_FRAME_MODE	imgGetAttribute imgSetAttribute
IMG_FRAME_START	imgSessionWaitSignal2 imgSessionWaitSignalAsy imgPulseCreate2
IMG_FRAME_VALID	imgPulseCreate2
IMG_FRAMETIME_10MINUTES	imgGetAttribute imgSetAttribute
IMG_FRAMETIME_10SECONDS	imgGetAttribute imgSetAttribute
IMG_FRAMETIME_1MINUTE	imgGetAttribute imgSetAttribute
IMG_FRAMETIME_1SECOND	imgGetAttribute

IMG_FRAMETIME_2MINUTES	imgSetAttribute imgGetAttribute imgSetAttribute
IMG_FRAMETIME_2SECONDS	imgGetAttribute imgSetAttribute
IMG_FRAMETIME_5MINUTES	imgGetAttribute imgSetAttribute
IMG_FRAMETIME_5SECONDS	imgGetAttribute imgSetAttribute
IMG_FRAMETIME_STANDARD	imgGetAttribute imgSetAttribute
IMG_GAIN_0DB	imgGetAttribute imgSetAttribute
IMG_GAIN_3DB	imgGetAttribute imgSetAttribute
IMG_GAIN_6DB	imgGetAttribute imgSetAttribute
IMG_HOST_FRAME	imgCreateBuffer
IMG_IMMEDIATE	imgPulseCreate2
IMG_LAST_BUFFER	imgSessionExamineBuffer
IMG_LINE_VALID	imgPulseCreate2
IMG_LUT_BINARY	imgGetAttribute imgSetAttribute
IMG_LUT_INVERSE	imgGetAttribute imgSetAttribute

IMG_LUT_INVERSE_BINARY

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_LUT_INVERSE_LOG

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_LUT_LOG

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_LUT_NORMAL

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_LUT_TYPE_DEFAULT

[imgSessionSetUserLUT8b](#)
[imgSessionSetUserLUT16](#)

IMG_LUT_TYPE_RED

[imgSessionSetUserLUT8b](#)

IMG_LUT_TYPE_GREEN

[imgSessionSetUserLUT8b](#)

IMG_LUT_TYPE_BLUE

[imgSessionSetUserLUT8b](#)

IMG_LUT_TYPE_TAP0

[imgSessionSetUserLUT8b](#)
[imgSessionSetUserLUT16](#)

IMG_LUT_TYPE_TAP1

[imgSessionSetUserLUT8b](#)
[imgSessionSetUserLUT16](#)

IMG_LUT_TYPE_TAP2

[imgSessionSetUserLUT8b](#)

IMG_LUT_TYPE_TAP3	<u>imgSessionSetUserLUT8b</u>
IMG_OVERWRITE_GET_OLDEST	<u>imgSessionCopyBufferByNl</u> <u>imgSessionCopyAreaByNl</u>
IMG_OVERWRITE_GET_NEXT_ITERATION	<u>imgSessionCopyBufferByNl</u> <u>imgSessionCopyAreaByNl</u>
IMG_OVERWRITE_FAIL	<u>imgSessionCopyBufferByNl</u> <u>imgSessionCopyAreaByNl</u>
IMG_OVERWRITE_GET_NEWEST	<u>imgSessionCopyBufferByNl</u> <u>imgSessionCopyAreaByNl</u>
IMG_PULSE_POLAR_ACTIVEH	<u>imgPulseCreate</u>
IMG_PULSE_POLAR_ACTIVEV	<u>imgPulseCreate</u>
IMG_ROI_FIT_LARGER	<u>imgSessionFitROI</u>
IMG_ROI_FIT_SMALLER	<u>imgSessionFitROI</u>
IMG_SCALE_DIV2	<u>imgGetAttribute</u> <u>imgSetAttribute</u>

IMG_SCALE_DIV4

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_SCALE_DIV8

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_SCALE_NONE

[imgGetAttribute](#)
[imgSetAttribute](#)

IMG_SIGNAL_EXTERNAL

[imgPulseCreate2](#)
[imgSessionLineTrigSource](#)
[imgSessionTriggerConfigu](#)
[imgSessionTriggerDrive2](#)
[imgSessionTriggerRead2](#)
[imgSessionTriggerRoute2](#)
[imgSessionWaitSignal2](#)
[imgSessionWaitSignalAsy](#)

IMG_SIGNAL_ISO_IN

[imgPulseCreate2](#)
[imgSessionLineTrigSource](#)
[imgSessionTriggerConfigu](#)
[imgSessionTriggerRead2](#)
[imgSessionTriggerRoute2](#)
[imgSessionWaitSignal2](#)
[imgSessionWaitSignalAsy](#)

IMG_SIGNAL_ISO_OUT

[imgPulseCreate2](#)
[imgSessionLineTrigSource](#)
[imgSessionTriggerConfigu](#)
[imgSessionTriggerDrive2](#)
[imgSessionTriggerRead2](#)
[imgSessionTriggerRoute2](#)
[imgSessionWaitSignal2](#)
[imgSessionWaitSignalAsy](#)

IMG_SIGNAL_NONE

[imgSessionTriggerRoute2](#)

IMG_SIGNAL_RTSI

[imgPulseCreate2](#)
[imgSessionLineTrigSource](#)
[imgSessionTriggerConfigu](#)

IMG_SIGNAL_SCALED_ENCODER	imgSessionTriggerDrive2 imgSessionTriggerRead2 imgSessionTriggerRoute2 imgSessionWaitSignal2 imgSessionWaitSignalAsy imgSessionLineTrigSource
IMG_SIGNAL_STATUS	imgPulseCreate2 imgSessionWaitSignal2 imgSessionWaitSignalAsy
IMG_TRIG_ACTION_BUFFER	imgSessionTriggerConfigu
IMG_TRIG_ACTION_BUFLIST	imgSessionTriggerConfigu
IMG_TRIG_ACTION_CAPTURE	imgSessionTriggerConfigu
IMG_TRIG_ACTION_NONE	imgSessionTriggerConfigu
IMG_TRIG_ACTION_STOP	imgSessionTriggerConfigu
IMG_TRIG_DRIVE_AQ_DONE	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_AQ_IN_PROGRESS	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_ASSERTED	imgSessionTriggerDrive2

IMG_TRIG_DRIVE_DISABLED	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_FRAME_DONE	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_FRAME_START	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_HSYNC	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_PIXEL_CLK	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_SCALED_ENCODER	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_UNASSERTED	imgSessionTriggerDrive2
IMG_TRIG_DRIVE_VSYNC	imgSessionTriggerDrive2
IMG_TRIG_POLAR_ACTIVEH	imgSessionTriggerConfigu imgSessionTriggerDrive2 imgSessionTriggerRead2
IMG_TRIG_POLAR_ACTIVEV	imgSessionTriggerConfigu imgSessionTriggerDrive2 imgSessionTriggerRead2
IMGPLOT_COLOR_RGB24	imgPlot
IMGPLOT_COLOR_RGB32	imgPlot
IMGPLOT_INVERT	imgPlot

IMGPLOT_MONO_10	imgPlot
IMGPLOT_MONO_12	imgPlot
IMGPLOT_MONO_14	imgPlot
IMGPLOT_MONO_16	imgPlot
IMGPLOT_MONO_32	imgPlot
IMGPLOT_MONO_8	imgPlot
INTERFACE_NAME_SIZE	imgInterfaceQueryNames
PULSE_MODE_SINGLE	imgPulseCreate2
PULSE_MODE_SINGLE_REARM	imgPulseCreate2
PULSE_MODE_TRAIN	imgPulseCreate2
PULSE_TIMEBASE_100KHZ	imgPulseCreate2
PULSE_TIMEBASE_50MHZ	imgPulseCreate2

PULSE_TIMEBASE_PIXELCLK

[imgPulseCreate2](#)

PULSE_TIMEBASE_SCALED_ENCODER

[imgPulseCreate2](#)

Error Codes

Every NI-IMAQ function is of the following form:

```
rval = Function_Name(parameter 1, parameter 2, ... parameter n);
```

Each function returns a status code (rval) that indicates the success or failure of the function. The following table describes the status codes returned by each NI-IMAQ function.

Error Code	Status Name
-1074397183	IMG_ERR_NCAP
-1074397182	IMG_ERR_OVRN
-1074397181	IMG_ERR_EMEM
-1074397180	IMG_ERR_OSER
-1074397179	IMG_ERR_PAR1
-1074397178	IMG_ERR_PAR2
-1074397177	IMG_ERR_PAR3
-1074397176	IMG_ERR_PAR4
-1074397175	IMG_ERR_PAR5
-1074397174	IMG_ERR_PAR6
-1074397173	IMG_ERR_PAR7
-1074397172	IMG_ERR_MXBF
-1074397171	IMG_ERR_DLLE

-1074397170	IMG_ERR_BSIZ
-1074397169	IMG_ERR_MXBI
-1074397168	IMG_ERR_ELCK
-1074397167	IMG_ERR_DISE
-1074397166	IMG_ERR_BBUF
-1074397165	IMG_ERR_NLCK
-1074397164	IMG_ERR_NCAM
-1074397163	IMG_ERR_BINT
-1074397162	IMG_ERR_BROW
-1074397161	IMG_ERR_BROI
-1074397160	IMG_ERR_BCMF
-1074397159	IMG_ERR_NVBL
-1074397158	IMG_ERR_NCFG
-1074397157	IMG_ERR_BBLF
-1074397156	IMG_ERR_BBLE
-1074397155	IMG_ERR_BBLB
-1074397154	IMG_ERR_NAIP

-1074397153	IMG_ERR_VLCK
-1074397152	IMG_ERR_BDMA
-1074397151	IMG_ERR_AIOP
-1074397150	IMG_ERR_TIMO
-1074397149	IMG_ERR_NBUF
-1074397148	IMG_ERR_ZBUF
-1074397147	IMG_ERR_HLPR
-1074397146	IMG_ERR_BTRG
-1074397145	IMG_ERR_NINF
-1074397144	IMG_ERR_NDLL
-1074397143	IMG_ERR_NFNC
-1074397142	IMG_ERR_NOSR
-1074397141	IMG_ERR_BTAC
-1074397140	IMG_ERR_FIFO
-1074397139	IMG_ERR_MLCK

-1074397138	IMG_ERR_ILCK
-1074397137	IMG_ERR_NEPK
-1074397136	IMG_ERR_SCLM
-1074397135	IMG_ERR_SCC1
-1074397134	IMG_ERR_SMALLALLOC
-1074397133	IMG_ERR_ALLOC
-1074397132	IMG_ERR_BADCAMTYPE
-1074397131	IMG_ERR_BADPIXTYPE
-1074397130	IMG_ERR_BADCAMPARAM
-1074397129	IMG_ERR_PALKEYDTCT
-1074397128	IMG_ERR_BFRQ
-1074397127	IMG_ERR_BITP
-1074397126	IMG_ERR_HWNC
-1074397125	IMG_ERR_SERIAL
-1074397124	IMG_ERR_MXPI
-1074397123	IMG_ERR_BPID
-1074397122	IMG_ERR_NEVR
-1074397121	IMG_ERR_SERIAL_TIMO

-1074397120	IMG_ERR_PG_TOO_MANY
-1074397119	IMG_ERR_PG_BAD_TRANS
-1074397118	IMG_ERR_PLNS
-1074397117	IMG_ERR_BPMD
-1074397116	IMG_ERR_NSAT
-1074397115	IMG_ERR_HYBRID
-1074397114	IMG_ERR_BADFILFMT
-1074397113	IMG_ERR_BADFILEXT
-1074397112	IMG_ERR_NRTSI
-1074397111	IMG_ERR_MXTRG
-1074397110	IMG_ERR_MXRC
-1074397109	IMG_ERR_OOR
-1074397108	IMG_ERR_NPROG
-1074397107	IMG_ERR_NEOM
-1074397106	IMG_ERR_BDTYPE
-1074397105	IMG_ERR_THRDACCDEN
-1074397104	IMG_ERR_BADFILWRT
-1074397103	IMG_ERR_AEXM
-1074397102	IMG_ERR_BAD_LUT_TYPE
-1074397101	IMG_ERR_ATTRIBUTE_NOT_READABLE

-1074397100	IMG_ERR_BOARD_NOT_SUPPORTED
-1074397099	IMG_ERR_BAD_FRAME_FIELD
-1074397098	IMG_ERR_INVALID_ATTRIBUTE
-1074397097	IMG_ERR_BAD_LINE_MAP
-1074397095	IMG_ERR_BAD_CHANNEL
-1074397094	IMG_ERR_BAD_CHROMA_FILTER
-1074397093	IMG_ERR_BAD_SCALE
-1074397091	IMG_ERR_BAD_TRIGGER_MODE
-1074397090	IMG_ERR_BAD_CLAMP_START
-1074397089	IMG_ERR_BAD_CLAMP_STOP
-1074397088	IMG_ERR_BAD_BRIGHTNESS
-1074397087	IMG_ERR_BAD_CONTRAST
-1074397086	IMG_ERR_BAD_SATURATION
-1074397085	IMG_ERR_BAD_TINT
-1074397084	IMG_ERR_BAD_HUE_OFF_ANGLE

-1074397083	IMG_ERR_BAD_ACQUIRE_FIELD
-1074397082	IMG_ERR_BAD_LUMA_BANDWIDTH
-1074397081	IMG_ERR_BAD_LUMA_COMB
-1074397080	IMG_ERR_BAD_CHROMA_PROCESS
-1074397079	IMG_ERR_BAD_CHROMA_BANDWIDTH
-1074397078	IMG_ERR_BAD_CHROMA_COMB
-1074397077	IMG_ERR_BAD_RGB_CORING
-1074397076	IMG_ERR_BAD_HUE_REPLACE_VALUE
-1074397075	IMG_ERR_BAD_RED_GAIN
-1074397074	IMG_ERR_BAD_GREEN_GAIN
-1074397073	IMG_ERR_BAD_BLUE_GAIN
-1074397072	IMG_ERR_BAD_START_FIELD
-1074397071	IMG_ERR_BAD_TAP_DIRECTION
-1074397070	IMG_ERR_BAD_MAX_IMAGE_RECT
-1074397069	IMG_ERR_BAD_TAP_TYPE
-1074397068	IMG_ERR_BAD_SYNC_RECT
-1074397067	IMG_ERR_BAD_ACQWINDOW_RECT

-1074397066	IMG_ERR_BAD_HSL_CORING
-1074397065	IMG_ERR_BAD_TAP_0_VALID_RECT
-1074397064	IMG_ERR_BAD_TAP_1_VALID_RECT
-1074397063	IMG_ERR_BAD_TAP_2_VALID_RECT
-1074397062	IMG_ERR_BAD_TAP_3_VALID_RECT
-1074397061	IMG_ERR_BAD_TAP_RECT
-1074397060	IMG_ERR_BAD_NUM_TAPS
-1074397059	IMG_ERR_BAD_TAP_NUM
-1074397058	IMG_ERR_BAD_QUAD_NUM
-1074397057	IMG_ERR_BAD_NUM_DATA_LINES
-1074397056	IMG_ERR_BAD_BITS_PER_COMPONENT
-1074397055	IMG_ERR_BAD_NUM_COMPONENTS
-1074397054	IMG_ERR_BAD_BIN_THRESHOLD_LOW
-1074397053	IMG_ERR_BAD_BIN_THRESHOLD_HIGH
-1074397052	IMG_ERR_BAD_BLACK_REF_VOLT
-1074397051	IMG_ERR_BAD_WHITE_REF_VOLT

-1074397050	IMG_ERR_BAD_FREQ_STD
-1074397049	IMG_ERR_BAD_HDELAY
-1074397048	IMG_ERR_BAD_LOCK_SPEED
-1074397047	IMG_ERR_BAD_BUFFER_LIST
-1074397046	IMG_ERR_BOARD_NOT_INITIALIZED
-1074397045	IMG_ERR_BAD_PCLK_SOURCE
-1074397044	IMG_ERR_BAD_VIDEO_LOCK_CHANNEL
-1074397043	IMG_ERR_BAD_LOCK_SEL
-1074397042	IMG_ERR_BAD_BAUD_RATE
-1074397041	IMG_ERR_BAD_STOP_BITS
-1074397040	IMG_ERR_BAD_DATA_BITS
-1074397039	IMG_ERR_BAD_PARITY
-1074397038	IMG_ERR_TERM_STRING_NOT_FOUND
-1074397037	IMG_ERR_SERIAL_READ_TIMEOUT

-1074397036	IMG_ERR_SERIAL_WRITE_TIMEOUT
-1074397035	IMG_ERR_BAD_SYNCHRONICITY
-1074397034	IMG_ERR_BAD_INTERLACING_CONFIG
-1074397032	IMG_ERR_BAD_CHIP_CODE
-1074397031	IMG_ERR_LUT_NOT_PRESENT
-1074397030	IMG_ERR_DSPFILTER_NOT_PRESENT
-1074397029	IMG_ERR_DEVICE_NOT_FOUND
-1074397028	IMG_ERR_ONBOARD_MEM_CONFIG
-1074397027	IMG_ERR_BAD_POINTER
-1074397026	IMG_ERR_BAD_BUFFER_LIST_INDEX
-1074397025	IMG_ERR_INVALID_BUFFER_ATTRIBUTE
-1074397024	IMG_ERR_INVALID_BUFFER_PTR
-1074397023	IMG_ERR_BUFFER_LIST_ALREADY_LOCKED

-1074397022	IMG_ERR_BAD_DEVICE_TYPE
-1074397021	IMG_ERR_BAD_BAR_SIZE
-1074397019	IMG_ERR_NO_VALID_COUNTER_RECT
-1074397018	IMG_ERR_ACQ_STOPPED
-1074397017	IMG_ERR_BAD_TRIGGER_ACTION
-1074397016	IMG_ERR_BAD_TRIGGER_POLARITY
-1074397015	IMG_ERR_BAD_TRIGGER_NUMBER
-1074397014	IMG_ERR_BUFFER_NOT_AVAILABLE
-1074397012	IMG_ERR_BAD_PULSE_ID
-1074397011	IMG_ERR_BAD_PULSE_TIMEBASE
-1074397010	IMG_ERR_BAD_PULSE_GATE
-1074397009	IMG_ERR_BAD_PULSE_GATE_POLARITY
-1074397008	IMG_ERR_BAD_PULSE_OUTPUT
-1074397007	IMG_ERR_BAD_PULSE_OUTPUT_POLARITY
-1074397006	IMG_ERR_BAD_PULSE_MODE

-1074397005	IMG_ERR_NOT_ENOUGH_RESOURCES
-1074397004	IMG_ERR_INVALID_RESOURCE
-1074397003	IMG_ERR_BAD_FVAL_ENABLE
-1074397002	IMG_ERR_BAD_WRITE_ENABLE_MODE
-1074397001	IMG_ERR_COMPONENT_MISMATCH
-1074397000	IMG_ERR_FPGA_PROGRAMMING_FAILED
-1074396999	IMG_ERR_CONTROL_FPGA_FAILED
-1074396998	IMG_ERR_CHIP_NOT_READABLE
-1074396997	IMG_ERR_CHIP_NOT_WRITABLE
-1074396996	IMG_ERR_I2C_BUS_FAILED
-1074396995	IMG_ERR_DEVICE_IN_USE
-1074396994	IMG_ERR_BAD_TAP_DATALANES
-1074396993	IMG_ERR_BAD_VIDEO_GAIN
-1074396992	IMG_ERR_VHA_MODE_NOT_ALLOWED

-1074396991	IMG_ERR_BAD_TRACKING_SPEED
-1074396990	IMG_ERR_BAD_COLOR_INPUT_SELECT
-1074396989	IMG_ERR_BAD_HAV_OFFSET
-1074396988	IMG_ERR_BAD_HS1_OFFSET
-1074396987	IMG_ERR_BAD_HS2_OFFSET
-1074396986	IMG_ERR_BAD_IF_CHROMA
-1074396985	IMG_ERR_BAD_COLOR_OUTPUT_FORMAT
-1074396984	IMG_ERR_BAD_SAMSUNG_SCHCMP
-1074396983	IMG_ERR_BAD_SAMSUNG_CDLY
-1074396982	IMG_ERR_BAD_SECAM_DETECT
-1074396981	IMG_ERR_BAD_FSC_DETECT
-1074396980	IMG_ERR_BAD_SAMSUNG_CFTC
-1074396979	IMG_ERR_BAD_SAMSUNG_CGTC
-1074396978	IMG_ERR_BAD_SAMSUNG_SAMPLE_RATE
-1074396977	IMG_ERR_BAD_SAMSUNG_VSYNC_EDGE
-1074396976	IMG_ERR_SAMSUNG_LUMA_GAIN_CTRL

-1074396975	IMG_ERR_BAD_SET_COMB_COEF
-1074396974	IMG_ERR_SAMSUNG_CHROMA_TRACK
-1074396973	IMG_ERR_SAMSUNG_DROP_LINES
-1074396972	IMG_ERR_VHA_OPTIMIZATION_NOT_ALLOWED
-1074396971	IMG_ERR_BAD_PG_TRANSITION
-1074396970	IMG_ERR_TOO_MANY_PG_TRANSITIONS
-1074396969	IMG_ERR_BAD_CL_DATA_CONFIG
-1074396968	IMG_ERR_BAD_OCCURRENCE
-1074396967	IMG_ERR_BAD_PG_MODE
-1074396966	IMG_ERR_BAD_PG_SOURCE
-1074396965	IMG_ERR_BAD_PG_GATE
-1074396964	IMG_ERR_BAD_PG_GATE_POLARITY
-1074396963	IMG_ERR_BAD_PG_WAVEFORM_INITIAL_STATE
-1074396962	IMG_ERR_INVALID_CAMERA_ATTRIBUTE
-1074396961	IMG_ERR_BOARD_CLOSED

-1074396960	IMG_ERR_FILE_NOT_FOUND
-1074396959	IMG_ERR_BAD_1409_DSP_FILE
-1074396958	IMG_ERR_BAD_SCARABXCV200_32_FILE
-1074396957	IMG_ERR_BAD_SCARABXCV200_16_FILE
-1074396956	IMG_ERR_BAD_CAMERA_LINK_FILE
-1074396955	IMG_ERR_BAD_1411_CSC_FILE
-1074396954	IMG_ERR_BAD_ERROR_CODE
-1074396953	IMG_ERR_DRIVER_TOO_OLD
-1074396952	IMG_ERR_INSTALLATION_CORRUPT
-1074396951	IMG_ERR_NO_ONBOARD_MEMORY
-1074396950	IMG_ERR_BAD_BAYER_PATTERN
-1074396949	IMG_ERR_CANNOT_INITIALIZE_BOARD
-1074396948	IMG_ERR_CALIBRATION_DATA_CORRUPT
-1074396947	IMG_ERR_DRIVER_FAULT
-1074396946	IMG_ERR_ADDRESS_OUT_OF_RANGE

-1074396945	IMG_ERR_ONBOARD_ACQUISITION
-1074396944	IMG_ERR_NOT_AN_ONBOARD_ACQUISITION
-1074396943	IMG_ERR_BOARD_ALREADY_INITIALIZED
-1074396942	IMG_ERR_NO_SERIAL_PORT
-1074396941	IMG_ERR_BAD_VENABLE_GATING_MODE
-1074396940	IMG_ERR_BAD_1407_LUT_FILE
-1074396939	IMG_ERR_BAD_SYNC_DETECT_LEVEL
-1074396938	IMG_ERR_BAD_1405_GAIN_FILE
-1074396937	IMG_ERR_CLAMP_DAC_NOT_PRESENT
-1074396936	IMG_ERR_GAIN_DAC_NOT_PRESENT
-1074396935	IMG_ERR_REF_DAC_NOT_PRESENT
-1074396934	IMG_ERR_BAD_SCARABXC2S200_FILE
-1074396933	IMG_ERR_BAD_LUT_GAIN
-1074396932	IMG_ERR_BAD_MAX_BUF_LIST_ITER

-1074396931	IMG_ERR_BAD_PG_LINE_NUM
-1074396930	IMG_ERR_BAD_BITS_PER_PIXEL
-1074396929	IMG_ERR_TRIGGER_ALARM
-1074396928	IMG_ERR_BAD_SCARABXC2S200_03052009_FILE
-1074396927	IMG_ERR_LUT_CONFIG
-1074396926	IMG_ERR_CONTROL_FPGA_REQUIRES_NEWER_DRIV
-1074396925	IMG_ERR_CONTROL_FPGA_PROGRAMMING_FAILED
-1074396924	IMG_ERR_BAD_TRIGGER_SIGNAL_LEVEL
-1074396923	IMG_ERR_CAMERA_FILE_REQUIRES_NEWER_DRIVEF
-1074396922	IMG_ERR_DUPLICATED_BUFFER
-1074396921	IMG_ERR_NO_ERROR
-1074396920	IMG_ERR_INTERFACE_NOT_SUPPORTED

-1074396919	IMG_ERR_BAD_PCLK_POLARITY
-1074396918	IMG_ERR_BAD_ENABLE_POLARITY
-1074396917	IMG_ERR_BAD_PCLK_SIGNAL_LEVEL
-1074396916	IMG_ERR_BAD_ENABLE_SIGNAL_LEVEL
-1074396915	IMG_ERR_BAD_DATA_SIGNAL_LEVEL
-1074396914	IMG_ERR_BAD_CTRL_SIGNAL_LEVEL
-1074396913	IMG_ERR_BAD_WINDOW_HANDLE
-1074396912	IMG_ERR_CANNOT_WRITE_FILE
-1074396911	IMG_ERR_CANNOT_READ_FILE
-1074396910	IMG_ERR_BAD_SIGNAL_TYPE
-1074396909	IMG_ERR_BAD_SAMPLES_PER_LINE
-1074396908	IMG_ERR_BAD_SAMPLES_PER_LINE_REF
-1074396907	IMG_ERR_USE_EXTERNAL_HSYNC
-1074396906	IMG_ERR_BUFFER_NOT_ALIGNED
-1074396905	IMG_ERR_ROWPIXELS_TOO_SMALL

-1074396904	IMG_ERR_ROWPIXELS_NOT_ALIGNED
-1074396903	IMG_ERR_ROI_WIDTH_NOT_ALIGNED
-1074396902	IMG_ERR_LINESCAN_NOT_ALLOWED
-1074396901	IMG_ERR_INTERFACE_FILE_REQUIRES_NEWER_DRIV
-1074396900	IMG_ERR_BAD_SKIP_COUNT
-1074396899	IMG_ERR_BAD_NUM_X_ZONES
-1074396898	IMG_ERR_BAD_NUM_Y_ZONES
-1074396897	IMG_ERR_BAD_NUM_TAPS_PER_X_ZONE
-1074396896	IMG_ERR_BAD_NUM_TAPS_PER_Y_ZONE
-1074396895	IMG_ERR_BAD_TEST_IMAGE_TYPE
-1074396894	IMG_ERR_CANNOT_ACQUIRE_FROM_CAMERA
-1074396893	IMG_ERR_BAD_CTRL_LINE_SOURCE

-1074396892	IMG_ERR_BAD_PIXEL_EXTRACTOR
-1074396891	IMG_ERR_BAD_NUM_TIME_SLOTS
-1074396890	IMG_ERR_BAD_PLL_VCO_DIVIDER
-1074396889	IMG_ERR_CRITICAL_TEMP
-1074396888	IMG_ERR_BAD_DPA_OFFSET
-1074396887	IMG_ERR_BAD_NUM_POST_TRIGGER_BUFFERS
-1074396886	IMG_ERR_BAD_DVAL_MODE
-1074396885	IMG_ERR_BAD_TRIG_GEN_REARM_SOURCE
-1074396884	IMG_ERR_BAD_ASM_GATE_SOURCE
-1074396883	IMG_ERR_TOO_MANY_BUFFERS
-1074396882	IMG_ERR_BAD_TAP_4_VALID_RECT
-1074396881	IMG_ERR_BAD_TAP_5_VALID_RECT
-1074396880	IMG_ERR_BAD_TAP_6_VALID_RECT
-1074396879	IMG_ERR_BAD_TAP_7_VALID_RECT

-1074396878	IMG_ERR_FRONT_END_BANDWIDTH_EXCEEDED
-1074396877	IMG_ERR_BAD_PORT_NUMBER
-1074396876	IMG_ERR_PORT_CONFIG_CONFLICT
-1074396875	IMG_ERR_BITSTREAM_INCOMPATIBLE
-1074396874	IMG_ERR_SERIAL_PORT_IN_USE
-1074396873	IMG_ERR_BAD_ENCODER_DIVIDE_FACTOR
-1074396872	IMG_ERR_ENCODER_NOT_SUPPORTED
-1074396871	IMG_ERR_BAD_ENCODER_POLARITY
-1074396870	IMG_ERR_BAD_ENCODER_FILTER
-1074396869	IMG_ERR_ENCODER_POSITION_NOT_SUPPORTED
-1074396868	IMG_ERR_IMAGE_IN_USE

-1074396867	IMG_ERR_BAD_SCARABXL4000_FILE
-1074396866	IMG_ERR_BAD_CAMERA_ATTRIBUTE_VALUE
-1074396865	IMG_ERR_BAD_PULSE_WIDTH
-1074396864	IMG_ERR_FPGA_FILE_NOT_FOUND
-1074396863	IMG_ERR_FPGA_FILE_CORRUPT
-1074396862	IMG_ERR_BAD_PULSE_DELAY
-1074396861	IMG_ERR_BAD_PG_IDLE_SIGNAL_LEVEL
-1074396860	IMG_ERR_BAD_PG_WAVEFORM_IDLE_STATE
-1074396859	IMG_ERR_64_BIT_MEMORY_NOT_SUPPORTED
-1074396858	IMG_ERR_64_BIT_MEMORY_UPDATE_AVAILABLE

-1074396857	IMG_ERR_32_BIT_MEMORY_LIMITATION
-1074396856	IMG_ERR_KERNEL_NOT_LOADED
-1074396855	IMG_ERR_BAD_SENSOR_SHUTTER_PERIOD
-1074396854	IMG_ERR_BAD_SENSOR_CCD_TYPE
-1074396853	IMG_ERR_BAD_SENSOR_PARTIAL_SCAN
-1074396852	IMG_ERR_BAD_SENSOR_BINNING
-1074396851	IMG_ERR_BAD_SENSOR_GAIN
-1074396850	IMG_ERR_BAD_SENSOR_BRIGHTNESS
-1074396849	IMG_ERR_BAD_LED_STATE
-1074396848	IMG_ERR_64_BIT_NOT_SUPPORTED
-1074396847	IMG_ERR_BAD_TRIGGER_DELAY

-1074396846	IMG_ERR_LIGHTING_CURRENT_EXCEEDS_LIMITS
-1074396845	IMG_ERR_LIGHTING_INVALID_MODE
-1074396844	IMG_ERR_LIGHTING_EXTERNAL_INVALID_MODE
-1074396843	IMG_ERR_BAD_SENSOR_EXPOSURE
-1074396842	IMG_ERR_BAD_FRAME_RATE
-1074396841	IMG_ERR_BAD_SENSOR_PARTIAL_SCAN_BINNING_C
-1074396840	IMG_ERR_SOFTWARE_TRIGGER_NOT_CONFIGURED
-1074396839	IMG_ERR_FREE_RUN_MODE_NOT_ALLOWED
-1074396838	IMG_ERR_BAD_LIGHTING_RAMPUP
-1074396837	IMG_ERR_AFE_CONFIG_TIMEOUT
-1074396836	IMG_ERR_LIGHTING_ARM_TIMEOUT
-1074396835	IMG_ERR_LIGHTING_SHORT_CIRCUIT

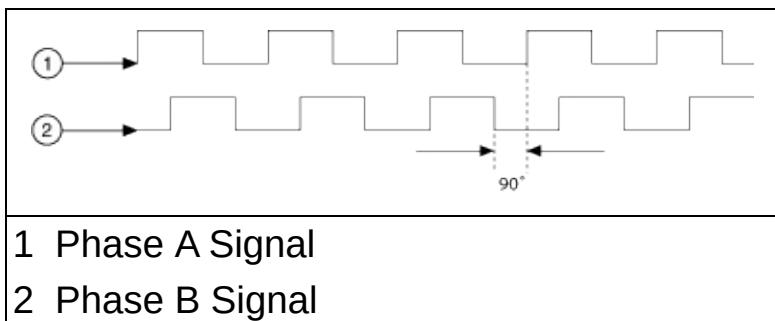
-1074396834	IMG_ERR_BAD_BOARD_HEALTH
-1074396833	IMG_ERR_LIGHTING_BAD_CONTINUOUS_CURRENT_L
-1074396832	IMG_ERR_LIGHTING_BAD_STROBE_DUTY_CYCLE_LIM
-1074396831	IMG_ERR_LIGHTING_BAD_STROBE_DURATION_LIMIT
-1074396830	IMG_ERR_BAD_LIGHTING_CURRENT_EXPOSURE_CO
-1074396829	IMG_ERR_LIGHTING_HEAD_CONFIG_NOT_FOUND
-1074396828	IMG_ERR_LIGHTING_HEAD_DATA_CORRUPT
-1074396827	IMG_ERR_LIGHTING_ABORT_TIMEOUT
-1074396826	IMG_ERR_LIGHTING_BAD_STROBE_CURRENT_LIMIT
-1074396825	IMG_ERR_DMA_ENGINE_UNRESPONSIVE



Quadrature Encoder Overview

A quadrature encoder uses two output channels, Phase A and Phase B, to track the position of a rotary shaft. Generally, this shaft is coupled to a motor drive that controls the movement of an object of interest. By monitoring the encoder Phase A and Phase B signals, you can obtain a precise measurement of the object's position.

To generate Phase A and Phase B signals, the quadrature encoder uses two code tracks with sectors positioned 90 degrees out of phase. The phase difference indicates the position and direction of rotation. If Phase A leads Phase B, the shaft is rotating in a clockwise direction. If Phase B leads Phase A, the shaft is rotating in a counter-clockwise direction. The following figure illustrates the Phase A signal leading the Phase B signal.

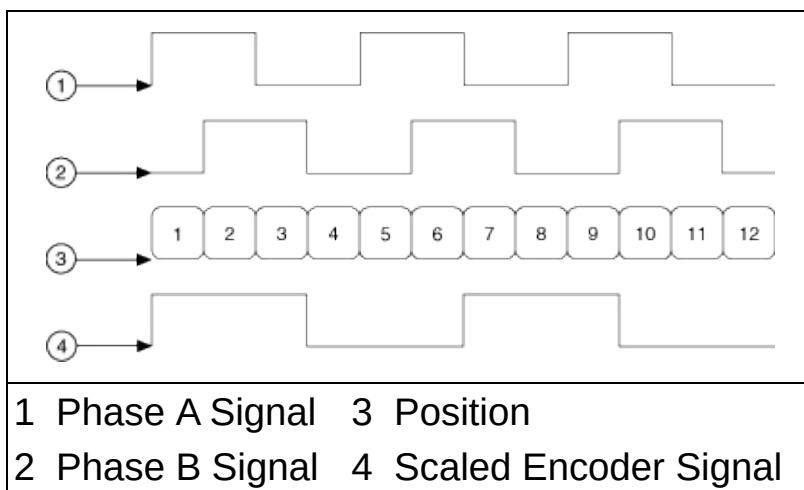


Compatible NI image acquisition devices include hardware that can be used to track both the position and direction of rotation of the Phase A and Phase B signals. For example, this information can be used in conjunction with a line scan camera to acquire lines synchronous to the movement of a conveyor belt, giving you the ability to specify your line rate in terms of positional units (such as inches or centimeters) rather than time.

Scaled Encoder Signal

The scaled encoder signal is an edge-sensitive signal that is used to track cumulative forward progression of the quadrature encoder Phase A and Phase B signals. The scaled encoder signal is derived by applying a divide factor to the raw positional signal that is encoded between Phase A and Phase B.

All NI image acquisition devices expect the raw positional signal to be encoded with quadrature encoding. The scaled encoder signal can be used as a line trigger, as a timebase for pulse generation, and it can be driven out on a trigger line for external usage. The following figure illustrates the scaled encoder signal that is produced when using a divide factor of six.



NI image acquisition devices that support multiple ports have a unique scaler per port. The unique scaler allows you to simultaneously acquire from multiple line scan cameras using different line rates that are all synchronous to the same quadrature encoder. Some NI image acquisition devices also support querying the absolute position counter value. Refer to the image acquisition device documentation to determine if the device supports querying the absolute position counter.

Refer to the [Quadrature Encoder Overview](#) for more information about quadrature encoders.

Variable Data Types

Every function description has a parameter table that lists the data types for each parameter. Refer to [Primary Data Types](#) for notation descriptions used in parameter tables and throughout the documentation for variable data types.

Primary Data Types

The following table shows the primary data type names and their ranges.

Name	Description	Intrinsic Type	Range
Int8	8-bit ASCII character	char	-128 to 127
UInt8	8-bit ASCII character	unsigned char	0 to 255
Int16	16-bit signed integer	short	-32,768 to 32,767
UInt16	16-bit unsigned integer	unsigned short	0 to 65,535
Int32	32-bit signed integer	long	-2,147,483,648 to 2,147,483,647
UInt32	32-bit unsigned integer	unsigned long	0 to 4,294,967,295
Int64	64-bit signed integer	long	-9,223,372,036,854,762,501 to 9,223,372,036,854,762,499
UInt64	64-bit unsigned integer	unsigned long	0 to 18,446,744,073,709,525,000

Programming Language Considerations

Apart from the data type differences, there are a few language-dependent considerations you need to be aware of when you use the NI-IMAQ API.



Note Be sure to include the NI-IMAQ function prototypes by including the appropriate NI-IMAQ header file in the source code.

[LabVIEW](#)

[LabWindows/CVI](#)

[Other Programming Environments](#)

[Code Examples](#)

LabVIEW

Refer to the *NI-IMAQ VI Reference Help* for information about how to use LabVIEW VIs with NI Vision.

LabWindows/CVI

Inside the LabWindows/CVI environment, select **Libraries»NI-IMAQ** to access the NI-IMAQ functions. Each function panel represents an NI-IMAQ function, which is displayed at the bottom of the panel.

The following table shows the organization of the LabWindows/CVI function panel tree and the NI-IMAQ function name that corresponds to each function panel.

LabWindows/CVI Function Panel	NI-IMAQ Function
Interface Functions	
Interface Open	imgInterfaceOpen
Session Open	imgSessionOpen
Close Object	imgClose
Interface Query Names	imgInterfaceQueryNames
Interface Reset	imgInterfaceReset
High-Level Snap Functions	
Snap	imgSnap
Snap Area	imgSnapArea
High-Level Grab Functions	
Grab	imgGrab
Grab Area	imgGrabArea
Grab Setup	imgGrabSetup
High-Level Ring and Sequence Functions	
Ring Setup	imgRingSetup
Sequence Setup	imgSequenceSetup
Session Start Acquisition	imgSessionStartAcquisition
Session Stop Acquisition	imgSessionStopAcquisition
High-Level Signal I/O Functions	
Encoder Reset Position	imgEncoderResetPosition

Line Trigger Source	imgSessionLineTrigSource2
Pulse Create	imgPulseCreate2
Pulse Dispose	imgPulseDispose
Pulse Rate	imgPulseRate
Pulse Start	imgPulseStart
Pulse Stop	imgPulseStop
Trigger Configure	imgSessionTriggerConfigure2
Trigger Clear	imgSessionTriggerClear
Trigger Drive	imgSessionTriggerDrive2
Trigger Read	imgSessionTriggerRead2
Trigger Route	imgSessionTriggerRoute2
Wait Signal	imgSessionWaitSignal2
Wait Signal Asynchronous	imgSessionWaitSignalAsync2

High-Level Miscellaneous Functions

Session Get Buffer Size	imgSessionGetBufferSize
Session Get ROI	imgSessionGetROI
Session Configure ROI	imgSessionConfigureROI
Session Status	imgSessionStatus
Session Fit ROI	imgSessionFitROI

Low-Level Acquisition Functions

Session Abort	imgSessionAbort
Session Acquire	imgSessionAcquire
Session Configure	imgSessionConfigure
Session Copy Area	imgSessionCopyArea
Session Copy Buffer	imgSessionCopyBuffer
Session Examine Buffer	imgSessionExamineBuffer
Session Release Buffer	imgSessionReleaseBuffer

Low-Level Attribute Functions

Get Attribute	imgGetAttribute
---------------	-----------------

Set Attribute	imgSetAttribute
Get Camera Attribute Numeric	imgGetCameraAttributeNumeric
Set Camera Attribute Numeric	imgSetCameraAttributeNumeric
Get Camera Attribute String	imgGetCameraAttributeString
Set Camera Attribute String	imgSetCameraAttributeString
Set User LUT 8 bit	imgSessionSetUserLUT8bit
Set User LUT 16 bit	ImgSessionSetUserLUT16bit

Low-Level Buffer Management Functions

Create Buffer	imgCreateBuffer
Create Buffer List	imgCreateBufList
Dispose Buffer	imgDisposeBuffer
Dispose Buffer List	imgDisposeBufList
Get Buffer Element	imgGetBufferElement
Set Buffer Element	imgSetBufferElement

Low-Level Utility Functions

Bayer Color Decode	imgBayerColorDecode
Calculate Bayer Color LUT	imgCalculateBayerColorLUT
Plot Buffer to DC	imgPlotDC
Plot Buffer to Window	imgPlot
Session Save Buffer	imgSessionSaveBufferEx
Show Error	imgShowError

Low-Level Serial Communication Functions

Session Serial Flush	imgSessionSerialFlush
Session Serial Read	imgSessionSerialRead
Session Serial Read Bytes	imgSessionSerialReadBytes
Session Serial Write	imgSessionSerialWrite

Other Programming Environments

If you are using other programming languages with the NI Vision system, such as Microsoft Visual C++, use the functions described in this help file. Refer to the *NI-IMAQ Help* and the *NI-IMAQ VI Reference Help* for information about programming environments.

Code Examples

You can find code examples in the <CVI>\samples\IMAQ directory, where <CVI> is the directory to which you installed LabWindows/CVI. You can find source code for other development environments in the <NI-IMAQ>\Sample directory, where <NI-IMAQ> is the directory to which you installed NI-IMAQ.

Glossary

A B C D E F G H I L M N O P Q R S T
U V W Y

A

A/D	Analog-to-digital.
AC	Alternating current.
acquisition window	The image size specific to a video standard or camera resolution.
active line region	The region of lines actively being stored; defined by a line start (relative to vertical sync signal) and a line count.
active pixel region	The region of pixels actively being stored; defined by a pixel start (relative to the horizontal sync signal) and a pixel count.
ADC	Analog-to-digital converter. An electronic device, often an integrated circuit, that converts an analog voltage to a digital number.
address	Character code that identifies a specific location (or series of locations) in memory.
ANSI	American National Standards Institute.
antichrominance filter	Removes the color information from the video signal.
API	Application programming interface.
area	A rectangular portion of an acquisition window or frame that is controlled and defined by software.
array	Ordered, indexed set of data elements of the same type.
ASIC	Application-specific integrated circuit. A proprietary semiconductor component designed and manufactured to perform a set of specific functions for a specific customer.
aspect ratio	The ratio of a picture or image's width to its height.

B

back porch	The area of the video signal between the rising edge of the horizontal sync signal and the active video information.
Bayer encoding	Method to produce color images with a single imaging sensor, as opposed to three individual sensors for the red, green, and blue components of light.
Bayer pattern	Color filter array pattern that can appear in four variations, depending on the current left and top offsets of the acquisition window: GBGB GRGR BGBG RGRG RGRG BGBG GRGR GBGB
bit depth	The number of bits per pixel.
black level	The level that represents the darkest an image can get. See reference also white reference level .
BMP	Bitmap. Image file format commonly used for 8-bit and color images (extension .bmp).
buffer	Temporary storage for acquired data.
bus	The group of conductors that interconnect individual circuitry in a computer, such as the PCI bus; typically the expansion vehicle to which I/O or other devices are connected.

C

cache	High-speed processor memory that buffers commonly used instructions or data to increase processing throughput.
CCIR	Comite Consultatif International des Radiocommunications. A committee that developed standards for color video signals.
chrominance	The color information in a video signal.
CMOS	Complementary metal-oxide semiconductor.
CompactPCI	Refers to the core specification defined by the PCI Industrial Computer Manufacturer's Group (PICMG).
compiler	A software utility that converts a source program in a high-level programming language, such as Basic, C, or Pascal, into an object or compiled program in machine language. Compiled programs run 10 to 1,000 times faster than interpreted programs. See also interpreter .
conversion device	Device that transforms a signal from one form to another; for example, analog-to-digital converters (ADCs) for analog input and digital-to-analog converters (DACs) for analog output.
CPU	Central processing unit.
CSYNC	Composite sync signal. A combination of the horizontal and vertical sync pulses.

D

D/A	Digital-to-analog.
DAC	Digital-to-analog converter; an electronic device, often an integrated circuit, that converts a digital number into a corresponding analog voltage or current.
DAQ	Data acquisition. (1) Collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing. (2) Collecting and measuring the same kinds of electrical signals with A/D or DIO devices plugged into a computer, and possibly generating control signals with D/A and/or DIO devices in the same computer.
DC	Direct current.
default setting	A default parameter value recorded in the driver; in many cases, the default input of a control is a certain value (often 0) that means use the current default setting.
DIN	Deutsche Industrie Norme. A format for electrical connectors.
distance calibration	Determination of the physical dimensions of a pixel by defining the physical dimensions of a line in the image.
distance function	Assigns to each pixel in an object a gray-level value equal to its shortest Euclidean distance from the border of the object.
DLL	Dynamic link library. A software module in Microsoft Windows containing executable code and data that can be called or used by Windows applications or other DLLs; functions and data in a DLL are loaded and linked at run time when they are referenced by a Windows application or other DLLs.
DMA	Direct memory access. A method by which data can be transferred to and from computer memory from and to a device or memory on the bus while the processor does something else; DMA is the fastest method of transferring data to/from computer memory.
DRAM	Dynamic RAM.
driver	Software that controls a specific hardware device such as an image acquisition device.
dynamic	The ratio of the largest signal level a circuit can handle to

E

EEPROM Electrically erasable programmable read-only memory. ROM that can be erased with an electrical signal and reprogrammed.

external trigger A voltage pulse from an external source that triggers an event such as A/D conversion.

F

field	For an interlaced video signal, a field is half the number of horizontal lines needed to represent a frame of video; the first field of a frame contains all of the odd-numbered lines, and the second field contains all of the even-numbered lines.
FIFO	First-in first-out memory buffer. The first data stored is the first data sent to the acceptor; FIFO buffers are used on image acquisition devices to temporarily store incoming data until that data can be retrieved.
flash ADC	An ADC whose output code is determined in a single step by a bank of comparators and encoding logic.
frame	A complete image; in interlaced formats, a frame is composed of two fields.
front porch	The area of a video signal between the start of the horizontal blank and the start of the horizontal sync.
function	A set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed.

G

- gain Applied value to compensate for discrepancies in the filter for a particular color.
- gamma The nonlinear change in the difference between the video signal's brightness level and the voltage level needed to produce that brightness.
- genlock The process of synchronizing a video source to the signal from a separate video source. The circuitry aligns the video timing signals by locking together the horizontal, vertical, and color subcarrier frequencies and phases and generates a pixel clock that clocks pixel data into memory for display or into another circuit for processing.
- grab Performs an acquisition that loops continually on one buffer. You obtain a copy of the acquisition buffer by grabbing a copy to a separate buffer that can be used for analysis.
- GUI Graphical user interface. An intuitive, easy-to-use means of communicating information to and from a computer program by means of graphical screen displays; GUIs can resemble the front panels of instruments or other objects associated with a computer program.

H

- hardware The physical components of a computer system, such as the circuit boards, plug-in boards, chassis, enclosures, peripherals, cables, and so on.
- HSYNC Horizontal sync signal. The synchronization pulse signal produced at the beginning of each video scan line that keeps a video monitor's horizontal scan rate in step with the transmission of each new line.
- hue Represents the dominant color of a pixel. The hue function is a continuous function that covers all the possible colors generated using the R, G, and B primaries. **See also** [RGB](#).

I

IEEE	Institute of Electrical and Electronics Engineers.
INL	Integral nonlinearity. A measure, in LSB, of the worst-case deviation from the ideal A/D or D/A transfer characteristic of the analog I/O circuitry.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that controls specific plug-in computer boards; instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW.
interlaced	A video frame composed of two interleaved fields; the number of lines in a field are half the number of lines in an interlaced frame.
interpreter	A software utility that executes source code from a high-level language, such as Java or Basic, by reading one line at a time and executing the specified operation. In contrast, a compiler converts all source code to executable machine code before execution. Compiled languages give significantly higher performance than interpreted languages. Examples of compiled languages are C, C++, and LabVIEW, while Java and Basic are generally interpreted languages. See also compiler .
interrupt	A computer signal indicating that the CPU should suspend its current task to service a designated activity.
interrupt level	The relative priority at which a device can interrupt.
I/O	Input/output. The transfer of data to/from a computer system involving communications channels, operator interface devices, or data acquisition and control interfaces.
IRE	A relative unit of measure (named for the Institute of Radio Engineers). 0 IRE corresponds to the blanking level of a video signal, 100 IRE to the white level. Note that for CIR/PAL video the black level is equal to the blanking level or 0 IRE, while for RS-170/NTSC video, the black level is at 7.5 IRE.
IRQ	Interrupt request. See also interrupt .
ISO Trigger	A high voltage isolated trigger.

L

library	A file containing compiled object modules, each comprised of one or more functions, that can be linked to other object modules that make use of these functions.
line count	The total number of horizontal lines in the picture.
LSB	Least significant bit.
luminance	The brightness information in the video picture. The luminance signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.
LUT	Lookup table. Table containing values used to transform the gray-level values of an image. For each gray-level value in the image, the corresponding new value is obtained from the lookup table. Also a selection in Measurement & Automation Explorer (MAX) for Vision that contains formulas that let you implement simple imaging operations such as contrast enhancement, data inversion, gamma manipulation, or other nonlinear transfer functions.

M

- MAX Measurement & Automation Explorer. The National Instruments Windows-based graphical configuration utility you can use to configure NI software and hardware, execute system diagnostics, add new channels and interfaces, and view the devices and instruments you have connected to your computer. MAX is installed on the desktop during the National Instruments driver software installation.
- memory See [buffer](#).
- buffer
- memory Continuous blocks of memory that can be accessed quickly
window by changing addresses on the local processor.
- MSB Most significant bit.
- MTBF Mean time between failure.
- mux Multiplexer. A switching device with multiple inputs that selectively connects one of its inputs to its output.

N

NI-IMAQ	Driver software for National Instruments image acquisition hardware.
noninterlaced	A video frame where all the lines are scanned sequentially, rather than being divided into two frames as in an interlaced video frame.
NTSC	National Television Standards Committee. The committee that developed the color video standard used primarily in North America, which uses 525 lines per frame. See also PAL .
NVRAM	Nonvolatile RAM. RAM that is not erased when a device loses power or is turned off.

O

one- Applies to pulse generation and acquisitions. A one-shot pulse
shot or acquisition happens only once.

P

PAL	Phase Alternation Line. One of the European video color standards; uses 625 lines per frame. See also NTSC .
PCI	Peripheral Component Interconnect. A high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA; it is achieving widespread acceptance as a standard for PCs and workstations and offers a theoretical maximum transfer rate of 133 Mbytes/s.
PCIe	PCI express. A high-performance expansion bus architecture originally developed by Intel to replace PCI. PCIe offers a theoretical maximum transfer rate that is dependent upon lane width. A x1 link theoretically provides 250 MB/s in each direction—to and from the device. Once overhead is accounted for, a x1 link can provide approximately 200 MB/s of input capability and 200 MB/s of output capability. Increasing the number of lanes in a link increases maximum throughput by approximately the same factor.
PCLK	Pixel clock signal. Times the sampling of pixels on a video line.
PGIA	Programmable gain instrumentation amplifier.
picture aspect ratio	The ratio of the active pixel region to the active line region; for standard video signals such as RS-170 or CCIR, the full-size picture aspect ratio typically is 4/3 (1.33).
pixel	Picture element. The smallest division that makes up the video scan line; for display on a computer monitor, a pixel's optimum dimension is square (aspect ratio of 1:1, or the width equal to the height).
pixel aspect ratio	The ratio between the physical horizontal size and the vertical size of the region covered by the pixel. An acquired pixel should optimally be square, thus the optimal value is 1.0; however, typically it falls between 0.95 and 1.05, depending on camera quality.
pixel clock	Divides the incoming horizontal video line into pixels.
pixel count	The total number of pixels between two horizontal sync signals; the pixel count determines the frequency of the pixel clock.
PLL	Phase-locked loop. Circuitry that provides a very stable pixel clock that is referenced to another signal, for example, an external reference signal.

Q

quadrature encoder An encoding technique for a rotating device where two tracks of information are placed on the device, with the signals on the tracks offset by 90 degrees from each other. The phase difference indicates the position and direction of rotation.

R

RAM	Random-access memory.
real time	A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.
relative accuracy	A measure in LSB of the accuracy of an ADC; it includes all nonlinearity and quantization errors but does not include offset and gain errors of the circuitry feeding the ADC.
resolution	(1) The number of rows and columns of pixels. An image composed of m rows and n columns has a resolution of . This image has n pixels along its horizontal axis and m pixels along its vertical axis. (2) The smallest signal increment that can be detected by a measurement system. Resolution can be expressed in bits, proportions, or a percentage of full scale. For example, a system has 12-bit resolution, one part in 4,096 resolution, and 0.0244 percent of full scale.
RGB	Color encoding scheme using red, green, and blue (RGB) color information where each pixel in the color image is encoded using 32 bits: 8 bits for red, 8 bits for green, 8 bits for blue, and 8 bits for the alpha value (unused).
ribbon cable	A flat cable in which the conductors are side by side.
ring	Performs an acquisition that loops continually on a specified number of buffers.
ROI	Region of interest. (1) An area of the image that is graphically selected from a window displaying the image. This area can be used focus further processing. (2) A hardware-programmable rectangular portion of the acquisition window.
ROM	Read-only memory.
RS-170	The U.S. standard used for black-and-white television.
RTSI bus	Real-Time System Integration Bus. The National Instruments timing bus that connects image acquisition and DAQ devices directly, by means of connectors on top of the devices, for precise synchronization of functions.

S

saturation	The amount of white added to a pure color. Saturation relates to the richness of a color. A saturation of zero corresponds to a pure color with no white added. Pink is a red with low saturation.
scaling down circuitry	Circuitry that scales down the resolution of a video signal.
scatter-gather DMA	A type of DMA that allows the DMA controller to reconfigure on-the-fly.
sequence	Performs an acquisition that acquires a specified number of buffers, then stops.
snap	Acquires a single frame or field to a buffer.
SRAM	Static RAM.
StillColor	A post-processing algorithm that allows the acquisition of high-quality color images generated either by an RGB or composite (NTSC or PAL) camera using a monochrome video acquisition device.
sync	Tells the display where to put a video picture; the horizontal sync indicates the picture's left-to-right placement and the vertical sync indicates top-to-bottom placement.
system RAM	RAM installed on a personal computer and used by the operating system, as contrasted with onboard RAM.

T

transfer rate	The rate, measured in bytes/s, at which data is moved from source to destination after software initialization and setup operations; the maximum rate at which the hardware can operate.
trigger	Any event that causes or starts some form of data capture.
trigger control and mapping circuitry	Circuitry that routes, monitors, and drives the external and RTSI bus trigger lines; you can configure each of these lines to start or stop acquisition on a rising or falling edge.
TTL	Transistor-transistor logic. A digital circuit composed of bipolar transistors wired in a certain manner. A typical medium-speed digital technology. Nominal TTL logic levels are 0 and 5 V.

U

UV plane **See** [YUV](#).

V

- VCO Voltage-controlled oscillator. An oscillator that changes frequency depending on a control signal; used in a PLL to generate a stable pixel clock.
- VI Virtual Instrument. (1) A combination of hardware and/or software elements, typically used with a PC, that has the functionality of a classic stand-alone instrument (2) A LabVIEW software module (VI), which consists of a front panel user interface and a block diagram program.
- video line A video line consists of a horizontal sync signal, back porch, active pixel region, and a front porch.
- VSYNC Vertical sync signal. The synchronization pulse generated at the beginning of each video field that tells the video monitor when to start a new field.

W

white reference level The level that defines what is white for a particular video system. **See also** [black reference level](#).

Y

YUV A representation of a color image used for the coding of NTSC or PAL video signals. The luminance information is called Y, while the chrominance information is represented by two components, U and V, that represent the coordinates in a color plane.

Important Information

[Warranty](#)

[Copyright](#)

[Trademarks](#)

[Patents](#)

[Warning Regarding Use of NI Products](#)

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action

accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about [National Instruments trademarks](#).

FireWire® is the registered trademark of Apple Computer, Inc.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix® and Tek are registered trademarks of Tektronix, Inc.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the patents.txt file on your CD, or ni.com/patents.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR

APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Technical Support and Professional Services

Visit the following sections of the award-winning National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Technical support resources at ni.com/support include the following:
 - **Self-Help Resources**—For answers and solutions, visit ni.com/support for software drivers and updates, a searchable [KnowledgeBase](#), [product manuals](#), step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on. Registered users also receive access to the [NI Discussion Forums](#) at ni.com/forums. NI Applications Engineers make sure every question submitted online receives an answer.
 - **Standard Service Program Membership**—This program entitles members to direct access to NI Applications Engineers via phone and email for one-to-one technical support, as well as exclusive access to on demand training modules via the [Services Resource Center](#). NI offers complementary membership for a full year after purchase, after which you may renew to continue your benefits.
- For information about other [technical support options](#) in your area, visit ni.com/services or [contact](#) your local office at ni.com/contact.
- [Training and Certification](#)—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- [System Integration](#)—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your [local office](#) or NI corporate headquarters. You also can visit the [Worldwide Offices](#) section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

imgSessionExamineBuffer

Usage

```
rval imgSessionExamineBuffer(SESSION_ID sid, uInt32 whichBuffer, void*  
bufferNumber, uInt32* bufferAddr);
```

Purpose

Extracts an image from a live acquisition. This function lets you lock an image out of a continuous loop sequence for processing when you are using a ring (continuous) sequence. If the requested image has been acquired and exists in memory, the function returns that image immediately. If the requested image has not yet been acquired, the function does not return until the image has been acquired or the timeout period has expired. If the requested image has already been overwritten, the function returns the most current image. If the buffer remains extracted long enough that the acquisition hardware wraps around the buffer list and encounters the extracted buffer again, the acquisition will stall, increment the lost frame count, and the extracted buffer will not be overwritten.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
whichBuffer	ulnt32	input
bufferNumber	void*	input
bufferAddr	ulnt32*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

whichBuffer: cumulative image number to extract. Pass IMG_CURRENT_BUFFER to get the buffer that is currently being acquired.

bufferNumber: on return, the function populates this parameter with the cumulative number of the returned image.

bufferAddr: on return, the function populates this parameter with the address of the locked image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note Use [imgSessionReleaseBuffer](#) to release the image being held with imgSessionExamineBuffer.

imgSetAttribute

Usage

```
rval imgSetAttribute(UInt32 void_id, UInt32 attr, UInt32 value);
```

Purpose

Sets an attribute value.

Parameters

Name	Type	Direction
void_id	ulnt32	input
attr	ulnt32	input
value	ulnt32, ulnt64*, or double*	input
rval	Int32	output

Parameter Discussion

void_id valid SESSION_ID or INTERFACE_ID.

attr: attribute to set. Refer to [Attributes](#) for valid types.

value: new attribute value. The type of this input is either uint32, a pointer to a uint64, or a pointer to a double.

If the attribute is of type uint32, use this function as follows:

```
imgSetAttribute (sid, IMG_ATTR_ROI_HEIGHT, value)
```

If the attribute is of type uint64, use this function as follows:

```
imgSetAttribute (sid, key,(uint32) &value)
```

If the attribute is of type double, use this function as follows:

```
imgSetAttribute (sid, IMG_ATTR_BLACK_REF_VOLT,(uint32) &value)
```

Refer to [Attributes](#) for the attribute type.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgSetBufferElement

Usage

```
rval imgSetBufferElement(BUFLIST_ID bid, uInt32 element, uInt32 itemType,  
uInt32 itemValue);
```

Purpose

Sets the value for a specified **itemType** for a buffer in a buffer list.

Parameters

Name	Type	Direction
bid	BUFLIST_ID	input
element	UInt32	input
itemType	UInt32	input
itemValue	UInt32	input
rval	Int32	output

Parameter Discussion

bid: valid BUFLIST_ID variable.

element: element number of the buffer list item to modify.

itemType: describes the parameter of the element to set, as specified by the following constants:

IMG_BUFF_ADDRESS	Specifies the buffer address portion of a buffer list element.
IMG_BUFF_CHANNEL	Specifies the channel from which to acquire an image.
IMG_BUFF_COMMAND	Specifies the command portion of a buffer list element.
IMG_BUFF_SIZE	Specifies the size portion of a buffer list element (the buffer size). Required for user-allocated buffers.
IMG_BUFF_SKIPCOUNT	Specifies the skip count portion of a buffer list element.

itemValue: indicates the value of the element type to set. Use the following constants to specify the IMG_BUFF_COMMAND:

IMG_CMD_LOOP	Specifies a buffer list command of LOOP. Used as the command for the last buffer element, this constant causes an acquisition to perform a continuous type of acquisition, such as a ring.
IMG_CMD_NEXT	Specifies a buffer list command of NEXT. This constant causes an acquisition to take place on the buffer and to proceed to the next buffer list element.
IMG_CMD_STOP	Specifies a buffer list command of STOP. Used as the command for the last buffer element, this constant causes an acquisition to perform a one-shot acquisition such as a sequence.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).



Note Refer to [Constants](#) for valid element and command types.

imgPlot

Usage

```
rval imgPlot(GUIHNDL window, const void* buffer, uInt32 leftBufOffset,  
uInt32 topBufOffset, uInt32 xsize, uInt32 ysize, uInt32 xpos, uInt32 ypos,  
uInt32 flags);
```

Purpose

Plots a buffer to a window given a native Windows handle. Use this function to display a buffer after it is acquired.

Parameters

Name	Type	Direction
window	GUIHNDL	input
buffer	const void*	input
leftBufOffset	ulInt32	input
topBufOffset	ulInt32	input
xsize	ulInt32	input
ysize	ulInt32	input
xpos	ulInt32	input
ypos	ulInt32	input
flags	ulInt32	input
rval	Int32	output

Parameter Discussion

window: native Windows handle designating the window in which to plot.

buffer: image to plot.

leftBufOffset: left offset into the image to start plotting.

topBufOffset: top offset into the image to start plotting.

xsize: width of the image, in pixels.

ysize: number of lines in the image.

xpos: left position to start plotting in the window.

ypos: top position to start plotting in the window.

flags: sets the display property. **flags** is used with the following constants:

IMGPLOT_COLOR_RGB32	Specifies a 32-bit color RGB image.
IMGPLOT_COLOR_HSL32	Specifies a 32-bit color HSL image.
IMGPLOT_INVERT	Specifies to invert the image when plotted.
IMGPLOT_MONO_8	Specifies an 8-bit monochrome image.
IMGPLOT_MONO_10	Specifies a 10-bit monochrome image.
IMGPLOT_MONO_12	Specifies a 12-bit monochrome image.
IMGPLOT_MONO_14	Specifies a 14-bit monochrome image.
IMGPLOT_MONO_16	Specifies a 16-bit monochrome image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPlotDC

Usage

```
rval imgPlotDC(GUIHNDL dc, const void* buffer, uInt32 leftBufOffset, uInt32  
topBufOffset, uInt32 xsize, uInt32 ysize, uInt32 xpos, uInt32 ypos, uInt32 flags);
```

Purpose

Plots a buffer to a device context given a device context handle.

Parameters

Name	Type	Direction
dc	GUIHNDL	input
buffer	const void*	input
leftBufOffset	ulInt32	input
topBufOffset	ulInt32	input
xsize	ulInt32	input
ysize	ulInt32	input
xpos	ulInt32	input
ypos	ulInt32	input
flags	ulInt32	input
rval	Int32	output

Parameter Discussion

dc: native Windows device context in which to draw.

buffer: image to plot.

leftBufOffset: left offset into the image to start plotting.

topBufOffset: top offset into the buffer to start plotting.

xsize: width of the image, in pixels.

ysize: number of lines in the image.

xpos: left position to start plotting in the window.

ypos: top position to start plotting in the window.

flags: sets the display property. **flags** is used with the following constants:

IMGPLOT_INVERT	Specifies to invert the image when plotted.
IMGPLOT_MONO_8	Specifies a 8-bit monochrome image.
IMGPLOT_MONO_10	Specifies a 10-bit monochrome image.
IMGPLOT_MONO_12	Specifies a 12-bit monochrome image.
IMGPLOT_MONO_14	Specifies a 14-bit monochrome image.
IMGPLOT_MONO_16	Specifies a 16-bit monochrome image.
IMGPLOT_COLOR_HSL32	Specifies a 32-bit color HSL image.
IMGPLOT_COLOR_RGB32	Specifies a 32-bit color RGB image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

imgPulseCreate

Usage

```
rval imgPulseCreate(UInt32 timebase, UInt32 delay, UInt32 width, UInt32 trigger,  
UInt32 triggerPolarity, UInt32 output, UInt32 outputPolarity, UInt32 pulseMode,  
PULSE_ID* plsID);
```

Purpose

Configures the attributes of a pulse. A single pulse consists of a delay phase (phase 1), followed by a pulse phase (phase 2), and then a return to the phase 1 level.

-  **Note** This function is obsolete. The replacement function is [imgPulseCreate2](#), which incorporates the functionality of imgPulseCreate but also returns additional information.
-  **Note** Only two pulses can occur at once.

Parameters

Name	Type	Direction
timebase	ulInt32	input
delay	ulInt32	input
width	ulInt32	input
trigger	ulInt32	input
triggerPolarity	ulInt32	input
output	ulInt32	input
outputPolarity	ulInt32	input
pulseMode	ulInt32	input
plsID	PULSE_ID*	output
rval	Int32	output

Parameter Discussion

timebase: timebase of the counter. **timebase** has the following possible values:

PULSE_TIMEBASE_50MHZ	Specifies a 50 MHz timebase to use for pulse generation.
PULSE_TIMEBASE_100KHZ	Specifies a 100 kHz timebase to use for pulse generation.
PULSE_TIMEBASE_PIXELCLK	Specifies the incoming pixel clock from the camera to use as a timebase for pulse generation.

delay: duration of the first phase of the pulse. Use the following formula to determine the actual time period that delay represents:

$$\text{delay} \times (\text{timebase resolution})$$

width: duration of the second phase of the pulse, phase 2. The unit is cycles of the timebase. Use the following formula to determine the actual time period that width represents:

$$\text{width} \times (\text{timebase resolution})$$

trigger: signal that will trigger the pulse. **trigger** can be one of the following constants:

IMG_AQ_DONE	Asserted at the end of an acquisition when the last piece of data has been transferred to memory.
IMG_AQ_IN_PROGRESS	Asserted when the device initiates an acquisition either through a software- or hardware-triggered start.
IMG_EXT_RTSI0	Specifies RTSI line 0.
IMG_EXT_RTSI1	Specifies RTSI line 1.
IMG_EXT_RTSI2	Specifies RTSI line 2.
IMG_EXT_RTSI3	Specifies RTSI line 3.
IMG_EXT_RTSI4	Specifies RTSI line 4.
IMG_EXT_RTSI5	Specifies RTSI line 5.

IMG_EXT_RTSI6	Specifies RTSI line 6.
IMG_EXT_TRIG0	Specifies the external trigger 0.
IMG_EXT_TRIG1	Specifies the external trigger 1.
IMG_EXT_TRIG2	Specifies the external trigger 2.
IMG_EXT_TRIG3	Specifies the external trigger 3.
IMG_FRAME_START	Asserted at the start of acquisition into each image buffer.
IMG_FRAME_DONE	Asserted at the end of acquisition into each image buffer.
IMG_IMMEDIATE	Causes the function to generate a pulse when the function is executed.

 **Note** `IMG_EXT_TRIG <0..3>` refers to the external trigger lines of the image acquisition device. `IMG_EXT_RTSI <0..6>` refers to the internal pins on the RTSI controller of the image acquisition device.

triggerPolarity: polarity of the signal input as defined by the following constants:

- `IMG_TRIG_POLAR_ACTIVEV` Triggers on a falling edge.
- `IMG_TRIG_POLAR_ACTIVEH` Triggers on a rising edge.

output: trigger line on which the pulse is generated. **output** can be one of the following constants:

- `IMG_EXT_TRIG0` Specifies the external trigger 0.
- `IMG_EXT_TRIG1` Specifies the external trigger 1.
- `IMG_EXT_TRIG2` Specifies the external trigger 2.
- `IMG_EXT_RTSI0` Specifies RTSI line 0.
- `IMG_EXT_RTSI1` Specifies RTSI line 1.
- `IMG_EXT_RTSI2` Specifies RTSI line 2.
- `IMG_EXT_RTSI3` Specifies RTSI line 3.
- `IMG_EXT_RTSI4` Specifies RTSI line 4.
- `IMG_EXT_RTSI5` Specifies RTSI line 5.
- `IMG_EXT_RTSI6` Specifies RTSI line 6.



Note `IMG_EXT_TRIG <0..3>` refers to the external trigger lines of the image acquisition device. `IMG_EXT_RTSI <0..6>` refers to the internal pins on the RTSI controller of the image acquisition device.

outputPolarity: polarity of the pulse output as defined by the following constants:

`IMG_PULSE_POLAR_ACTIVEL` Drives the line high during the delay phase, and drives the line low during the pulse phase.

`IMG_PULSE_POLAR_ACTIVEH` Drives the line low during the delay phase, and drives the line high during the pulse phase.

pulseMode: value that indicates if the pulse is generated once or continuously. **pulseMode** can be one of the following constants:

`PULSE_MODE_TRAIN` Pulse is generated continuously after the trigger is asserted. Choose this option to generate a continuous pulse train that is inactive for the number of cycles specified in the **delay** parameter, and active for the number of cycles specified in the **width** parameter. When the pulse train is started, it continues periodically until you call `imgPulseStop`, `imgPulseDispose`, or `imgClose`.

`PULSE_MODE_SINGLE` Pulse occurs one time when the first trigger occurs. Choose this option to generate a single pulse. On the first occurrence of `signal_source`, the output line stays inactive for the number of cycles specified in the **delay** parameter, and becomes active for the number of cycles specified in the **width** parameter. Future occurrences of `signal_source` do not affect the output line.

PULSE_MODE_SINGLE_REARM Pulse occurs one time on each trigger occurrence. Choose this option to generate a rearmed single shot pulse. On every occurrence of signal_source, the output line stays inactive for the number of cycles specified in the **delay** parameter, and becomes active for the number of cycles specified in the **width** parameter. When the pulse is started, output toggles for each occurrence of signal_source until you call imgPulseStop, imgPulseDispose, or imgClose.

pIsID: pointer to a variable to receive the pulse ID. If the function succeeds, the variable is populated with a valid PULSE_ID that can be used in subsequent functions.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

imgSessionWaitSignal

Usage

```
rval imgSessionWaitSignal(SESSION_ID sid, uInt32 signal, uInt32 state, uInt32  
timeout);
```

Purpose

Waits for a signal to be in a given state. This function returns when either the specified signal is in a given state or the wait times out.



Note This function is obsolete. The replacement function is [imgSessionWaitSignal2](#), which incorporates the functionality of imgSessionWaitSignal but also returns additional information.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
signal	UInt32	input
state	UInt32	input
timeout	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

signal: signal to wait for. The signal can one of the following constants:

IMG_AQ_IN_PROGRESS	Asserted when the device initiates an acquisition either through a software- or hardware-triggered start.
IMG_AQ_DONE	Asserted at the end of an acquisition when the last piece of data has been transferred to memory.
IMG_FRAME_START	Asserted at the start of acquisition into each image buffer.
IMG_FRAME_DONE	Asserted at the end of acquisition into each image buffer.
IMG_BUF_COMPLETE	Asserted when an image buffer has been transferred to memory and is available for image processing.
IMG_EXT_TRIG0	Specifies the external trigger 0.
IMG_EXT_TRIG1	Specifies the external trigger 1.
IMG_EXT_TRIG2	Specifies the external trigger 2.
IMG_EXT_TRIG3	Specifies the external trigger 3.
IMG_EXT_RTSI0	Specifies RTSI line 0.
IMG_EXT_RTSI1	Specifies RTSI line 1.
IMG_EXT_RTSI2	Specifies RTSI line 2.
IMG_EXT_RTSI3	Specifies RTSI line 3.
IMG_EXT_RTSI4	Specifies RTSI line 4.
IMG_EXT_RTSI5	Specifies RTSI line 5.
IMG_EXT_RTSI6	Specifies RTSI line 6.



Note IMG_EXT_TRIG <0..3> refers to the external trigger lines of the image acquisition device. IMG_EXT_RTSI <0..6> refers to the internal pins on the RTSI controller of the image acquisition device.

state: state of the signal to wait for. **state** can be one of the following constants:

IMG_SIGNAL_STATE_RISING Waits for a rising edge.

IMG_SIGNAL_STATE_FALLING Waits for a falling edge.

timeout: time, in milliseconds, to wait for the appropriate state. If the appropriate state does not occur within the timeout period, the function returns IMG_ERR_TIMEOUT.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

imgSessionWaitSignalAsync

Usage

```
rval imgSessionWaitSignalAsync(SESSION_ID sid, UInt32 signal, UInt32 state,  
CALL_BACK_PTR function, void* data);
```

Purpose

Waits for a signal to be in a given state and when the signal is in that state, calls a user-defined function.



Note This function is obsolete. The replacement function is [imgSessionWaitSignalAsync2](#), which incorporates the functionality of imgSessionWaitSignalAsync but also returns additional information.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
signal	uInt32	input
state	uInt32	input
function	CALL_BACK_PTR	input
data	void*	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

signal: signal to wait for. The signal can one of the following constants:

IMG_AQ_IN_PROGRESS	Asserted when the device initiates an acquisition either through a software- or hardware-triggered start.
IMG_AQ_DONE	Asserted at the end of an acquisition when the last piece of data has been transferred to memory.
IMG_FRAME_START	Asserted at the start of acquisition into each image buffer.
IMG_FRAME_DONE	Asserted at the end of acquisition into each image buffer.
IMG_BUF_COMPLETE	Asserted when an image buffer has been transferred to memory and is available for image processing.
IMG_EXT_TRIG0	Specifies the external trigger 0.
IMG_EXT_TRIG1	Specifies the external trigger 1.
IMG_EXT_TRIG2	Specifies the external trigger 2.
IMG_EXT_TRIG3	Specifies the external trigger 3.
IMG_EXT_RTSI0	Specifies RTSI line 0.
IMG_EXT_RTSI1	Specifies RTSI line 1.
IMG_EXT_RTSI2	Specifies RTSI line 2.
IMG_EXT_RTSI3	Specifies RTSI line 3.
IMG_EXT_RTSI4	Specifies RTSI line 4.
IMG_EXT_RTSI5	Specifies RTSI line 5.
IMG_EXT_RTSI6	Specifies RTSI line 6.



Note IMG_EXT_TRIG <0..3> refers to the external trigger lines of the image acquisition device. IMG_EXT_RTSI <0..6> refers to the internal pins on the RTSI controller of the image acquisition device.

state: state of the signal to wait for. **state** can be one of the following constants:

IMG_SIGNAL_STATE_RISING Waits for a rising edge.

IMG_SIGNAL_STATE_FALLING Waits for a falling edge.

IMG_SIGNAL_STATE_HIGH Returns immediately if the signal is high. Otherwise, waits for a rising edge.

IMG_SIGNAL_STATE_LOW Returns immediately if the signal is low. Otherwise, waits for a falling edge.

function: pointer to the callback function. Your function should match the following prototype:

```
uInt32(*function)(SESSION_ID sid, IMG_ERR err, uInt32 signal, void*  
userdata)
```

 **Note** The return value of the callback function determines the behavior of the driver for subsequent signal assertions. Return zero to disregard future signal assertions. Return a non-zero value to reinstate the callback function.

data value that is passed to the callback function. The value can be a pointer to user data.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

imgSessionTriggerConfigure

Usage

```
rval imgSessionTriggerConfigure(SESSION_ID sid, uInt32 trigger, uInt32  
polarity, uInt32 timeout, uInt32 action);
```

Purpose

Configures an acquisition to start based on an external trigger.



Note This function is obsolete. The replacement function is [imgSessionTriggerConfigure2](#), which incorporates the functionality of imgSessionTriggerConfigure but also returns additional information.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
trigger	UInt32	input
polarity	UInt32	input
timeout	UInt32	input
action	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

trigger: trigger line to use. **trigger** can be one of the following constants:

[IMG_EXT_TRIG0](#) [IMG_EXT_RTSI2](#)

[IMG_EXT_TRIG1](#) [IMG_EXT_RTSI3](#)

[IMG_EXT_TRIG2](#) [IMG_EXT_RTSI4](#)

[IMG_EXT_TRIG3](#) [IMG_EXT_RTSI5](#)

[IMG_EXT_RTSI0](#) [IMG_EXT_RTSI6](#)

[IMG_EXT_RTSI1](#)

 **Note** `IMG_EXT_TRIG <0..3>` refers to the external trigger lines of the image acquisition device. `IMG_EXT_RTSI <0..6>` refers to the internal pins on the RTSI controller of the image acquisition device.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEV` Triggers on a falling edge.

`IMG_TRIG_POLAR_ACTIVEH` Triggers on a rising edge.

timeout: time, in milliseconds, to wait for the trigger to occur.

action: action to take when the trigger edge occurs. The following are valid values for **action**:

`IMG_TRIG_ACTION_NONE` Clears the trigger.

`IMG_TRIG_ACTION_CAPTURE` Starts acquiring.

`IMG_TRIG_ACTION_BUFLIST` Acquires the buffer list once.

`IMG_TRIG_ACTION_BUFFER` Acquires a single image.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

imgSessionTriggerDrive

Usage

```
rval imgSessionTriggerDrive(SESSION_ID sid, uInt32 trigger, uInt32 polarity,  
uInt32 source);
```

Purpose

Configures the specified trigger line to drive a signal out.



Note This function is obsolete. The replacement function is [imgSessionTriggerDrive2](#), which incorporates the functionality of imgSessionTriggerDrive but also returns additional information.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
trigger	UInt32	input
polarity	UInt32	input
source	UInt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

trigger: trigger line to drive. **trigger** can be one of the following constants:

[IMG_EXT_TRIG0](#) [IMG_EXT_RTSI2](#)

[IMG_EXT_TRIG1](#) [IMG_EXT_RTSI3](#)

[IMG_EXT_TRIG2](#) [IMG_EXT_RTSI4](#)

[IMG_EXT_TRIG3](#) [IMG_EXT_RTSI5](#)

[IMG_EXT_RTSI0](#) [IMG_EXT_RTSI6](#)

[IMG_EXT_RTSI1](#)

 **Note** `IMG_EXT_TRIG <0..3>` refers to the external trigger lines of the image acquisition device. `IMG_EXT_RTSI <0..6>` refers to the internal pins on the RTSI controller of the image acquisition device.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEV` Drives the line low when the signal is true.

`IMG_TRIG_POLAR_ACTIVEH` Drives the line high when the signal is true.

source: specifies the signal that drives the trigger line as specified by the following constants:

`IMG_TRIG_DRIVE_DISABLED`

`IMG_TRIG_DRIVE_AQ_IN_PROGRESS`

`IMG_TRIG_DRIVE_AQ_DONE`

`IMG_TRIG_DRIVE_ASSERTED`

`IMG_TRIG_DRIVE_UNASSERTED`

`IMG_TRIG_DRIVE_HSYNC`

`IMG_TRIG_DRIVE_VSYNC`

`IMG_TRIG_DRIVE_FRAME_START`

IMG_TRIG_DRIVE_FRAME_DONE

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

imgSessionTriggerRead

Usage

```
rval imgSessionTriggerRead(SESSION_ID sid, uInt32 trigger, uInt32 polarity,  
uInt32* status);
```

Purpose

Reads the current value of the specified trigger line.



Note This function is obsolete. The replacement function is [imgSessionTriggerRead2](#), which incorporates the functionality of imgSessionTriggerRead but also returns additional information.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
trigger	UInt32	input
polarity	UInt32	input
status	UInt32*	output
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

trigger: trigger line to read. **trigger** can be one of the following constants:

[IMG_EXT_TRIG0](#) [IMG_EXT_RTSI2](#)

[IMG_EXT_TRIG1](#) [IMG_EXT_RTSI3](#)

[IMG_EXT_TRIG2](#) [IMG_EXT_RTSI4](#)

[IMG_EXT_TRIG3](#) [IMG_EXT_RTSI5](#)

[IMG_EXT_RTSI0](#) [IMG_EXT_RTSI6](#)

[IMG_EXT_RTSI1](#)

 **Note** `IMG_EXT_TRIG <0..3>` refers to the external trigger lines of the image acquisition device. `IMG_EXT_RTSI <0..6>` refers to the internal pins on the RTSI controller of the image acquisition device.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEL` Asserts the trigger line when the signal is low.

`IMG_TRIG_POLAR_ACTIVEH` Asserts the trigger line when the signal is high.

status: pointer to a variable to receive the state of the trigger. Upon return, the function sets **status** to a non-zero value if the trigger is asserted and to a zero value if the trigger is unasserted.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

Branch Offices

Office	Telephone Number
Australia	1800 300 800
Austria	43 662 457990-0
Belgium	32 (0) 2 757 0020
Brazil	55 11 3262 3599
Canada	800 433 3488
China	86 21 5050 9800
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	358 (0) 9 725 72511
France	33 (0) 1 57 66 24 24
Germany	49 89 7413130
India	91 80 41190000
Israel	972 0 3 6393737
Italy	39 02 41309277
Japan	0120-527196 / 81 3 5472 2970
Korea	82 02 3451 3400
Lebanon	961 (0) 1 33 28 28
Malaysia	1800 887710
Mexico	01 800 010 0793
Netherlands	31 (0) 348 433 466
New Zealand	0800 553 322
Norway	47 (0) 66 90 76 60
Poland	48 22 3390150
Portugal	351 210 311 210
Russia	7 495 783 6851
Singapore	1800 226 5886
Slovenia	386 3 425 42 00

South Africa	27 0 11 805 8197
Spain	34 91 640 0085
Sweden	46 (0) 8 587 895 00
Switzerland	41 56 2005151
Taiwan	886 02 2377 2222
Thailand	662 278 6777
Turkey	90 212 279 3031
United Kingdom	44 (0) 1635 523545
United States (Corporate)	512 683 0100

Obsolete Functions

Obsolete functions are functions from a previous version of NI-IMAQ that have been replaced by newer functions. Though the current version of NI-IMAQ still supports these functions, you should use the newer functions whenever possible.

Obsolete Function Panels

The following table lists the Obsolete functions. The first column contains the name of the class. The second column contains names of individual function panels. Each Obsolete function panel represents one function.

Class	Function Name
Obsolete	imgPulseCreate
Obsolete	imgSessionLineTrigSource
Obsolete	imgSessionTriggerConfigure
Obsolete	imgSessionTriggerDrive
Obsolete	imgSessionTriggerRead
Obsolete	imgTriggerRoute
Obsolete	imgSessionWaitSignal
Obsolete	imgSessionWaitSignalAsync

imgSessionLineTrigSource

Usage

```
rval imgSessionLineTrigSource(SESSION_ID sid, uInt32 trigger, uInt32  
polarity, uInt32 skipCount);
```

Purpose

Configures triggering per line for acquisition from a line scan camera. Use this function to require a trigger to start the acquisition of each line from a line scan camera.

-  **Note** This function is obsolete. The replacement function is [imgSessionLineTrigSource2](#), which incorporates the functionality of imgSessionLineTrigSource but also returns additional information.
-  **Note** This function requires a camera that has the functionality to externally control the line rate. To implement this functionality, the external line rate signal must be cabled to one of the image acquisition control lines.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
trigger	ulnt32	input
polarity	ulnt32	input
skipCount	ulnt32	input
rval	Int32	output

Parameter Discussion

sid: valid SESSION_ID.

trigger: trigger line to use. **trigger** can be one of the following constants:

[IMG_EXT_TRIG0](#) [IMG_EXT_RTSI2](#)

[IMG_EXT_TRIG1](#) [IMG_EXT_RTSI3](#)

[IMG_EXT_TRIG2](#) [IMG_EXT_RTSI4](#)

[IMG_EXT_TRIG3](#) [IMG_EXT_RTSI5](#)

[IMG_EXT_RTSI0](#) [IMG_EXT_RTSI6](#)

[IMG_EXT_RTSI1](#)

 **Note** `IMG_EXT_TRIG <0..3>` refers to the external trigger lines of the image acquisition device. `IMG_EXT_RTSI <0..6>` refers to the internal pins on the RTSI controller of the image acquisition device.

polarity: polarity of the trigger line. **polarity** can be one of the following constants:

`IMG_TRIG_POLAR_ACTIVEL` Triggers on a falling edge.

`IMG_TRIG_POLAR_ACTIVEH` Triggers on a rising edge.

skipCount: number of triggers to skip before acquiring a new line. For example, if you are using an encoder to trigger lines and it outputs 1,000 ticks per revolution, but you want to acquire only 10 lines per revolution, set this parameter to 99. Set this parameter to 0 to acquire a line on every trigger.

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.

imgSessionTriggerRoute

Usage

```
rval imgSessionTriggerRoute(SESSION_ID sid, uInt32 srcTrig, uInt32 dstTrig);
```

Purpose

Drives the destination trigger line with the signal on the source trigger line.



Note This function is obsolete. The replacement function is [imgSessionTriggerRoute2](#), which incorporates the functionality of imgSessionTriggerRoute but also returns additional information.

Parameters

Name	Type	Direction
sid	SESSION_ID	input
srcTrig	UInt32	input
dstTrig	UInt32	input

Parameter Discussion

sid: valid SESSION_ID.

srcTrig: source trigger line. **srcTrig** can be one of the following constants:

IMG_EXT_TRIG0 IMG_EXT_TRIG1

IMG_EXT_TRIG2 IMG_EXT_TRIG3

IMG_EXT_RTSI0 IMG_EXT_RTSI1

IMG_EXT_RTSI2 IMG_EXT_RTSI3

IMG_EXT_RTSI4 IMG_EXT_RTSI5

IMG_EXT_RTSI6

dstTrig: destination trigger line. **dstTrig** can be one of the following constants:

IMG_EXT_TRIG0 IMG_EXT_TRIG1

IMG_EXT_TRIG2 IMG_EXT_TRIG3

IMG_EXT_RTSI0 IMG_EXT_RTSI1

IMG_EXT_RTSI2 IMG_EXT_RTSI3

IMG_EXT_RTSI4 IMG_EXT_RTSI5

IMG_EXT_RTSI6

Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgShowError](#).

For more information, refer to the [Obsolete](#) functions topic.