

# IMAQ™

---

## NI-IMAQ™ Function Reference Manual

## **Worldwide Technical Support and Product Information**

[www.natinst.com](http://www.natinst.com)

## **National Instruments Corporate Headquarters**

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

## **Worldwide Offices**

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,  
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, China 0755 3904939, Denmark 45 76 26 00,  
Finland 09 725 725 11, France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,  
India 91805275406, Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456,  
Mexico (D.F.) 5 280 7625, Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, Norway 32 27 73 00,  
Singapore 2265886, Spain (Madrid) 91 640 0085, Spain (Barcelona) 93 582 0251, Sweden 08 587 895 00,  
Switzerland 056 200 51 51, Taiwan 02 2377 1200, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to [techpubs@natinst.com](mailto:techpubs@natinst.com).

© Copyright 1996, 1999 National Instruments Corporation. All rights reserved.

# Important Information

---

## 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 THEREOF 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.

## Trademarks

ComponentWorks™, CVI™, IMAQ™, LabVIEW™, natinst.com™, NI-DAQ™, NI-IMAQ™, PXI™, RTSI™, SCXI™, and StillColor™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

## WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

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. Applications of National Instruments products involving medical or clinical treatment can create a potential for death or bodily injury caused by product failure, or by errors on the part of the user or application designer. 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.

# Contents

---

## About This Manual

Conventions .....	ix
National Instruments Documentation .....	x
Related Documentation.....	x

## Chapter 1

### Introduction

Status Codes.....	1-1
Variable Data Types .....	1-1
Primary Types .....	1-1
Arrays .....	1-2
Programming Language Considerations.....	1-2
LabVIEW .....	1-2
LabWindows/CVI.....	1-2
Other Programming Environments.....	1-6
Code Examples.....	1-6
Geometric Definitions.....	1-6
Architecture .....	1-8

## Chapter 2

### Generic Functions

imgInterfaceOpen .....	2-2
imgSessionOpen .....	2-3
imgClose .....	2-4

## Chapter 3

### High-Level Functions

Snap Functions.....	3-1
imgSnap.....	3-2
imgSnapArea .....	3-3
Grab Functions.....	3-5
imgGrabSetup.....	3-6
imgGrab.....	3-7
imgGrabArea .....	3-8
Ring and Sequence Functions.....	3-10
imgRingSetup .....	3-11
imgSequenceSetup .....	3-12

imgSessionStartAcquisition .....	3-14
imgSessionStopAcquisition .....	3-15
Signal I/O Functions .....	3-16
imgSessionTriggerConfigure .....	3-17
imgSessionLineTrigSource .....	3-19
imgSessionTriggerClear .....	3-20
imgSessionTriggerDrive .....	3-21
imgSessionTriggerRead .....	3-23
imgSessionWaitSignal .....	3-24
imgSessionWaitSignalAsync .....	3-26
imgPulseCreate .....	3-28
imgPulseDispose .....	3-31
imgPulseRate .....	3-32
imgPulseStart .....	3-33
imgPulseStop .....	3-34
Miscellaneous Functions .....	3-35
imgSessionStatus .....	3-36
imgSessionSetROI .....	3-37
imgSessionGetROI .....	3-39
imgSessionGetBufferSize .....	3-40

## Chapter 4

### Low-Level Functions

Acquisition Functions .....	4-1
imgMemLock .....	4-2
imgMemUnlock .....	4-3
imgSessionAbort .....	4-4
imgSessionAcquire .....	4-5
imgSessionConfigure .....	4-6
imgSessionCopyArea .....	4-7
imgSessionCopyBuffer .....	4-9
imgSessionExamineBuffer .....	4-10
imgSessionReleaseBuffer .....	4-12
Attribute Functions .....	4-13
imgGetAttribute .....	4-14
imgGetCameraAttributeNumeric .....	4-15
imgGetCameraAttributeString .....	4-16
imgSessionGetLostFramesList .....	4-17
imgSessionSetUserLUT8bits .....	4-18
imgSessionSetUserLUT16bits .....	4-19
imgSetAttribute .....	4-20
imgSetCameraAttributeNumeric .....	4-21
imgSetCameraAttributeString .....	4-22

Buffer Management Functions .....	4-23
imgCreateBuffer .....	4-24
imgCreateBufList .....	4-26
imgDisposeBuffer.....	4-27
imgDisposeBufList.....	4-28
imgGetBufferElement .....	4-29
imgSessionClearBuffer.....	4-31
imgSetBufferElement .....	4-32
Interface Functions .....	4-34
imgInterfaceQueryNames.....	4-35
imgInterfaceReset.....	4-36
Utility Functions .....	4-37
imgPlot .....	4-38
imgSessionSaveBufferEx .....	4-40
imgShowError .....	4-42

## Appendix A

### Attributes and Constants

NI-IMAQ Attributes .....	A-1
Constants.....	A-7

## Appendix B

### Status Codes

## Appendix C

### Technical Support Resources

## Glossary

## Index

## Figures

Figure 1-1.	Geometric Relationship .....	1-7
Figure 1-2.	NI-IMAQ Architecture .....	1-8

## Tables

Table 1-1.	Primary Type Names .....	1-1
Table 1-2.	The LabWindows/CVI Function Tree for Image Acquisition.....	1-3
Table A-1.	Attribute Summary.....	A-1
Table A-2.	Constants Summary .....	A-7
Table B-1.	Status Code Summary.....	B-1

# About This Manual

---

The *NI-IMAQ Function Reference Manual* is for users of the NI-IMAQ software for PCI bus computers. NI-IMAQ is a powerful application programming interface (API) between your image acquisition (IMAQ) application and the National Instruments IMAQ board for PCI bus computers.

## Conventions

---

The following conventions appear in this manual:

» 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.



This icon denotes a note, which alerts you to important information.

**bold**

Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

*italic*

Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font 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, and code excerpts.



# National Instruments Documentation

---

The *NI-IMAQ Function Reference Manual* is one piece of the documentation set for your IMAQ system. You could have any of several types of manuals, depending on the hardware and software in your system. Use the different types of manuals you have as follows:

- Your IMAQ hardware documentation—These documents have detailed information about the IMAQ hardware that plugs into or is connected to your computer. Use these manuals for hardware installation and configuration instructions, hardware specification information, and application hints.
- Software documentation—Examples of software documentation you might have are the LabVIEW and LabWindows/CVI documentation, the IMAQ Vision documentation, and the NI-IMAQ documentation. After you have set up your hardware system, use either the application software (LabVIEW or LabWindows/CVI) or the NI-IMAQ documentation to help you write your application. If you have a large and complicated system, it is worthwhile to look through the software documentation before you configure your hardware.
- Accessory installation guide or manuals—If you are using accessory products, read the installation guides. They explain how to physically connect the relevant pieces of the system. Consult these guides when you are making your connections.

## Related Documentation

---

The following documents contain information that you might find helpful as you read this manual:

- Your computer technical reference manual

# Introduction

This chapter contains important information about how to apply the function descriptions in this manual to your programming language and environment.

## Status Codes

Every NI-IMAQ function is of the following form:

**rval** = Function\_Name (parameter 1, parameter 2, ... parameter *n*)

where  $n > 0$ . Each function returns a status code (**rval**) that indicates the success or failure of the function, as discussed in Appendix B, [Status Codes](#).

## Variable Data Types

The NI-IMAQ application programming interface (API) is almost identical in Windows 98/95 and Windows NT, except for some of the parameter data types in each of the environments. LabWindows/CVI uses the same data types as Windows NT/98/95. The following sections describe the notation used in those parameter tables and throughout the manual for variable data types.

## Primary Types

Table 1-1 shows the primary type names and their ranges.

**Table 1-1.** Primary Type Names

Type Name	Description	Range	Type
Int8	8-bit ASCII character	0 to 127, -128 to 0	char
uInt8	8-bit ASCII character	0 to 255	char
Int16	16-bit signed integer	-32,768 to 32,767	short

**Table 1-1.** Primary Type Names (Continued)

Type Name	Description	Range	Type
uInt16	16-bit unsigned integer	0 to 65,535	unsigned short
Int32	32-bit signed integer	-2,147,483,648 to 2,147,483,647	long
uInt32	32-bit unsigned integer	0 to 4,294,967,295	unsigned long

## Arrays

When a primary type is inside square brackets (for example, [Int16]) an array of the type named is required for that parameter.

## 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 your source code.

## LabVIEW

For information on how to use LabVIEW VIs with your IMAQ system, refer to the *NI-IMAQ VI Reference Manual*.

## LabWindows/CVI

Inside the LabWindows/CVI environment, the NI-IMAQ functions appear in **Libraries»NI-IMAQ Library**. Each function panel represents an NI-IMAQ function, which is displayed at the bottom of the panel.

Table 1-2 shows how the LabWindows/CVI function panel tree is organized, and the NI-IMAQ function name that corresponds to each function panel.

**Table 1-2.** The LabWindows/CVI Function Tree for Image Acquisition

<b>LabWindows/CVI Function Panel</b>	<b>NI-IMAQ Function</b>
<b>Generic Functions</b>	
Close Object	imgClose
Interface Open	imgInterfaceOpen
Session Open	imgSessionOpen
<b>High-Level Snap Functions</b>	
Snap	imgSnap
Snap Area	imgSnapArea
<b>High-Level Grab Functions</b>	
Grab	imgGrab
Grab Area	imgGrabArea
Grab Setup	imgGrabSetup
<b>High-Level Ring and Sequence Functions</b>	
Ring Setup	imgRingSetup
Sequence Setup	imgSequenceSetup
Session Start Acquisition	imgSessionStartAcquisition
Session Stop Acquisition	imgSessionStopAcquisition
<b>High-Level Signal I/O Functions</b>	
Line Trigger Source	imgSessionLineTrigSource
Pulse Create	imgPulseCreate
Pulse Dispose	imgPulseDispose
Pulse Rate	imgPulseRate

**Table 1-2.** The LabWindows/CVI Function Tree for Image Acquisition (Continued)

<b>LabWindows/CVI Function Panel</b>	<b>NI-IMAQ Function</b>
Pulse Start	imgPulseStart
Pulse Stop	imgPulseStop
Trigger Configure	imgSessionTriggerConfigure
Trigger Clear	imgSessionTriggerClear
Trigger Drive	imgSessionTriggerDrive
Trigger Read	imgSessionTriggerRead
Wait Signal	imgSessionWaitSignal
Wait Signal Asynchronous	imgSessionWaitSignalAsync
<b>High-Level Miscellaneous Functions</b>	
Session Get Buffer Size	imgSessionGetBufferSize
Session Get ROI	imgSessionGetROI
Session Set ROI	imgSessionSetROI
Session Status	imgSessionStatus
<b>Low-Level Acquisition Functions</b>	
Lock Buffer List Memory	imgMemLock
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
Unlock Buffer List Memory	imgMemUnlock

**Table 1-2.** The LabWindows/CVI Function Tree for Image Acquisition (Continued)

<b>LabWindows/CVI Function Panel</b>	<b>NI-IMAQ Function</b>
<b>Low-Level Attribute Functions</b>	
Get Attribute	imgGetAttribute
Get Camera Attribute Numeric	imgGetCameraAttributeNumeric
Get Camera Attribute String	imgGetCameraAttributeString
Get Lost Frames List	imgSessionGetLostFramesList
Set Attribute	imgSetAttribute
Set Camera Attribute Numeric	imgSetCameraAttributeNumeric
Set Camera Attribute String	imgSetCameraAttributeString
Set User LUT 8 bits	imgSessionSetUserLUT8bits
Set User LUT 16 bits	ImgSessionSetUserLUT16bits
<b>Low-Level Buffer Management Functions</b>	
Create Buffer	imgCreateBuffer
Create Buffer List	imgCreateBufList
Dispose Buffer	imgDisposeBuffer
Dispose Buffer List	imgDisposeBufList
Get Buffer Element	imgGetBufferElement
Session Clear Buffer	imgSessionClearBuffer
Set Array Pointer Value	imgSetArrayPointerValue
Set Buffer Element	imgSetBufferElement
<b>Low-Level Interface Functions</b>	
Interface Lock	imgInterfaceLock
Interface Query Names	imgInterfaceQueryNames
Interface Reset	imgInterfaceReset
Interface Unlock	imgInterfaceUnlock

**Table 1-2.** The LabWindows/CVI Function Tree for Image Acquisition (Continued)

LabWindows/CVI Function Panel	NI-IMAQ Function
<b>Low-Level Utility Functions</b>	
Plot Buffer to Window	imgPlot
Session Save Buffer	imgSessionSaveBufferEx
Show Error	imgShowError

## Other Programming Environments

For information on using other programming languages, such as Microsoft Visual C++, with your IMAQ system, refer to the *NI-IMAQ User Manual*.

## Code Examples

You can find code examples in the same directory in which you installed the NI-IMAQ driver software. You can find source code common to all environments in the `Samples` default subfolder.

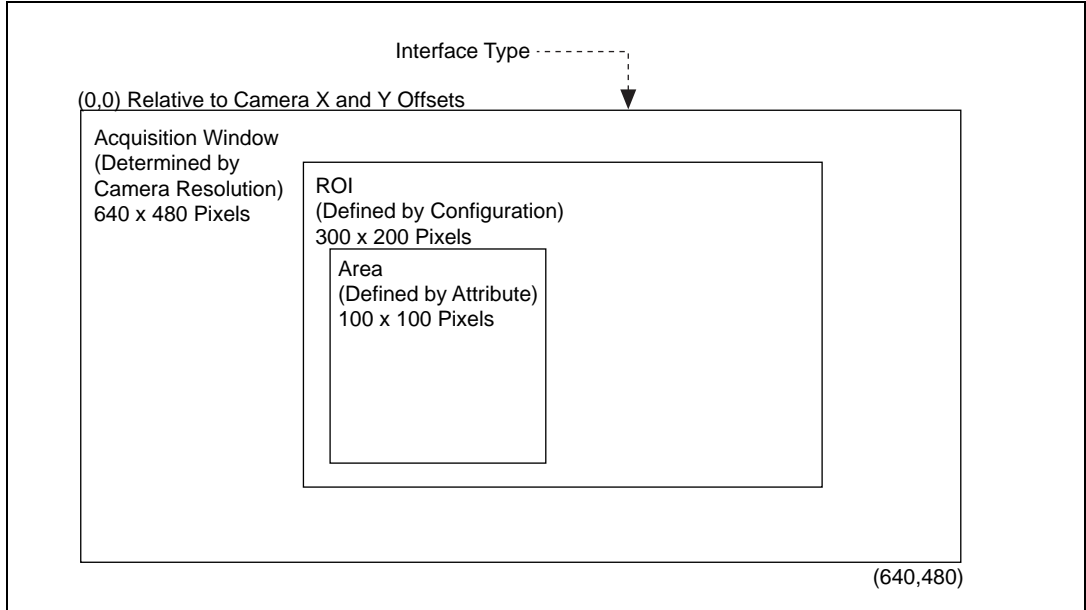
## Geometric Definitions

---

Here are a few definitions you should be familiar with when performing image acquisition tasks:

- An *acquisition window* is the image size specific to a video standard or camera resolution. The default is 640 by 480 pixels. The window's starting position (0,0) varies according to camera.
- A *region of interest (ROI)* is a hardware-programmable rectangular portion of the acquisition window. This is a specific area of the image to acquire.
- An *area* is a rectangular portion of an ROI that software defines and controls.

Figure 1-1 illustrates the geometric relationship of these terms.

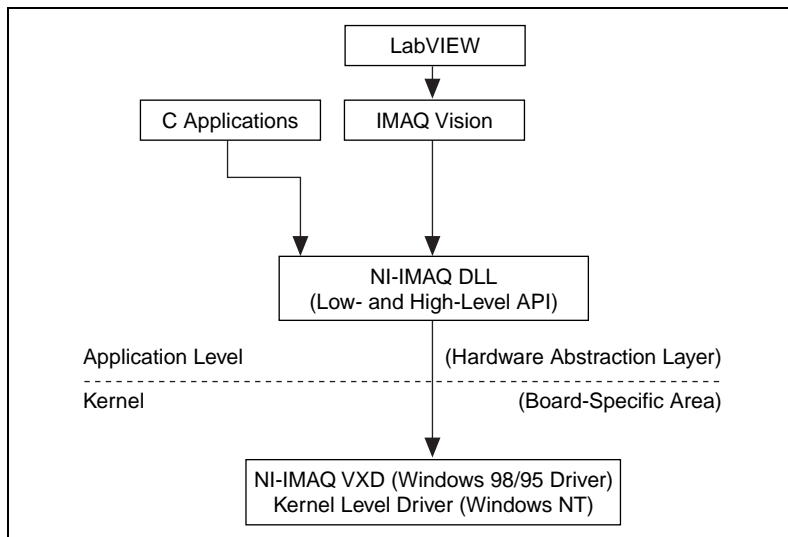


**Figure 1-1.** Geometric Relationship



# Architecture

A block diagram of the NI-IMAQ architecture shown in Figure 1-2 illustrates the low- and mid-level architecture for IMAQ devices.



**Figure 1-2.** NI-IMAQ Architecture

The architecture uses a *hardware abstraction layer*, which separates software API capabilities, such as general acquisition and control functions, from hardware-specific information. This layer lets you use new IMAQ hardware without having to recompile your applications.

---

# Generic Functions

This chapter contains a detailed explanation of each generic NI-IMAQ function. The functions are arranged according to the order in which you will use them.

Generic functions include `imgInterfaceOpen`, `imgSessionOpen`, and `imgClose`. You will use these functions in combination with both high- and low-level functions. These functions set up your interface and session, and close both when you are finished with your application.

## imgInterfaceOpen

---

### Format

**rval** = imgInterfaceOpen(**char\*** interface\_name, **INTERFACE\_ID\*** pifid)

### Purpose

Opens by name an interface as specified in the Measurement & Automation Explorer for IMAQ. If it is successful, this function returns an INTERFACE\_ID.

### Parameters

Name	Type	Direction	Description
<b>interface_name</b>	char*	input	null-terminated string containing the name of interface to open
<b>pifid</b>	INTERFACE_ID*	output	pointer to INTERFACE_ID type variable
<b>rval</b>	Int32	output	status

### Parameter Discussion

**interface\_name** needs a null-terminated string that is the name of the interface to open, such as img0, img1, and so on.

**pifid** passes a pointer to an area of memory reserved as an INTERFACE\_ID type variable. If the function succeeds, the variable will contain a valid INTERFACE\_ID that can be used in subsequent functions.

**rval** returns the following status codes:

IMG_ERR_BINT	bad interface
IMG_ERR_GOOD	no error occurred
IMG_ERR_OVRN	too many interfaces open
IMG_ERR_PAR1	null pointer
IMG_ERR_PAR2	null pointer



**Note** You can use imgInterfaceQueryNames to retrieve a valid list of interface names.

## imgSessionOpen

---

### Format

**rval** = imgSessionOpen(**INTERFACE\_ID ifid**, **SESSION\_ID\* psid**)

### Purpose

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

### Parameters

Name	Type	Direction	Description
<b>ifid</b>	INTERFACE_ID	input	interface ID to open session
<b>psid</b>	SESSION_ID*	output	pointer to a session ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**ifid** is a valid INTERFACE\_ID type variable.

**psid** passes a pointer to an area of memory reserved for a SESSION\_ID type variable. If the function succeeds, the variable will contain a valid SESSION\_ID that can be used in subsequent functions.

**rval** returns the following status codes:

IMG_ERR_BCMF	bad camera file (check syntax)
IMG_ERR_GOOD	no error occurred
IMG_ERR_OVRN	too many interfaces open
IMG_ERR_PAR1	invalid INTERFACE_ID
IMG_ERR_PAR2	null pointer

## imgClose

---

### Format

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

### Purpose

Closes a session or interface and unlocks and releases all buffers associated with the data type.

### Parameters

Name	Type	Direction	Description
<b>void_id</b>	uInt32	input	session or interface ID
<b>freeResources</b>	uInt32	input	cleanup flag
<b>rval</b>	Int32	output	status

### Parameter Discussion

**void\_id** is a valid SESSION\_ID or INTERFACE\_ID type variable.

**freeResources** is the cleanup flag. If **freeResources** is TRUE, it indicates that all buffers and buffer lists associated with the session are to be released. If **freeResources** is FALSE, it indicates no buffer cleanup should be performed.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid INTERFACE_ID or SESSION_ID



**Note** Closing an interface closes all sessions attached to that interface.

---

# High-Level Functions

This chapter contains a detailed explanation of each high-level NI-IMAQ function. The functions are arranged according to the category of image acquisition procedure—snap, grab, ring and sequence functions, signal input/output (I/O) functions, or miscellaneous functions—and then the order in which you will use them.

Using high-level functions, you can easily perform such functions as acquiring images in single-shot (snap) or continuous (ring) mode without advanced knowledge of the NI-IMAQ low-level function calls and image acquisition details.

## Snap Functions

---

Snap functions include `imgSnap` and `imgSnapArea`. You can use these functions to acquire a single image after opening a valid session, using `imgInterfaceOpen` and `imgSessionOpen` to obtain a valid `SESSION_ID`.

## imgSnap

---

### Format

**rval** = imgSnap(**SESSION\_ID** **sid**, **void\*** **bufAddr**)

### Purpose

Performs a single frame or field acquisition. This function uses the following attributes to perform an image acquisition:

IMG_ATTR_ACQWINDOW_LEFT	IMG_ATTR_ROI_TOP
IMG_ATTR_ACQWINDOW_WIDTH	IMG_ATTR_ROI_HEIGHT
IMG_ATTR_ACQWINDOW_TOP	IMG_ATTR_ROI_WIDTH
IMG_ATTR_ACQWINDOW_HEIGHT	IMG_ATTR_ROWBYTES
IMG_ATTR_ROI_LEFT	IMG_ATTR_YOFF_BUFFER

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>bufAddr</b>	void*	input	pointer to the buffer address
<b>rval</b>	Int32	output	no error

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**bufAddr** points to an area of memory in which to store the image. If **bufAddr** points to a NULL pointer, this call will allocate an appropriate size buffer and return the buffer address in the location specified by **bufAddr**.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSnapArea

---

### Format

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

### Purpose

Performs an area-specific frame or field acquisition. This function does not modify any attributes.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>bufAddr</b>	void*	input	pointer to the buffer address
<b>top</b>	uInt32	input	top ordinate of the first pixel transferred
<b>left</b>	uInt32	input	left ordinate of the first pixel transferred
<b>height</b>	uInt32	input	height of rectangle to transfer
<b>width</b>	uInt32	input	width of the rectangle to transfer
<b>rowPixels</b>	uInt32	input	used in calculating the address of the next line
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**bufAddr** points to an area of memory in which to store the image. If **bufAddr** points to a NULL pointer, this call will allocate an appropriate size buffer and return the buffer address in the location specified by **bufAddr**.

**top** indicates the top vertical offset of the first pixel transferred.

**left** indicates the left horizontal offset of the first pixel transferred.

**height** indicates the height of area to transfer.



**width** indicates the width of the area to transfer.

**rowPixels** indicates the exact pixel-width of the horizontal line to acquire. This parameter specifies the number of pixel to add to the line pointer for the next scan line. This value must be greater than or equal to the width parameter. Passing a zero for this value causes it to be ignored.

**rval** returns `IMG_ERR_GOOD` if no error occurs.

# Grab Functions

---

Grab functions include `imgGrabSetup`, `imgGrab`, and `imgGrabArea`. You can use the grab functions to perform a continuous acquisition.

To use the grab functions, you must first call `imgGrabSetup` to configure the session for grabbing and optionally start the acquisition process. If you do not start the acquisition via `imgGrabSetup`, you must start it by calling `imgSessionStartAcquisition` prior to calling the `imgGrab` and `imgGrabArea` functions. After the acquisition has started, you obtain an image copy by calling the `imgGrab` and `imgGrabArea`. To stop the acquisition, call `imgSessionStopAcquisition`.

## imgGrabSetup

---

### Format

**rval** = `imgGrabSetup(SESSION_ID sid, uInt32 startNow)`

### Purpose

Configures and optionally starts a continuous acquisition. This function uses the following attributes to perform an image acquisition:

IMG_ATTR_ACQWINDOW_LEFT	IMG_ATTR_ROI_TOP
IMG_ATTR_ACQWINDOW_WIDTH	IMG_ATTR_ROI_HEIGHT
IMG_ATTR_ACQWINDOW_TOP	IMG_ATTR_ROI_WIDTH
IMG_ATTR_ACQWINDOW_HEIGHT	IMG_ATTR_ROWBYTES
IMG_ATTR_ROI_LEFT	IMG_ATTR_YOFF_BUFFER

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>startNow</b>	uInt32	input	start acquisition after setup completes
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**startNow** starts a grab acquisition after setup has been completed. A non-zero value here specifies that the continuous acquisition should start immediately. If the value is zero, start the grab acquisition with `imgSessionStartAcquisition`.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

# imgGrab

---

## Format

**rval** = imgGrab(**SESSION\_ID** sid, void\* bufAddr, uInt32 syncOnVB)

## Purpose

Performs a transfer from a continuous acquisition session. Call this function only after calling imgGrabSetup. This function uses the following attributes:

IMG_ATTR_ROI_LEFT	IMG_ATTR_ROI_WIDTH
IMG_ATTR_ROI_TOP	IMG_ATTR_ROWBYTES
IMG_ATTR_ROI_HEIGHT	IMG_ATTR_YOFF_BUFFER

## Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>bufAddr</b>	void*	input	pointer to the buffer address
<b>syncOnVB</b>	uInt32	input	vertical blank flag
<b>rval</b>	Int32	output	status

## Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**bufAddr** is a pointer to the buffer address. If **bufAddr** points to a NULL pointer, this call will allocate an appropriate size buffer and return the buffer address in the location specified by **bufAddr**.

**syncOnVB** indicates a wait for a vertical blank. If this parameter is TRUE, the transfer will be done according to and using the video synchronization. Using this option avoids mixing two different time bases within the same video field. If **syncOnVB** is FALSE, the transfer is done without considering the video synchronization.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgGrabArea

---

### Format

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

### Purpose

Performs a transfer from a continuous acquisition using the given parameters. This function does not modify any attributes. Call this function only after calling `imgGrabSetup`.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>bufAddr</b>	void*	input	pointer to the buffer address
<b>syncOnVB</b>	uInt32	input	vertical blank flag
<b>top</b>	uInt32	input	top ordinate of the first pixel transferred
<b>left</b>	uInt32	input	left ordinate of the first pixel transferred
<b>height</b>	uInt32	input	height of rectangle to transfer
<b>width</b>	uInt32	input	width of the rectangle to transfer
<b>rowPixels</b>	uInt32	input	used in calculating the address of the next line
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**bufAddr** points to an area of memory in which to store the image. If **bufAddr** points to a NULL pointer, this call will allocate an appropriate size buffer and return the buffer address in the location specified by **bufAddr**.

**syncOnVB** indicates a wait for a vertical blank. If **syncOnVB** is TRUE, the transfer will be done according to and using the video synchronization. Using this option avoids mixing two different time bases within the same video field. If **syncOnVB** is FALSE, the transfer is done without considering the video synchronization.

**top** indicates the top vertical offset of the first pixel transferred.

**left** indicates the left horizontal offset of the first pixel transferred.

**height** indicates the height of area to transfer.

**width** indicates the width of the area to transfer.

**rowPixels** indicates the exact pixel-width of the horizontal line to acquire. This parameter specifies the number of pixels to add to the line pointer for the next scan line. This value must be greater than or equal to the width parameter. Passing a zero for this value causes it to be ignored. This parameter is used for byte alignment or if the image buffer contains a border for image processing.

**rval** returns `IMG_ERR_GOOD` if no error occurs.

## Ring and Sequence Functions

---

Ring and sequence functions include `imgRingSetup`, `imgSequenceSetup`, `imgSessionStartAcquisition`, and `imgSessionStopAcquisition`. You can use these functions to perform a continuous acquisition that loops or stops after a certain number of images have been captured.

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 via `imgRingSetup` or `imgSequenceSetup`, you must call `imgSessionStartAcquisition` to start it.

## imgRingSetup

---

### Format

```
rval = imgRingSetup(SESSION_ID sid, uInt32 numberBuffer, void* bufferList[],
                    uInt32 skipCount, uInt32 startnow)
```

### Purpose

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

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>numberBuffer</b>	uInt32	input	number of buffers used in the ring session
<b>bufferList[]</b>	void*	input	array of buffer pointers
<b>skipCount</b>	uInt32	input	number of frames or field to skip before each acquisition
<b>startnow</b>	uInt32	input	start acquisition after setup completes
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**numberBuffer** indicates the number of buffers in the buffer list.

**bufferList[]** is an array of buffer pointers. For each element in the buffer list that is initialized to NULL, **bufferList[]** will allocate a buffer and return this buffer address in the array element. If **buffer[0]** contains a NULL pointer, this call allocates the number of buffers required and returns the buffer addresses in **bufferList[]**.

**skipCount** indicates the number of frames or field to skip before each acquisition. This number is the same for all acquisitions.

**startnow** starts a grab acquisition after setup has been completed. A non-zero value for **startnow** specifies that the continuous acquisition should start immediately. If **startnow** is zero, start the grab acquisition with `imgSessionStartAcquisition`.

**rval** returns IMG\_ERR\_GOOD if no error occurs.



## imgSequenceSetup

---

### Format

```
rval = imgSequenceSetup(SESSION_ID sid, uInt32 numberBuffer, void* bufferList[],
                        uInt32 skipCount[], uInt32 startnow, uInt32 async)
```

### Purpose

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

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>numberBuffer</b>	uInt32	input	number of buffers used in the ring session
<b>bufferList[]</b>	void*	input	array of buffer pointers
<b>skipCount[]</b>	uInt32	input	array containing the number of frames or field to skip before each acquisition
<b>startNow</b>	uInt32	input	start acquisition after setup completes
<b>async</b>	uInt32	input	indicates either a synchronous or an asynchronous acquisition
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**numberBuffer** indicates the number of buffers in the buflist.

**bufferList[]** is an array of buffer pointers. For each element in the buffer list that is initialized to NULL, **bufferList[]** will allocate a buffer and return this buffer address in the array element.

**skipCount[]** is an array containing the number of frames or fields to skip before each acquisition.

**startNow** starts a grab acquisition after setup has been completed. A non-zero value for **startNow** specifies that the continuous acquisition should start immediately. If **startNow** is zero, start the grab acquisition with `imgSessionStartAcquisition`.

**async** indicates either an asynchronous or a synchronous acquisition. If **async** is non-zero, it indicates an asynchronous acquisition. If **async** is zero, it indicates a synchronous acquisition. This parameter is only valid if the **startNow** parameter is non-zero.

**rval** returns `IMG_ERR_GOOD` if no error occurs.

## imgSessionStartAcquisition

---

### Format

**rval** = `imgSessionStartAcquisition(SESSION_ID sid)`

### Purpose

Starts a session acquisition identified by **sid**. Use this function with `grab`, `ring`, and `sequence` functions.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionStopAcquisition

---

### Format

**rval** = `imgSessionStopAcquisition(SESSION_ID sid)`

### Purpose

Stops a session acquisition identified by **sid**. Use this function with `grab`, `ring`, and `sequence` functions.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## Signal I/O Functions

---

Signal I/O functions include `imgSessionTriggerConfigure`, `imgSessionLineTrigSource`, `imgSessionTriggerClear`, `imgSessionTriggerDrive`, `imgSessionTriggerRead`, `imgSessionWaitSignal`, `imgSessionWaitSignalAsync`, `imgPulseCreate`, `imgPulseDispose`, `imgPulseRate`, `imgPulseStart`, and `imgPulseStop`.

You can use signal I/O functions to control the trigger lines on IMAQ 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.

## imgSessionTriggerConfigure

---

### Format

```
rval = imgSessionTriggerConfigure(SESSION_ID sid, uInt32 trig_num,
                                  uInt32 trig_polarity, uInt32 time_out,
                                  uInt32 trig_action)
```

### Purpose

Configures an acquisition to start based on an external trigger.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>trig_num</b>	uInt32	input	trigger line on which pulse is generated
<b>trig_polarity</b>	uInt32	input	polarity
<b>time_out</b>	uInt32	input	time to wait for image
<b>trig_action</b>	uInt32	input	start acquisition based on trigger, start each buffer list based on trigger, trigger each buffer
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid Session\_ID type variable.

**trig\_num** is the source of the trigger signal as specified by the constants:

IMG_EXT_TRIG0	IMG_EXT_RTISI2
IMG_EXT_TRIG1	IMG_EXT_RTISI3
IMG_EXT_TRIG2	IMG_EXT_RTISI4
IMG_EXT_TRIG3	IMG_EXT_RTISI5
IMG_EXT_RTISI0	IMG_EXT_RTISI6
IMG_EXT_RTISI1	

**trig\_polarity** is the polarity of the trigger line as defined by the constants:

IMG_TRIG_POLAR_ACTIVEL
IMG_TRIG_POLAR_ACTIVEH

**time\_out** is the amount of time in milliseconds to wait for the trigger to occur and the image to be captured.

**trig\_action** specifies if an assertion edge of **trig\_num** should start an acquisition. Values are:

- IMG\_TRIG\_ACTION\_NONE
- IMG\_TRIG\_ACTION\_CAPTURE
- IMG\_TRIG\_ACTION\_BUFLIST
- IMG\_TRIG\_ACTION\_BUFFER

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionLineTrigSource

---

```
rval = imgSessionLineTrigSource(SESSION_ID sid, uInt32 trig_source,
                                uInt32 trig_polarity, uInt32 skip_number)
```

### 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.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>trig_source</b>	uInt32	input	source of the trigger signal
<b>trig_polarity</b>	uInt32	input	polarity of the trigger line
<b>skip_number</b>	uInt32	input	number of triggers to skip
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**trig\_source** is the source of the trigger signal as specified by the following constants:

IMG_EXT_TRIG0	IMG_EXT_RTISI2
IMG_EXT_TRIG1	IMG_EXT_RTISI3
IMG_EXT_TRIG2	IMG_EXT_RTISI4
IMG_EXT_TRIG3	IMG_EXT_RTISI5
IMG_EXT_RTISI0	IMG_EXT_RTISI6
IMG_EXT_RTISI1	IMG_FIXED_FREQUENCY

**trig\_polarity** is the polarity of the trigger line as defined by the constants:

IMG_TRIG_POLAR_ACTIVEL
IMG_TRIG_POLAR_ACTIVEH

**skip\_number** is the number of triggers to wait 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 **skip\_number** to 99. Set **skip\_number** to 0 to acquire a line on every trigger from the encoder.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
--------------	-------------------



## **imgSessionTriggerClear**

---

### **Format**

**rval** = `imgSessionTriggerClear(SESSION_ID sid)`

### **Purpose**

Disables all triggers on the session.

### **Parameters**

<b>Name</b>	<b>Type</b>	<b>Direction</b>	<b>Description</b>
<b>sid</b>	SESSION_ID	input	session ID
<b>rval</b>	Int32	output	status

### **Parameter Discussion**

**sid** is a valid SESSION\_ID type variable.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionTriggerDrive

---

### Format

```
rval = imgSessionTriggerDrive(SESSION_ID sid, uInt32 trig_num, uInt32 trig_polarity,
                               uInt32 trig_drive)
```

### Purpose

Configures the specified trigger line to drive a signal out.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>trig_num</b>	uInt32	input	trigger line on which pulse is generated
<b>trig_polarity</b>	uInt32	input	polarity
<b>trig_drive</b>	uInt32	input	HIGH, LOW, internal status signals
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**trig\_num** is the trigger line to drive as specified by the 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	

**trig\_polarity** is the polarity of the trigger line as defined by the constants:

IMG_TRIG_POLAR_ACTIVEL
IMG_TRIG_POLAR_ACTIVEH

**trig\_drive** specifies the signal that will drive the trigger line as specified by the 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\_PIXEL\_CLK (not valid on PCI-1424)
- IMG\_TRIG\_DRIVE\_FRAME\_START
- IMG\_TRIG\_DRIVE\_FRAME\_DONE

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionTriggerRead

---

### Format

```
rval = imgSessionTriggerRead(SESSION_ID sid, uInt32 trig_num, uInt32 trig_polarity,
                             uInt32* status)
```

### Purpose

Reads the current value of the specified trigger line.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>trig_num</b>	uInt32	input	trigger line on which pulse is generated
<b>trig_polarity</b>	uInt32	input	polarity
<b>status</b>	uInt32*	output	value of trigger line
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**trig\_num** is the trigger line to read as specified by the 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	

**trig\_polarity** is the polarity of the trigger line as defined by the constants:

IMG_TRIG_POLAR_ACTIVEL
IMG_TRIG_POLAR_ACTIVEH

**status** is a pointer to an area of memory reserved as a trigger status variable. Returns TRUE if the trigger is currently asserted, FALSE if it is unasserted.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionWaitSignal

---

### Format

```
rval = imgSessionWaitSignal(SESSION_ID sid, uInt32 signal, uInt32 signal_pol,
                             uInt32 timeout)
```

### Purpose

Waits for a signal to be asserted. This function will return when the specified signal is asserted.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>signal</b>	uInt32	input	signal which will cause the pulse to be generated
<b>signal_pol</b>	uInt32	input	polarity of signal
<b>timeout</b>	uInt32	input	time to wait for image
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**signal** is the assertion edge of the signal that will cause the function to return as specified by the constants:

IMG_AQ_IN_PROGRESS	IMG_EXT_TRIG3
IMG_AQ_DONE	IMG_EXT_RTSl0
IMG_FRAME_START	IMG_EXT_RTSl1
IMG_FRAME_DONE	IMG_EXT_RTSl2
IMG_BUF_COMPLETE	IMG_EXT_RTSl3
IMG_EXT_TRIG0	IMG_EXT_RTSl4
IMG_EXT_TRIG1	IMG_EXT_RTSl5
IMG_EXT_TRIG2	IMG_EXT_RTSl6

**signal\_pol** is the polarity of the signal input as defined by the constants:

IMG\_TRIG\_POLAR\_ACTIVEL  
IMG\_TRIG\_POLAR\_ACTIVEH



**Note** This input is valued only for external triggers and RTSI lines. It is ignored for all other signals.

**timeout** is the amount of time in milliseconds to wait for the assertion edge of signal.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionWaitSignalAsync

---

### Format

```
rval = imgSessionWaitSignalAsync(SESSION_ID sid, uInt32 signal, uInt32 signal_pol,
                                CALL_BACK_PTR funcptr, void* callback_data)
```

### Purpose

Monitors for a signal to be asserted and invokes a user-defined callback when the signal is asserted.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>signal</b>	uInt32	input	signal which will cause the pulse to be generated
<b>signal_pol</b>	uInt32	input	polarity of signal
<b>funcptr</b>	CALL_BACK_PTR	input	callback function
<b>callback_data</b>	void*	input	four-byte value passed to the callback function
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**signal** is the assertion edge of the signal that will cause the callback function to be invoked as defined by the constants:

IMG_AQ_IN_PROGRESS	IMG_EXT_TRIG3
IMG_AQ_DONE	IMG_EXT_RTSI0
IMG_FRAME_START	IMG_EXT_RTSI1
IMG_FRAME_DONE	IMG_EXT_RTSI2
IMG_BUF_COMPLETE	IMG_EXT_RTSI3
IMG_EXT_TRIG0	IMG_EXT_RTSI4
IMG_EXT_TRIG1	IMG_EXT_RTSI5
IMG_EXT_TRIG2	IMG_EXT_RTSI6

**signal\_pol** is the polarity of the signal input as defined by the constants:

```
IMG_TRIG_POLAR_ACTIVEL
IMG_TRIG_POLAR_ACTIVEH
```



**Note** This input is valid only for external triggers and RTSI lines. It is ignored for all other signals.

**funcptr** is a pointer to the callback function, which is specified by the following function prototype: `uInt32 (*CALLBACK_PTR) (SESSION_ID sid, IMG_ERR err, uInt32 signal, void* userdata)`.

**callback\_data** is a four-byte value that is passed to the callback function. The value can be a pointer to user data.

**rval** returns the following status codes:

<code>IMG_ERR_GOOD</code>	no error occurred
<code>IMG_ERR_BPMD</code>	bad pulse mode
<code>IMG_ERR_EMEM</code>	not enough memory to create status thread
<code>IMG_ERR_OSER</code>	bad thread creation
<code>IMG_ERR_PAR1</code>	parameter 1 error



## imgPulseCreate

---

### Format

```
rval = imgPulseCreate(uInt32 timebase, uInt32 delay, uInt32 width, uInt32 signal_source,
                    uInt32 signal_polarity, uInt32 output, uInt32 output_polarity,
                    uInt32 pulse_mode, 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.

### Parameters

Name	Type	Direction	Description
<b>timebase</b>	uInt32	input	timebase value
<b>delay</b>	uInt32	input	interval before the pulse
<b>width</b>	uInt32	input	interval of the pulse
<b>signal_source</b>	uInt32	input	signal that triggers the pulse generation
<b>signal_polarity</b>	uInt32	input	polarity of the signal
<b>output</b>	uInt32	input	trigger line on which the pulse is generated
<b>output_polarity</b>	uInt32	input	the polarity of the pulse output
<b>pulse_mode</b>	uInt32	input	indicates if the pulse is repeated
<b>plsID</b>	PULSE_ID*	input	pulse ID created and configured with pulse functions
<b>rval</b>	Int32	output	status

## Parameter Discussion

**timebase** selects the timebase, or resolution, to be used by the counter. **timebase** has the following possible values:

```
PULSE_TIMEBASE_50MHZ
PULSE_TIMEBASE_100KHZ
PULSE_TIMEBASE_PIXELCLK
```

**delay** is the desired duration of the first phase of the pulse, phase 1. The unit is cycles of the **timebase**. Use the following formula to determine the actual time period that delay represents:

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

**width** is the desired 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:

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

**signal\_source** specifies the signal that will cause the pulse to be generated. The assertion edge of the following signals can initiate pulse generation, as specified by the following constants:

IMG_IMMEDIATE	IMG_EXT_TRIG3
IMG_AQ_IN_PROGRESS	IMG_EXT_RTISI0
IMG_AQ_DONE	IMG_EXT_RTISI1
IMG_FRAME_START	IMG_EXT_RTISI2
IMG_FRAME_DONE	IMG_EXT_RTISI3
IMG_EXT_TRIG0	IMG_EXT_RTISI4
IMG_EXT_TRIG1	IMG_EXT_RTISI5
IMG_EXT_TRIG2	IMG_EXT_RTISI6

**signal\_polarity** is the polarity of the signal input as defined by the constants:

```
IMG_TRIG_POLAR_ACTIVEL
IMG_TRIG_POLAR_ACTIVEH
```



**Note** This input is valued only for external triggers and RTSI lines. It is ignored for all other signals.

**output** is the trigger line on which the pulse is generated, as specified by the following constants:

IMG_EXT_TRIG0	IMG_EXT_RTISI2
IMG_EXT_TRIG1	IMG_EXT_RTISI3
IMG_EXT_TRIG2	IMG_EXT_RTISI4
IMG_EXT_TRIG2	IMG_EXT_RTISI5
IMG_EXT_RTISI0	IMG_EXT_RTISI6
IMG_EXT_RTISI1	

**output\_polarity** is the polarity of the pulse output as defined by the following constants:

IMG\_PULSE\_POLAR\_ACTIVEL  
 IMG\_PULSE\_POLAR\_ACTIVEH

**pulse\_mode** indicates if the pulse is generated once or continuously, as specified by the following constants:

PULSE\_MODE\_TRAIN  
 PULSE\_MODE\_SINGLE  
 PULSE\_MODE\_SINGLE\_REARM

**plsID** passes a pointer to an area of memory reserved as a PULSE\_ID type variable. If the function succeeds, the variable will contain a valid PULSE\_ID that can be used in subsequent functions.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BPMD	bad pulse mode
IMG_ERR_MXPI	exhausted pulse IDs
IMG_ERR_PAR1	parameter 1 error
IMG_ERR_PAR2	parameter 2 error
IMG_ERR_PAR4	parameter 4 error
IMG_ERR_PAR5	parameter 5 error
IMG_ERR_PAR6	parameter 6 error
IMG_ERR_PAR7	parameter 7 error

## imgPulseDispose

---

### Format

**rval** = `imgPulseDispose(PULSE_ID plsID)`

### Purpose

Disposes of a pulse ID.

### Parameters

Name	Type	Direction	Description
<b>plsID</b>	PULSE_ID	input	pulse ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**plsID** is a valid PULSE\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BPID	bad pulse ID

## imgPulseRate

---

### Format

```
rval = imgPulseRate(uInt32 delaytime, uInt32 widthtime, uInt32* delay, uInt32* width,
                    uInt32* timebase)
```

### Purpose

Converts delay and width into delay, width, and timebase values needed by `imgPulseCreate`.

### Parameters

Name	Type	Direction	Description
<b>delaytime</b>	uInt32	input	interval before the pulse in microseconds
<b>widthtime</b>	uInt32	input	interval of the pulse in microseconds
<b>delay</b>	uInt32*	output	interval before the pulse in cycles
<b>width</b>	uInt32*	output	interval of the pulse in cycles
<b>timebase</b>	uInt32*	output	timebase source
<b>rval</b>	Int32	output	status

### Parameter Discussion

**delaytime** is the desired duration of the first phase of the pulse in microseconds.

**widthtime** is the desired duration of the second phase of the pulse in microseconds.

**delay** represents the number of cycles of timebase of the first phase of the pulse.

**width** represents the number of cycles of timebase of the second phase of the pulse.

**timebase** is a code that represents the timebase on the board that the counter uses to produce the pulse.

**rval** returns `IMG_ERR_GOOD` if no error occurred.

## imgPulseStart

---

**rval** = `imgPulseStart(PULSE_ID pid, SESSION_ID sid)`

### Purpose

Starts the generation of a pulse. You must call `imgPulseCreate` first to configure the pulse.

### Parameters

Name	Type	Direction	Description
<b>pid</b>	PULSE_ID	input	pulse ID
<b>sid</b>	SESSION_ID	input	session ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**pid** is a valid PULSE\_ID type variable.

**sid** is a valid SESSION\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BPID	bad pulse ID
IMG_ERR_PAR2	invalid SESSION_ID

## imgPulseStop

---

**rval** = `imgPulseStop(PULSE_ID pid)`

### Purpose

Stops the generation of a pulse.

### Parameters

Name	Type	Direction	Description
<b>pid</b>	PULSE_ID	input	pulse ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**pid** is a valid PULSE\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BPID	bad pulse ID
IMG_ERR_PLNS	pulse not started error

# Miscellaneous Functions

---

Miscellaneous functions include `imgSessionStatus`, `imgSessionSetROI`, `imgSessionGetROI`, and `imgSessionGetBufferSize`.

These functions obtain status information on a session, get and set a region of interest, and get the buffer size required for a session based on current attributes.



## imgSessionStatus

---

### Format

**rval** = `imgSessionStatus(SESSION_ID sid, uInt32* status, uInt32* bufferNumber)`

### Purpose

Gets the current session status.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>status</b>	uInt32*	output	current session status
<b>bufferNumber</b>	uInt32*	output	pointer to an area of memory reserved for the current buffer number
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**status** indicates the current session status. If **status** is non-zero, the session currently is acquiring. If **status** is zero, the session is idle. This is the value returned by IMG\_ATTR\_ACQ\_IN\_PROGRESS in `imgGetAttribute`.

**bufferNumber** is a pointer to an area of memory reserved for the current buffer number.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionSetROI

---

### Format

**rval** = `imgSessionSetROI(SESSION_ID sid, uInt32 top, uInt32 left, uInt32 height, uInt32 width)`

### Purpose

Sets acquisition region of interest. You would typically make this call after creating a session and before calling the `imgSessionStartAcquisition`. This function modifies the following attributes:

IMG\_ATTR\_ROI\_TOP  
 IMG\_ATTR\_ROI\_LEFT  
 IMG\_ATTR\_ROI\_HEIGHT  
 IMG\_ATTR\_ROI\_WIDTH

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>top</b>	uInt32	output	top ordinate of the first pixel transferred
<b>left</b>	uInt32	output	left ordinate of the first pixel transferred
<b>height</b>	uInt32	output	height of rectangle to transfer
<b>width</b>	uInt32	output	width of the rectangle to transfer
<b>rval</b>	Int32	output	status

## Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**top** indicates the top vertical offset of the first pixel transferred.

**left** indicates the left horizontal offset of the first pixel transferred.

**height** indicates the height of area to transfer.

**width** indicates the width of the area to transfer.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionGetROI

---

### Format

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

### Purpose

Gets acquisition region of interest.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>top</b>	uInt32*	output	current value of IMG_ATTR_ROI_TOP
<b>left</b>	uInt32*	output	current value of IMG_ATTR_ROI_LEFT
<b>height</b>	uInt32*	output	current value of IMG_ATTR_ROI_HEIGHT
<b>width</b>	uInt32*	output	current value of IMG_ATTR_ROI_WIDTH
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**top** indicates the top vertical offset of the first pixel transferred.

**left** indicates the left horizontal offset of the first pixel transferred.

**height** indicates the height of area to transfer.

**width** indicates the width of the area to transfer.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

## imgSessionGetBufferSize

---

### Format

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

### Purpose

Gets the minimum buffer size needed for frame buffer allocation. This function calculates the buffer size by using the following attributes:

IMG\_ATTR\_ROI\_HEIGHT  
 IMG\_ATTR\_ROWBYTES  
 IMG\_ATTR\_YOFF\_BUFFER

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>sizeNeeded</b>	uInt32*	output	buffer size
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**sizeNeeded** returns the buffer size needed for an image based on the attributes listed above.

**rval** returns IMG\_ERR\_GOOD if no error occurs.

---

# Low-Level Functions

This chapter contains a detailed explanation of each low-level NI-IMAQ function. The functions are arranged alphabetically under the type of image acquisition procedure—acquisition, attribute, buffer management, interface, and utility.

Low-level functions let you perform in-depth tasks that require a more advanced understanding of the IMAQ hardware and image acquisition, such as directly controlling video parameters like gain and offset level or locking down a buffer during a continuous acquisition.

## Acquisition Functions

---

Acquisition functions include `imgMemLock`, `imgMemUnlock`, `imgSessionAbort`, `imgSessionAcquire`, `imgSessionConfigure`, `imgSessionCopyArea`, `imgSessionCopyBuffer`, `imgSessionExamineBuffer`, and `imgSessionReleaseBuffer`.

You can use these functions to configure, start, and abort an image acquisition. These functions also let you examine a buffer during acquisition.

## imgMemLock

---

### Format

**rval** = `imgMemLock(BUFLIST_ID bid)`

### Purpose

Locks all session-associated image buffers in memory in preparation for an acquisition. The buffers must be successfully locked down before an acquisition can begin. The function does not do anything if the buffers exist in onboard memory.

### Parameters

Name	Type	Direction	Description
<b>bid</b>	BUFLIST_ID	input	buffer list ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**bid** is a valid BUFLIST\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_BBUF	bad buffer pointer in list
IMG_ERR_ELCK	cannot lock buffers down, no more memory
IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID

## imgMemUnlock

---

### Format

**rval** = `imgMemUnlock(BUFLIST_ID bid)`

### Purpose

Unlocks all session-associated buffers.

### Parameters

Name	Type	Direction	Description
<b>bid</b>	BUFLIST_ID	input	buffer list ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**bid** is a valid BUFLIST\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_BBUF	bad buffer pointer in list
IMG_ERR_ELCK	cannot lock buffers down, no more memory
IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID



## imgSessionAbort

---

### Format

```
rval = imgSessionAbort(SESSION_ID sid, uInt32* buf_num)
```

### Purpose

Stops asynchronous acquisition or synchronous continuous acquisition immediately.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>buf_num</b>	uInt32*	output	pointer to the last valid buffer number
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**buf\_num** points to an area of memory to return the last valid buffer number.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	null pointer

## imgSessionAcquire

---

### Format

**rval** = `imgSessionAcquire`(SESSION\_ID **sid**, uInt32 **async**, CALL\_BACK\_PTR **callback**)

### Purpose

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

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>async</b>	uInt32	input	asynchronous flag
<b>callback</b>	CALL_BACK_PTR	input	user completion routine
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**async** is the asynchronous flag. If **async** is non-zero, it indicates an asynchronous acquisition. If **async** is zero, it indicates a synchronous acquisition.

**callback** is a pointer to a user completion routine. This routine is called at the completion of the acquisition.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID

## imgSessionConfigure

---

### Format

**rval** = `imgSessionConfigure`(**SESSION\_ID** **sid**, **BUFLIST\_ID** **buflist**)

### Purpose

Specifies the buffer list to use with this session. A valid **BUFLIST\_ID** must be passed. Upon successful completion of this call, you can then call `imgSessionAcquire`.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>buflist</b>	BUFLIST_ID	input	a valid BUFLIST_ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid **SESSION\_ID** type variable.

**buflist** is a valid **BUFLIST\_ID** type variable.

**rval** returns the following status codes:

<b>IMG_ERR_AIOP</b>	cannot perform request, acquisition in progress
<b>IMG_ERR_ELCK</b>	cannot lock buffers down, no more memory
<b>IMG_ERR_GOOD</b>	no error occurred
<b>IMG_ERR_NCAM</b>	no camera defined for this channel
<b>IMG_ERR_PAR1</b>	invalid <b>SESSION_ID</b>
<b>IMG_ERR_PAR2</b>	invalid <b>BUFLIST_ID</b>

## imgSessionCopyArea

---

### Format

```
rval = imgSessionCopyArea(SESSION_ID sid, uInt32 buf_num, uInt32 top, uInt32 left,
                          uInt32 height, uInt32 width, Ptr buffer, uInt32 rowPixels,
                          uInt32 vsync)
```

### Purpose

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

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>buf_num</b>	uInt32	input	element number of buffer to copy
<b>top</b>	uInt32	input	top vertical coordinate of area
<b>left</b>	uInt32	input	left horizontal coordinate of area
<b>width</b>	uInt32	input	width of area
<b>height</b>	uInt32	input	height of area
<b>buffer</b>	Ptr	input	pointer to user image memory
<b>rowPixels</b>	uInt32	input	used in calculating the address of the next line
<b>vsync</b>	uInt32	input	wait until the next vertical blank to perform the copy
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**buf\_num** indicates a valid buffer list element number of the buffer to copy from.

**top** indicates the top vertical coordinate of the area to copy.

**left** indicates the left horizontal coordinate.

**width** indicates the width of the area to copy.

**height** indicates the height of the area to copy.

**buffer** indicates an address to a buffer that is large enough to hold the data.

**rowPixels** indicates the exact pixel-width of the horizontal line to acquire. This parameter specifies the number of pixels to add to the line pointer for the next scan line. This value must be greater than or equal to the width parameter. Passing a zero for this value causes it to be ignored.

**vsync** controls when copying occurs. If **vsync** is TRUE, *imgSessionCopyArea* waits until the next vertical blank to perform the copy. If **vsync** is FALSE, *imgSessionCopyArea* does not wait.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid buffer element number
IMG_ERR_BROI	area requested is greater than the region of interest

## imgSessionCopyBuffer

---

### Format

```
rval = imgSessionCopyBuffer(SESSION_ID sid, uInt32 buf_num, uInt8* buffer,
                             uInt32 vsync)
```

### Purpose

Copies a session's image data to a user buffer format. If the board is capable, a DMA transfer is initiated to the host memory.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>buf_num</b>	uInt32	input	element number of buffer to copy
<b>buffer</b>	uInt8*	input	pointer to user buffer
<b>vsync</b>	uInt32	input	wait until the next vertical blank to perform
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**buf\_num** indicates a buffer list element number that corresponds to the buffer you want to copy.

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

**vsync** controls a session's image data transfer. If **vsync** is TRUE, *imgSessionCopyBuffer* waits until a host vertical blank occurs before transferring. If **vsync** is FALSE, *imgSessionCopyBuffer* does not wait for a host vertical blank.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid buffer element number
IMG_ERR_PAR3	null pointer

## imgSessionExamineBuffer

---

### Format

```
rval = imgSessionExamineBuffer(SESSION_ID sid, uInt32 whichBuffer,
                               uInt32* bufferNumber, uInt32* bufferAddr)
```

### Purpose

Extracts a buffer from a live acquisition. This function lets you lock a buffer out of a continuous loop sequence for processing when you are using a ring (continuous) sequence.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>whichBuffer</b>	uInt32	input	identifies which buffer to get
<b>bufferNumber</b>	uInt32*	input	element number of the returned buffer
<b>bufferAddr</b>	uInt32*	input	a pointer to an address to store the address of the locked buffer
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**whichBuffer** identifies which cumulative frame buffer to get.

**bufferNumber** returns the buffer number of the returned buffer.

**bufferAddr** is a pointer to an address to store the address of the locked buffer.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_NBUF	no buffers available, too early in acquisition
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid command
IMG_ERR_PAR5	null pointer or invalid element number
IMG_ERR_PAR6	null pointer



**Note** Use `imgSessionReleaseBuffer` to release the buffer being held with `imgSessionExamineBuffer`.



## imgSessionReleaseBuffer

---

### Format

**rval** = `imgSessionReleaseBuffer(SESSION_ID sid)`

### Purpose

Releases a buffer that was previously held with `imgSessionExamineBuffer`. This function has the effect of re-entering a buffer into a continuous ring buffer pool after analysis.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID

# Attribute Functions

---

Attribute functions include `imgGetAttribute`, `imgGetCameraAttributeNumeric`, `imgGetCameraAttributeString`, `imgSessionGetLostFramesList`, `imgSessionSetUserLUT8bits`, `imgSessionSetUserLUT16bits`, `imgSetAttribute`, `imgSetCameraAttributeNumeric`, and `imgSetCameraAttributeString`.

You can use these functions to examine and change NI-IMAQ and camera attributes.

When changing NI-IMAQ attributes, remember that attributes are either interface- or session-(channel) specific.

Some attribute changes such as gain or white reference that are session-specific can take effect while a live acquisition is in progress. In this case, the driver will wait for a vertical blank before making the change. Most session attributes, however, require that you call `imgSessionConfigure` to reconfigure the driver, especially when changing the ROI width or height.

NI-IMAQ will not let you change any attribute that would have a detrimental effect on any acquisition in progress. If NI-IMAQ lets you change an attribute during a live acquisition, you should see the effect of the change immediately. If NI-IMAQ does not let you change an attribute during a live acquisition, stop the acquisition, change the attribute, call `imgSessionConfigure`, and restart the acquisition.

Calling `imgSessionConfigure` reprograms the video hardware and recalculates the DMA based on the attributes of the buffers in the session's buffer list.

## imgGetAttribute

---

### Format

**rval** = imgGetAttribute(**uInt32 void\_id**, **uInt32 type**, **void\* value**)

### Purpose

Returns an attribute for an interface or session.

### Parameters

Name	Type	Direction	Description
<b>void_id</b>	uInt32	input	session or interface ID
<b>type</b>	uInt32	input	attribute type
<b>value</b>	void*	input/ output	pointer to a place attribute value
<b>rval</b>	Int32	output	status

### Parameter Discussion

**void\_id** indicates an area of memory reserved for a valid SESSION\_ID or INTERFACE\_ID type variable.

**type** passes a valid SESSION\_ID or INTERFACE\_ID attribute type. See Appendix A, [Attributes and Constants](#), for valid types.

**value** passes a pointer to place attribute value.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid INTERFACE_ID or SESSION_ID
IMG_ERR_PAR2	invalid attribute type
IMG_ERR_PAR3	null pointer

## imgGetCameraAttributeNumeric

---

### Format

**rval** = *imgGetCameraAttributeNumeric* (**SESSION\_ID** *sid*, **Int8\*** *attributeString*, **double\*** *currentValueNumeric*)

### Purpose

Gets the value of numeric camera attributes. Consult the `<my camera>.txt` file in the `ni-imaq\camera` directory for information on valid attributes for your camera.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>attributeString</b>	int8*	input	string containing attribute name
<b>currentValueNumeric</b>	double*	input	pointer to place attribute value
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**attributeString** is a string containing the attribute name.

**currentValueNumeric** is a pointer to place the attribute value. After the function is called, this parameter will contain the current value of the attribute.

**rval** returns IMG\_ERR\_GOOD if no error occurred.

## imgGetCameraAttributeString

---

### Format

```
rval = imgGetCameraAttributeString (SESSION_ID sid, Int8* attributeString,
                                   Int8* currentValueString,
                                   UInt32 sizeofCurrentValueString)
```

### Purpose

Gets the value of camera attributes. Consult the `<my camera>.txt` file in the `ni-imaq\camera` directory for information on valid attributes for your camera. Use this function to get the value of string, integer, or float attribute types. If the attribute is a numeric type (integer or float), use `sscanf` to convert the string into a numeric data type.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>attributeString</b>	int8*	input	string containing attribute name
<b>currentValueString</b>	int8*	input	string containing attribute value
<b>sizeofCurrentValueString</b>	uInt32	input	size of <b>currentValueString</b>
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**attributeString** is a string containing the attribute name.

**currentValueString** is a pointer to an array in memory large enough to hold the attribute value returned. After the function is called, this parameter will contain the current value of the attribute.

**sizeofCurrentValueString** is the size of the array in memory pointed to by **currentValueString**.

**rval** returns IMG\_ERR\_GOOD if no error occurred.

## imgSessionGetLostFramesList

---

```
rval = imgSessionGetLostFramesList(SESSION_ID sid, uInt32* framelist,  
                                     uInt32 numEntries)
```

### Purpose

Gets information about frames that were overwritten during a continuous acquisition. Use this function during a ring acquisition to determine if any frames were overwritten before being examined.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>framelist</b>	uInt32*	input	pointer to cumulative frame number array
<b>numEntries</b>	uInt32	input	size of <b>framelist</b> array
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**framelist** is a pointer to an array of memory allocated by the user that will contain the cumulative frame number of any image buffer that was overwritten during an acquisition.

**numEntries** is the size of the **framelist** array.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid pointer to <b>framelist</b>

## imgSessionSetUserLUT8bits

---

**rval** = `imgSessionSetUserLUT8bits(SESSION_ID sid, uInt32 lutType, uInt8* lut)`

### Purpose

Downloads a custom 8-bit LUT to your IMAQ device. If your camera has more than one LUT, you must program all of them. 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 can also override the default LUT on a per-tap basis. This function works with analog IMAQ devices or digital cameras with 8-bit taps.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>lutType</b>	uInt32	input	LUT to be downloaded
<b>lut</b>	uInt8*	input	pointer to user buffer containing the LUT
<b>rval</b>	Int32	output	no error

### Parameter Discussion

**sid** is a valid `SESSION_ID` type variable.

**lutType** indicates the type of LUT to be written. Valid values are:

<code>IMG_LUT_TYPE_DEFAULT</code>	<code>IMG_LUT_TYPE_TAP0</code>
<code>IMG_LUT_TYPE_RED</code>	<code>IMG_LUT_TYPE_TAP1</code>
<code>IMG_LUT_TYPE_GREEN</code>	<code>IMG_LUT_TYPE_TAP2</code>
<code>IMG_LUT_TYPE_BLUE</code>	<code>IMG_LUT_TYPE_TAP3</code>

**lut** points to an area of memory that contains the LUT to be downloaded to the image acquisition board. This array must contain 256 elements.

**rval** returns the following status codes:

<code>IMG_ERR_GOOD</code>	no error
<code>IMG_ERR_GOOD</code>	no error
<code>IMG_ERR_PAR1</code>	invalid session ID
<code>IMG_ERR_PAR2</code>	unknown LUT type
<code>IMG_ERR_PAR3</code>	null buffer passed as LUT
<code>IMG_ERR_EMEM</code>	not enough memory to perform the operation
<code>IMG_ERR_HWNC</code>	the board does not support this LUT type

## imgSessionSetUserLUT16bits

**rval** = imgSessionSetUserLUT16bits(**SESSION\_ID** sid, **uInt32** lutType, **uInt16\*** lut)

### Purpose

Downloads a custom 16-bit LUT to your IMAQ device. If your camera has more than one LUT, you must program all of them. 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 can also override the default LUT on a per-tap basis. This function works with digital cameras of 10-, 12-, 14-, or 16-bit pixel depths.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>lutType</b>	uInt32	input	LUT to be downloaded
<b>lut</b>	uInt16*	input	pointer to user buffer containing the LUT
<b>rval</b>	Int32	output	no error

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**lutType** indicates the type of LUT to be written. Valid values are:

IMG\_LUT\_TYPE\_DEFAULT  
 IMG\_LUT\_TYPE\_TAP0  
 IMG\_LUT\_TYPE\_TAP1

**lut** points to an area of memory that contains the LUT to be downloaded to the IMAQ 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).

**rval** returns the following status codes:

IMG_ERR_GOOD	no error
IMG_ERR_PAR1	invalid session ID
IMG_ERR_PAR2	unknown LUT type
IMG_ERR_PAR3	null buffer passed as LUT
IMG_ERR_EMEM	not enough memory to perform the operation
IMG_ERR_HWNC	the board does not support this LUT type



## imgSetAttribute

---

### Format

**rval** = `imgSetAttribute(uInt32 void_id, uInt32 type, uInt32 value)`

### Purpose

Sets an attribute for an interface or session.

### Parameters

Name	Type	Direction	Description
<b>void_id</b>	uInt32	input	session or interface ID
<b>type</b>	uInt32	input	attribute type
<b>value</b>	uInt32 or double*	input	new attribute value
<b>rval</b>	Int32	output	status

### Parameter Discussion

**void\_id** indicates an area of memory reserved for a valid `SESSION_ID` or `INTERFACE_ID` type variable.

**type** passes a valid `SESSION_ID` or `INTERFACE_ID` attribute type. See Appendix A, *Attributes and Constants*, for valid types.

**value** is the new attribute value. The type of this input is either a `uInt32` or a pointer to a double. See Appendix A, *Attributes and Constants*, for the attribute type.

**rval** returns the following status codes:

<code>IMG_ERR_GOOD</code>	no error occurred
<code>IMG_ERR_PAR1</code>	invalid <code>INTERFACE_ID</code> or <code>SESSION_ID</code>
<code>IMG_ERR_PAR2</code>	invalid attribute type
<code>IMG_ERR_PAR3</code>	illegal attribute value

## imgSetCameraAttributeNumeric

---

### Format

**rval** = `imgSetCameraAttributeNumeric` (**SESSION\_ID** **sid**, **Int8\*** **attributeString**, **double** **newValueNumeric**)

### Purpose

Sets the value of numeric camera attributes. Consult the `<my camera>.txt` file in the `ni-imaq\camera` directory for information on valid attributes for your camera.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>attributeString</b>	int8*	input	string containing attribute name
<b>newValueNumeric</b>	double	input	pointer to place attribute value
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**attributeString** is a string containing the attribute name.

**newValueNumeric** is a numeric containing the new value of the attribute. Valid attribute values can be found in the `<my camera>.txt` file.

**rval** returns the following status codes:

IMG\_ERR\_GOOD            no error occurred

## imgSetCameraAttributeString

---

### Format

```
rval = imgSetCameraAttributeString(SESSION_ID sid, Int8* attributeString,
                                   Int8* newValueString)
```

### Purpose

Sets the value of camera attributes. Consult the `<my camera>.txt` file in the `ni-imaq\camera` directory for information on valid attributes for your camera. Use this function to set the value of string, integer, or float attribute types. If the attribute is a numeric type (integer or float), this function will convert the string input into a numeric value.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>attributeString</b>	int8*	input	string containing attribute name
<b>newValueString</b>	int8*	input	string containing attribute value
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**attributeString** is a string containing the attribute name.

**newValueString** is a string containing the new value of the attribute. Valid attribute values can be found in the `<my camera>.txt` file.

**rval** returns the following status codes:

IMG\_ERR\_GOOD          no error occurred

## Buffer Management Functions

---

Buffer management functions include `imgCreateBuffer`, `imgCreateBufList`, `imgDisposeBuffer`, `imgDisposeBufList`, `imgGetBufferElement`, `imgSessionClearBuffer`, and `imgSetBufferElement`.

You can use these 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

---

### Format

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

### Purpose

Creates a user frame buffer based on the geometric definitions of the associated session. Passing a null or zero for the SESSION\_ID is valid. In this case, you must pass a buffer size. If buffer size is zero, the buffer size is computed based on the ROI height\*rowPixels for the associated session multiplied times the number of bytes per pixel. An error is returned if the buffer size is smaller than the minimum buffer size required for the session.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>where</b>	uInt32	input	indicates where the buffer should be stored
<b>bufferSize</b>	uInt32	input	size of buffer to create
<b>bufPtrAddr</b>	void*	output	pointer to return the buffer address
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**where** indicates if the buffer should be stored in system memory or in onboard memory on the IMAQ device, as specified by the constants:

```
IMG_HOST_FRAME
IMG_DEVICE_FRAME
```

**bufferSize** indicates the size of the buffer you want to create.

**bufPtrAddr** is a pointer to an area of memory that stores the new buffer address.



**Note** If IMG\_DEVICE\_FRAME is used, the bufPtrAddr returned should not be accessed.

**rval** returns the following status codes:

IMG_ERR_BSIZ	buffer size used is too small for attributes
IMG_ERR_EMEM	not enough memory to perform the operation
IMG_ERR_MXBF	too many buffers already allocated
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid memory constant
IMG_ERR_PAR3	invalid buffer size
IMG_ERR_PAR4	null pointer
IMG_ERR_NVBL	hardware limitation

## imgCreateBufList

---

### Format

**rval** = `imgCreateBufList(uInt32 numElements, BUFLIST_ID* bid)`

### Purpose

Creates a buffer list. This buffer list is passed to `imgSessionConfigure`. The buffer list is initialized to an empty default state and must be filled before calling `imgSessionConfigure`. Use `imgSetBufferElement` to fill the buffer list.

### Parameters

Name	Type	Direction	Description
<b>numElements</b>	uInt32	input	number of entries in the buffer list
<b>bid</b>	BUFLIST_ID*	output	pointer to the new buffer list
<b>rval</b>	Int32	output	status

### Parameter Discussion

**numElements** indicates the maximum number of elements the buffer list should contain.

**bid** is a pointer to an area of memory that contains a BUFLIST\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid number of elements
IMG_ERR_PAR3	null buffer list ID pointer

## imgDisposeBuffer

---

### Format

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

### Purpose

Disposes of a user frame buffer.

### Parameters

Name	Type	Direction	Description
<b>buffPtrAddr</b>	void*	input	pointer to a buffer
<b>rval</b>	Int32	output	status

### Parameter Discussion

**buffPtrAddr** is a pointer to a private buffer or a buffer created by `imgCreateBuffer`.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	null pointer



**Note** Make sure no active buffer lists contain this buffer before disposing of the user frame buffer.



## imgDisposeBufList

---

### Format

**rval** = `imgDisposeBufList(BUFLIST_ID bid, uInt32 freeResources)`

### Purpose

Purges all image buffers associated with this buffer list. You must call `imgSessionConfigure` to reconfigure any session that was attached to the purged buffer list.

### Parameters

Name	Type	Direction	Description
<b>bid</b>	BUFLIST_ID	input	valid BUFLIST_ID
<b>freeResources</b>	uInt32	input	determines whether buffers and buffer list will be disposed or just the buffer list
<b>rval</b>	Int32	output	status

### Parameter Discussion

**bid** is a valid BUFLIST\_ID type variable.

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

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid BUFLIST_ID

## imgGetBufferElement

---

### Format

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

### Purpose

Gets an element of a specific type from a buffer list.

### Parameters

Name	Type	Direction	Description
<b>bid</b>	BUFLIST_ID	input	valid BUFLIST_ID
<b>element</b>	uInt32	input	element number the buffer list
<b>itemType</b>	uInt32	input	element item to get
<b>itemValue</b>	void*	input	new value for item
<b>rval</b>	Int32	output	status

### Parameter Discussion

**bid** is a valid BUFLIST\_ID type variable.

**element** is the element number of the buffer list item to examine.

**itemType** passes a valid buffer list element type as specified by the constants:

- IMG\_BUFF\_ADDRESS
- IMG\_BUFF\_CHANNEL
- IMG\_BUFF\_COMMAND
- IMG\_BUFF\_SIZE
- IMG\_BUFF\_SKIPCOUNT

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

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid BUFLIST_ID
IMG_ERR_PAR2	bad element number
IMG_ERR_PAR3	bad item type
IMG_ERR_PAR4	null pointer



**Note** See *Constants* in Appendix A, *Attributes and Constants*, for valid element and command types.

## imgSessionClearBuffer

---

### Format

**rval** = `imgSessionClearBuffer`(SESSION\_ID **sid**, uInt32 **buf\_num**, uInt8 **pixel\_value**)

### Purpose

Clears a session's image data to the specified pixel value.

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>buf_num</b>	uInt32	input	element number of the buffer to clear
<b>pixel_value</b>	uInt8	input	pixel value to set data to
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**buf\_num** indicates a valid buffer list element number.

**pixel\_value** indicates a pixel value to set all the buffer data with.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_NCFG	invalid action, no buffers configured for session
IMG_ERR_ZBUF	zero buffer size, no bytes filled
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	buffer element number out of range

## imgSetBufferElement

---

### Format

```
rval = imgSetBufferElement(BUFLIST_ID bid, uInt32 element, uInt32 itemType,
                           uInt32 itemValue)
```

### Purpose

Sets a buffer list element of a given type to a specific value.

### Parameters

Name	Type	Direction	Description
<b>bid</b>	BUFLIST_ID	input	a valid BUFLIST_ID
<b>element</b>	uInt32	input	element number the buffer list
<b>itemType</b>	uInt32	input	element item to set
<b>itemValue</b>	uInt32	input	new value for item
<b>rval</b>	Int32	output	status

### Parameter Discussion

**bid** is a valid BUFLIST\_ID type variable.

**element** is the element number of the buffer list item to modify.

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

```

IMG_BUFF_ADDRESS      IMG_BUFF_SIZE
IMG_BUFF_CHANNEL     IMG_BUFF_SKIPCOUNT
IMG_BUFF_COMMAND     IMG_BUFF_TRIGGER
IMG_BUFF_NUMBUFS
```

**itemValue** indicates the value of the element type to set. Use the following constants to specify the IMG\_BUFF\_COMMAND itemType:

```

IMG_CMD_LOOP
IMG_CMD_NEXT
IMG_CMD_PASS
IMG_CMD_STOP
```

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid BUFLIST_ID
IMG_ERR_PAR2	bad element number
IMG_ERR_PAR3	bad item type
IMG_ERR_PAR4	bad command type



**Note** See *Constants* in Appendix A, *Attributes and Constants*, for valid element and command types.

## Interface Functions

---

Interface functions include `imgInterfaceQueryNames` and `imgInterfaceReset`.

You can use these functions to perform operations specific to an interface. All interface functions require a valid `INTERFACE_ID`.

Interface functions operate on a board-wide basis. When you make a call to an interface function, it will affect all sessions connected to that interface in all processes. Interface function changes are global and must be done with care.

## imgInterfaceQueryNames

---

### Format

**rval** = imgInterfaceQueryNames(**uInt32** index, **Int8\*** queryName)

### 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 the **index** parameter initialized to zero. Each successive call increments the **index** parameter by one.

### Parameters

Name	Type	Direction	Description
<b>index</b>	uInt32	input	interface number to obtain
<b>queryName</b>	Int8*	input	pointer to an array of strings
<b>rval</b>	Int32	output	status

### Parameter Discussion

**index** is the interface number to obtain.

**queryNames** is a pointer to an array in memory large enough to hold the interface name returned (INTERFACE\_NAME\_SIZE).

**rval** returns the following status codes:

IMG_ERR_EMEM	not enough memory to perform the operation
IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	null pointer



## imgInterfaceReset

---

### Format

**rval** = imgInterfaceReset(**INTERFACE\_ID ifid**)

### Purpose

Performs a hardware reset on the interface type and returns a status, either good or bad. This function sets the hardware associated with the interface to its default state and resets the onboard DMA controller.

### Parameters

Name	Type	Direction	Description
<b>ifid</b>	INTERFACE_ID	output	interface ID
<b>rval</b>	Int32	output	status

### Parameter Discussion

**ifid** is a valid INTERFACE\_ID type variable.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	bad INTERFACE_ID

# Utility Functions

---

Utility functions include `imgPlot`, `imgSessionSaveBufferEx`, and `imgShowError`.

You can use these functions to display an image in a window, save an image to a file, or to get detailed error information.

## imgPlot

---

### Format

```
rval = imgPlot(GUIHNDL window, 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 window handle. This function is an easy way to display a buffer after it is acquired.

### Parameters

Name	Type	Direction	Description
<b>window</b>	GUIHNDL	input	window handle
<b>buffer</b>	void*	input	pointer to video data
<b>leftBufOffset</b>	uInt32	input	x-offset into the buffer to start plotting
<b>topBufOffset</b>	uInt32	input	y-offset into the buffer to start plotting
<b>xsize</b>	uInt32	input	width of the image
<b>ysize</b>	uInt32	input	height of the image
<b>xpos</b>	uInt32	input	x-position to start plotting in the window
<b>ypos</b>	uInt32	input	y-position to start plotting in the window
<b>flags</b>	uInt32	input	set display property
<b>rval</b>	Int32	output	status

### Parameter Discussion

**window** is a native window handle designating the window to plot in.

**buffer** is a pointer to an area of memory containing a video frame buffer.

**leftBufOffset** is the left offset into the buffer to start plotting.

**topBufOffset** is the top offset into the buffer to start plotting.

**xsize** is the pixel width of the image.

**ysize** is the number of scanlines (pixel height) in the image.

**xpos** is the left position to start plotting in the window.

**ypos** is the top position to start plotting in the window.

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

IMGPLOT_INVERT	IMGPLOT_MONO_12
IMGPLOT_COLOR_RGB24	IMGPLOT_MONO_14
IMGPLOT_COLOR_RGB32	IMGPLOT_MONO_16
IMGPLOT_MONO_8	IMGPLOT_MONO_32
IMGPLOT_MONO_10	

Use `IMGPLOT_COLOR_RGB24` and `IMGPLOT_COLOR_RGB32` constants to display a color image. You can use the `IMGPLOT_INVERT` constant with either `IMGPLOT_COLOR_RGB24` or `IMGPLOT_COLOR_RGB32` constants to invert the image.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	bad window handle
IMG_ERR_PAR2	null pointer

## imgSessionSaveBufferEx

---

### Format

**rval** = `imgSessionSaveBufferEx(SESSION_ID sid, void* buffer, Int8* file_name)`

### Purpose

Saves a buffer of a session to disk in a native operating system-specific format such as bitmap or tag image file format (TIFF).

### Parameters

Name	Type	Direction	Description
<b>sid</b>	SESSION_ID	input	session ID
<b>buffer</b>	void*	input	buffer to save
<b>file_name</b>	Int8*	input	null terminated string describing a file name (optional path)
<b>rval</b>	Int32	output	status

### Parameter Discussion

**sid** is a valid SESSION\_ID type variable.

**buffer** is a pointer to an image buffer to save.



**Note** Information such as bits per pixel and buffer size will be taken from the current session settings. Therefore, this buffer must be associated with the current session.

**file\_name** indicates a file name used to save the image. If the filename has the extension `.bmp`, the image will be saved as a bitmap file. Bitmap files support 8-bit monochrome and 32-bit RGB color images. If the filename extension is `.tif`, the image will be saved as a TIFF file. TIFF files support all image types. If the filename extension is `.png`, the image will be saved as an uncompressed 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, you should use the PNG file format.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BADFILEEXT	unknown file extension
IMG_ERR_BADFILFMT	bad file format
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid buffer element number
IMG_ERR_PAR3	null pointer

## imgShowError

---

### Format

**rval** = imgShowError(**IMG\_ERR** error, **char\*** text)

### Purpose

Returns a null terminated string describing the error code.

### Parameters

Name	Type	Direction	Description
<b>error</b>	IMG_ERR	input	a valid error code
<b>text</b>	char*	input	pointer to a character array of no less than 255 bytes
<b>rval</b>	Int32	output	status

### Parameter Discussion

**error** is a valid NI-IMAQ error code.

**text** is a pointer to an area of memory reserved for an error string.

**rval** returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid error code
IMG_ERR_PAR2	null pointer

# Attributes and Constants

This appendix describes the attributes and constants used by NI-IMAQ.

## NI-IMAQ Attributes

Attributes describe a specific property of a session or interface. A summary of NI-IMAQ attributes is listed in Table A-1.

*Attribute* describes the constant name of the attribute. The *type* is the attribute's data type. *I/S* describes whether the get/set attribute function requires an `INTERFACE_ID` (I) or `SESSION_ID` (S) type parameter. *Imd.* describes whether the effect of setting the attribute is immediate (Yes), or whether it requires a subsequent `imgSessionConfigure` to take effect (No). *R/W* describes whether the attribute is read only (R), write only (W), or both (R/W). *Description* describes what values the attribute can take and the effect the setting of the attribute has or what values are returned.

**Table A-1.** Attribute Summary

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_ACQ_IN_PROGRESS	uInt32	S	Yes	R	Returns TRUE if an acquisition is in progress on the camera associated with this session
IMG_ATTR_ACQUIRE_FIELD	uInt32	S	Yes	R/W	Sets the field acquired when <code>IMG_ATTR_FRAME_FIELD</code> is set to <code>FIELD_MODE</code> . Possible values are: <img_acquire_odd </img_acquire_odd  <img_acquire_even </img_acquire_even  <img_acquire_all </img_acquire_all  The IMAQ PCI/PXI-1408 supports only <code>IMG_ACQUIRE_ALL</code> .
IMG_ATTR_ACQWINDOW_HEIGHT	uInt32	S	No	R/W	Gets/sets the acquisition window height of the camera/channel associated with this session
IMG_ATTR_ACQWINDOW_LEFT	uInt32	S	No	R/W	Gets/sets the acquisition window left of the camera/channel associated with this session
IMG_ATTR_ACQWINDOW_TOP	uInt32	S	No	R/W	Gets/sets the acquisition window top of the camera/channel associated with this session
IMG_ATTR_ACQWINDOW_WIDTH	uInt32	S	No	R/W	Gets/sets the acquisition window width of the camera/channel associated with this session
IMG_ATTR_BITSPERPIXEL	uInt32	S	Yes	R	Returns the bits per pixel value of the camera/channel associated with this session



**Table A-1.** Attribute Summary (Continued)

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_BLACK_REF_VOLT	double*	S	Yes	R/W	Sets the black reference value, in volts, of the channel associated with this session. Values are 0 V to 1.26 V.
IMG_ATTR_BYTESPERPIXEL	uInt32	S	Yes	R	Returns the bytes per pixel value of the camera/channel associated with this session
IMG_ATTR_CHROMA_FILTER	uInt32	S	Yes	R/W	Gets/sets the antichrominance filter to be used. Values are: IMG_FILTER_NONE IMG_FILTER_NTSC IMG_FILTER_PAL
IMG_ATTR_COLOR	uInt32	I	Yes	R	Returns TRUE if the interface board is color-capable
IMG_ATTR_COLOR_AVG_COUNT	uInt32	S	Yes	R/W	Gets/sets the number of color images to be acquired and averaged for one output image (1–128). Default value is 1.
IMG_ATTR_COLOR_BRIGHTNESS	double*	S	No	R/W	Adjusts the brightness of an image—the amount of white light added to or subtracted from each image pixel. The range is –50 to +50 IRE (percentage of the white level) in steps of 1, with a default of 0 IRE.
IMG_ATTR_COLOR_CHROMA_BANDWIDTH	uInt32	S	No	R/W	Specifies the resulting bandwidth of the chroma information of the image. Values are: IMG_COLOR_CHROMA_BANDWIDTH_HIGH IMG_COLOR_CHROMA_BANDWIDTH_LOW
IMG_ATTR_COLOR_CHROMA_COMB	uInt32	S	No	R/W	Selects the type of comb filter used in the chroma path. Values are: IMG_COLOR_COMB_OFF IMG_COLOR_COMB_1LINE IMG_COLOR_COMB_2LINES
IMG_ATTR_COLOR_CHROMA_PHASE	double*	S	No	R/W	Specifies the value of a correction angle that can be applied to the chroma vector (adjustment of tint). This attribute is only active when an NTSC camera is used. The range is –180 to +180 degrees, with a default of 0 degrees.
IMG_ATTR_COLOR_CHROMA_PROCESS	uInt32	S	No	R/W	Specifies the processing applied to the chroma signal. Values are: IMG_COLOR_CHROMA_PROCESS_ALWAYS_OFF IMG_COLOR_CHROMA_PROCESS_ALWAYS_ON IMG_COLOR_CHROMA_PROCESS_AUTODETECT
IMG_ATTR_COLOR_CHROMA_TRAP	uInt32	S	No	R/W	Enables the chroma trap filter in the luma signal path. Should always be disabled in S-Video / Y/C mode. Values are: 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
IMG_ATTR_COLOR_CONTRAST	double*	S	No	R/W	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. The range is 0.4 to 1.5 with a default of 1.00

**Table A-1.** Attribute Summary (Continued)

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_COLOR_GAIN	uInt32	S	Yes	R/W	Gets/sets the hardware gain of the IMAQ device when StillColor is selected. The gain settings are: 1.00 (0) 1.33 (1) 2.00 (2)
IMG_ATTR_COLOR_HIGH_REF_VOLT	double*	S	Yes	R/W	Gets/sets the hardware black reference of your IMAQ device when StillColor is selected. Values are 0 V to 1.26 V.
IMG_ATTR_COLOR_HSL_CORING_LEVEL	uInt32	S	No	R/W	Sets the HSL coring level when Image Representation is set to HSL. In HSL mode, on any image pixel, if the saturation value (S) of the pixel is lower than the specified value, then the Hue value (H) of the pixel will be set to zero. The range is 0 to 255 lsb, with a default of 0 lsb.
IMG_ATTR_COLOR_HUE_OFFSET_ANGLE	double*	S	No	R/W	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. This angle can be offset to move the discontinuity point (at 0 modulo 360 degrees) to another angle value. The range is -180 to +180 degrees with a default of 0 degrees.
IMG_ATTR_COLOR_IMAGE_REP	uInt32	S	No	R/W	Specifies the type of image data that will be returned when a color image is acquired. Values are: IMG_COLOR_REP_RGB32 IMG_COLOR_REP_RED8 IMG_COLOR_REP_GREEN8 IMG_COLOR_REP_BLUE8 IMG_COLOR_REP_LUM8 IMG_COLOR_REP_HUE8 IMG_COLOR_REP_SAT8 IMG_COLOR_REP_INT8 IMG_COLOR_REP_LUM16 IMG_COLOR_REP_HUE16 IMG_COLOR_REP_SAT16 IMG_COLOR_REP_INT16 IMG_COLOR_REP_RGB24 IMG_COLOR_REP_RGB16 IMG_COLOR_REP_HSL32 IMG_COLOR_REP_HSI32
IMG_ATTR_COLOR_LOW_REF_VOLT	double*	S	Yes	R/W	Gets/sets the hardware white reference of your IMAQ device when StillColor is selected. Values are 0 V to 1.26 V.
IMG_ATTR_COLOR_LUMA_BANDWIDTH	uInt32	S	No	R/W	Selects different bandwidths for the luminance signal. The four valid options are: IMG_COLOR_LUMA_BANDWIDTH_FULL IMG_COLOR_LUMA_BANDWIDTH_HIGH IMG_COLOR_LUMA_BANDWIDTH_MEDIUM IMG_COLOR_LUMA_BANDWIDTH_LOW
IMG_ATTR_COLOR_LUMA_COMB	uInt32	S	No	R/W	Selects the type of comb filter used in the luma path. IMG_COLOR_COMB_OFF IMG_COLOR_COMB_1LINE IMG_COLOR_COMB_2LINES

**Table A-1.** Attribute Summary (Continued)

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_COLOR_MODE	uInt32	S	Yes	R/W	Sets or gets the color acquisition mode. Values are: IMG_COLOR_MODE_DISABLED IMG_COLOR_MODE_RGB IMG_COLOR_MODE_COMPOSITE_STLC
IMG_ATTR_COLOR_NTSC_SETUP_ENABLE	uInt32	S	No	R/W	Enables the setup correction of 7.5 IRE in NTSC mode. A standard NTSC signal has a setup level that moves up the black level 7.5% of the white level (or 7.5 IRE). Correction for this setup can be performed during acquisition by enabling this attribute. FALSE disables the setup correction (default in PAL or CCIR mode) and TRUE enables the setup correction (default in NTSC or RS-170 mode).
IMG_ATTR_COLOR_NTSC_SETUP_VALUE	double*	S	No	R/W	Gets/sets the NTSC setup compensation value. The unit is IRE (percentage of white level). Default value is 7.5%. (StillColor NTSC only)
IMG_ATTR_COLOR_PEAKING_ENABLE	uInt32	S	No	R/W	Enables the peaking filter in the luma path. Values are: FALSE—Peaking filter disabled (default) TRUE—Peaking filter enabled
IMG_ATTR_COLOR_RGB_CORING_LEVEL	uInt32	S	No	R/W	Selects between four different coring levels. On any image pixel, if the color saturation of the pixel is lower than the specified value, then the saturation will be set to zero, resulting in a monochrome pixel. Values are: IMG_COLOR_RGB_CORING_LEVEL_NOCORING IMG_COLOR_RGB_CORING_LEVEL_C1 IMG_COLOR_RGB_CORING_LEVEL_C3 IMG_COLOR_RGB_CORING_LEVEL_C7
IMG_ATTR_COLOR_SATURATION	double*	S	No	R/W	Adjusts the saturation of the image—a factor multiplied to the chroma information of the image. The rang is 0.5 to 1.5 with a default of 1.00.
IMG_ATTR_COLOR_SW_CHROMA_FILTER	uInt32	S	Yes	R/W	Gets/sets the software filter to clean the chroma signal (StillColor NTSC only): Disabled (0) Enabled (1)
IMG_ATTR_COLOR_TINT	double*	S	Yes	R/W	Gets/sets the tint of your image. Tint is specified in degrees and corresponds to the rotation of the UV color plane. Default value is 0.
IMG_ATTR_CURRENT_BUFLIST	uInt32	S	Yes	R	Returns the BUFLIST_ID of the buffer list associated with this session
IMG_ATTR_FRAME_COUNT	uInt32	S	Yes	R	Returns the number of frames acquired since the start of an acquisition
IMG_ATTR_FRAME_FIELD	uInt32	S	Yes	R/W	Sets/gets the current mode of the session. IMG_FIELD_MODE sets field mode acquisition. In this mode, use the IMG_ATTR_ACQUIRE_FIELD attribute to set the field acquired. IMG_FRAME_MODE sets frame mode acquisition. In this mode, use the IMG_ATTR_START_FIELD attribute to set the field to acquire first.

**Table A-1.** Attribute Summary (Continued)

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_FRAMEWAIT_MSEC	uInt32	S	No	R/W	Gets/sets the timeout value for a frame. Values are: IMG_FRAMETIME_STANDARD IMG_FRAMETIME_1SECOND IMG_FRAMETIME_2SECONDS IMG_FRAMETIME_5SECONDS IMG_FRAMETIME_10SECONDS IMG_FRAMETIME_1MINUTE IMG_FRAMETIME_2MINUTES IMG_FRAMETIME_5MINUTES IMG_FRAMETIME_10MINUTES
IMG_ATTR_FREE_BUFFERS	uInt32	S	Yes	R	Returns the number of reserved driver buffers currently left
IMG_ATTR_GAIN	uInt32	S	Yes	R/W	Set the video gain for the channel associated with this session. Values are: IMG_GAIN_0DB IMG_GAIN_3DB IMG_GAIN_6DB
IMG_ATTR_HASRAM	uInt32	I	Yes	R	Returns TRUE if the interface board has onboard memory
IMG_ATTR_HORZ_RESOLUTION	uInt32	I	Yes	R	Returns the maximum horizontal resolution of the interface
IMG_ATTR_HSCALE	uInt32	S	No	R/W	Gets/sets the horizontal hardware scaling factor for the channel associated with this session. Values are: IMG_SCALE_NONE IMG_SCALE_DIV2 IMG_SCALE_DIV4 IMG_SCALE_DIV8
IMG_ATTR_INVERT	uInt32	S	No	R/W	Gets/sets the invert image mode. Beneficial when calling <code>imgPlot</code> . Values are: 0 = no invert—image in memory is right-side up 1 = invert—image in memory is upside down
IMG_ATTR_LAST_VALID_BUFFER	uInt32	S	Yes	R	Returns a buffer element number of the last received frame buffer
IMG_ATTR_LAST_VALID_FRAME	uInt32	S	Yes	R	Returns the cumulative buffer index (frame#)
IMG_ATTR_LINE_COUNT	uInt32	S	Yes	R	Returns the current line count of the frame being acquired
IMG_ATTR_LUT	uInt32	S	Yes	R/W	Programs the lookup table for the given interface. Pass a constant to indicate the LUT you wish to use or you can pass a pointer to your own LUT. Constant values are: IMG_LUT_NORMAL IMG_LUT_INVERSE IMG_LUT_LOG IMG_LUT_INVERSE_LOG IMG_LUT_BINARY IMG_LUT_INVERSE_BINARY
IMG_ATTR_MEM_LOCKED	uInt32	S	Yes	R	Returns TRUE if the session's buffer list is locked in memory

**Table A-1.** Attribute Summary (Continued)

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_NUM_BUFFERS	uInt32	S	Yes	R	Returns the number of buffers in the buffer list associated with the session
IMG_ATTR_PCLK_DETECT	uInt32	S	Yes	R/W	Determines if NI-IMAQ checks for the existence of a pixel clock before starting an acquisition. The default value is TRUE. If your camera has a pixel clock less than 5 MHz, this detection may fail and you should set this attribute to FALSE. This attribute is valid only on the PCI-1424.
IMG_ATTR_PIXDEPTH	uInt32	I	Yes	R	Returns the maximum pixel depth of the interface board in bytes
IMG_ATTR_RAMSIZE	uInt32	I	Yes	R	Returns the size of the RAM on the interface board
IMG_ATTR_ROI_HEIGHT	uInt32	S	No	R/W	Gets/sets the region of interest height of the camera/channel associated with this session
IMG_ATTR_ROI_LEFT	uInt32	S	Yes	R/W	Gets/sets the region of interest left of the camera/channel associated with this session
IMG_ATTR_ROI_TOP	uInt32	S	Yes	R/W	Gets/sets the region of interest top of the camera/channel associated with this session
IMG_ATTR_ROI_WIDTH	uInt32	S	No	R/W	Gets/sets the region of interest width of the camera/channel associated with this session
IMG_ATTR_ROWBYTES	uInt32	S	No	R/W	Set the true width of a horizontal line in memory. Used to calculate the next lines memory offset
IMG_ATTR_START_FIELD	uInt32	S	No	R/W	Sets/gets the start field setting of the camera when IMG_ATTR_FRAME_FIELD is set to FRAME_MODE. Possible values are: IMG_FIELD_ODD IMG_FIELD_EVEN
IMG_ATTR_TRIGGER_MODE	uInt32	S	No	R/W	Gets/sets the trigger mode for the channel associated with this session. Values are: IMG_TRIGMODE_NONE IMG_TRIGMODE_NOREPEAT IMG_TRIGMODE_REPEAT
IMG_ATTR_VERT_RESOLUTION	uInt32	I	Yes	R	Returns the maximum vertical resolution of the interface
IMG_ATTR_VSCALE	uInt32	S	No	R	Gets/sets the vertical hardware scaling factor for the channel associated with this session. Values are: IMG_SCALE_NONE IMG_SCALE_DIV2 IMG_SCALE_DIV4 IMG_SCALE_DIV8
IMG_ATTR_WHITE_REF_VOLT	double*	S	Yes	R/W	Sets the white reference value, in volts, of the channel associated with this session. Values are 0 V to 1.26 V.

**Table A-1.** Attribute Summary (Continued)

Attribute	Type	I/S	Imd.	R/W	Description
IMG_ATTR_XOFF_BUFFER	uint32	S	No	R/W	Gets/sets the buffer x-left offset for image displacement. Use this attribute to acquire an image into a private buffer at a different location other than the top-left corner. You must use a private buffer when using this attribute.
IMG_ATTR_YOFF_BUFFER	uint32	S	No	R/W	Gets/sets the buffer y-line offset for image displacement. Use this attribute to acquire an image into a private buffer at a different location other than the top-left corner. You must use a private buffer when using this attribute.

## Constants

Constants help clearly define specific function parameter values. These constants are included in your `niimaq.h` header file. Use these constants when coding the `imgGetAttribute` and `imgSetAttribute` functions when required.

Table A-2 lists the constant name, the function to which the constant applies, and a general description.

**Table A-2.** Constants Summary

Constant	Use With	Description
IMG_ACQUIRE_ALL	<code>imgGetAttribute</code> <code>imgSetAttribute</code>	Acquire all fields
IMG_ACQUIRE_EVEN	<code>imgGetAttribute</code> <code>imgSetAttribute</code>	Acquire only even fields
IMG_ACQUIRE_ODD	<code>imgGetAttribute</code> <code>imgSetAttribute</code>	Acquire only odd fields
IMG_AQ_DONE	<code>imgSessionWaitSignal</code> <code>imgSessionWaitSignalAsync</code> <code>imgPulseCreate</code>	Asserted at the end of an acquisition when the last piece of data has been transferred to memory
IMG_AQ_IN_PROGRESS	<code>imgSessionWaitSignal</code> <code>imgSessionWaitSignalAsync</code> <code>imgPulseCreate</code>	Asserted when the board initiates an acquisition either through a software- or hardware-triggered start
IMG_BOARD_INTERFACE	<code>imgGetAttribute</code>	Specifies the type of interface is a plug-in board

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_BUF_COMPLETE	imgSessionWaitSignal imgSessionWaitSignalAsync imgPulseCreate	Asserted when an image buffer has been transferred to memory and is available for image processing
IMG_BUFF_ADDRESS	imgGetBufferElement imgSetBufferElement	Specifies the buffer address portion of a buffer list element
IMG_BUFF_CHANNEL	imgGetBufferElement imgSetBufferElement	Specifies the channel from which to acquire an image
IMG_BUFF_COMMAND	imgGetBufferElement imgSetBufferElement	Specifies the command portion of a buffer list element
IMG_BUFF_NUMBUFS	imgGetBufferElement imgSetBufferElement	Specifies the number of items in a buffer list
IMG_BUFF_SIZE	imgGetBufferElement imgSetBufferElement	Specifies the size portion of a buffer list element (the buffer's size). Required for user-allocated buffers.
IMG_BUFF_SKIPCOUNT	imgGetBufferElement imgSetBufferElement	Specifies the skip count portion of a buffer list element
IMG_CMD_LOOP	imgGetBufferElement imgSetBufferElement (with IMG_BUFF_COMMAND constant)	Specifies a buffer list command of LOOP. Used as the command for the last buffer element, this causes an acquisition to perform a continuous type of acquisition such as a ring.
IMG_CMD_NEXT	imgGetBufferElement imgSetBufferElement (with IMG_BUFF_COMMAND constant)	Specifies a buffer list command of NEXT. This causes an acquisition to take place on the buffer and to proceed to the next buffer list element.
IMG_CMD_PASS	imgGetBufferElement imgSetBufferElement (with IMG_BUFF_COMMAND constant)	Specifies a buffer list command of PASS. Any buffer list element with this command is ignored. Use to reserved space for fast configuration.

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_CMD_STOP	imgGetBufferElement imgSetBufferElement (with IMG_BUFF_COMMAND constant)	Specifies a buffer list command of STOP. Used as the command for the last buffer element, this causes an acquisition to perform a one shot acquisition such as a sequence.
IMG_COLOR_CHROMA_BANDWIDTH_HIGH	imgSetAttribute imgGetAttribute	Highest bandwidth (default)
IMG_COLOR_CHROMA_BANDWIDTH_LOW	imgSetAttribute imgGetAttribute	Lowest bandwidth
IMG_COLOR_CHROMA_PROCESS_ALWAYS_OFF	imgSetAttribute imgGetAttribute	Use when a monochrome camera is used (default for CCIR or RS-170)
IMG_COLOR_CHROMA_PROCESS_ALWAYS_ON	imgSetAttribute imgGetAttribute	Use when a color camera is used (default for NTSC or PAL)
IMG_COLOR_CHROMA_PROCESS_AUTODETECT	imgSetAttribute imgGetAttribute	Can be used if the camera type (monochrome or color) is unknown.
IMG_COLOR_COMB_1LINE	imgSetAttribute imgGetAttribute	Comb filtering using one delayed line
IMG_COLOR_COMB_2LINES	imgSetAttribute imgGetAttribute	Comb filtering using two delayed lines
IMG_COLOR_COMB_OFF	imgSetAttribute imgGetAttribute	Comb filter disabled (default in S-Video/ Y/C mode)
IMG_COLOR_LUMA_BANDWIDTH_FULL	imgSetAttribute imgGetAttribute	All filters including decimation filter disabled. Default value in CCIR or RS-170 mode
IMG_COLOR_LUMA_BANDWIDTH_HIGH	imgSetAttribute imgGetAttribute	Highest available bandwidth with decimation filter enabled. Default value for PAL or NTSC mode.
IMG_COLOR_LUMA_BANDWIDTH_LOW	imgSetAttribute imgGetAttribute	Decimation filter enabled, lowest bandwidth
IMG_COLOR_LUMA_BANDWIDTH_MEDIUM	imgSetAttribute imgGetAttribute	Decimation filter enabled, medium bandwidth



**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_COLOR_MODE_COMPOSITE_STLC	imgGetAttribute imgSetAttribute	Specifies that color mode is StillColor Composite
IMG_COLOR_MODE_DISABLED	imgGetAttribute imgSetAttribute	Specifies that color mode is disabled
IMG_COLOR_MODE_RGB	imgGetAttribute imgSetAttribute	Specifies that color mode is StillColor RGB
IMG_COLOR_REP_BLUE8	imgGetAttribute imgSetAttribute	Specifies 8-bit Blue color output
IMG_COLOR_REP_GREEN8	imgGetAttribute imgSetAttribute	Specifies 8-bit Green color output
IMG_COLOR_REP_HSI32	imgGetAttribute imgSetAttribute	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	imgGetAttribute imgSetAttribute	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	imgGetAttribute imgSetAttribute	Specifies 16-bit Hue color output
IMG_COLOR_REP_HUE8	imgGetAttribute imgSetAttribute	Specifies 8-bit Hue color output
IMG_COLOR_REP_INT16	imgGetAttribute imgSetAttribute	Specifies 16-bit Intensity color output
IMG_COLOR_REP_INT8	imgGetAttribute imgSetAttribute	Specifies 8-bit Intensity color output
IMG_COLOR_REP_LUM16	imgGetAttribute imgSetAttribute	Specifies 16-bit Luminance color output
IMG_COLOR_REP_LUM8	imgGetAttribute imgSetAttribute	Specifies 8-bit Luminance color output
IMG_COLOR_REP_RED8	imgGetAttribute imgSetAttribute	Specifies 8-bit Red color output
IMG_COLOR_REP_RGB16	imgGetAttribute imgSetAttribute	Specifies 16-bit RGB color output (x 555)
IMG_COLOR_REP_RGB24	imgGetAttribute imgSetAttribute	Specifies 24-bit RGB color output (default)

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_COLOR_REP_RGB32	imgGetAttribute imgSetAttribute	Specifies 32-bit RGB color output
IMG_COLOR_REP_RGB48	imgGetAttribute imgSetAttribute	Specifies 48-bit RGB color output
IMG_COLOR_REP_SAT16	imgGetAttribute imgSetAttribute	Specifies 16-bit Saturation color output
IMG_COLOR_REP_SAT8	imgGetAttribute imgSetAttribute	Specifies 8-bit Saturation color output
IMG_COLOR_RGB_CORING_LEVEL_NOCORING	imgSetAttribute imgGetAttribute	The coring function is disabled
IMG_COLOR_RGB_CORING_LEVEL_C1	imgSetAttribute imgGetAttribute	Coring activated for saturation equal or below 1 lsb
IMG_COLOR_RGB_CORING_LEVEL_C3	imgSetAttribute imgGetAttribute	Coring activated for saturation equal or below 3 lsb
IMG_COLOR_RGB_CORING_LEVEL_C7	imgSetAttribute imgGetAttribute	Coring activated for saturation equal or below 7 lsb
IMG_CURRENT_BUFFER	imgSessionExamineBuffer	Specifies to examine current buffer in a live acquisition. Waits until vertical blank to return the buffer to you.
IMG_DEVICE_FRAME	imgCreateBuffer	Specifies the new buffer is created in onboard memory (not supported on the IMAQ PCI-1408)
IMG_EXT_RTISI0	imgPulseCreate imgSessionWaitSignal imgSessionWaitSignalAsync imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 0
IMG_EXT_RTISI1	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 1
IMG_EXT_RTISI2	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 2

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_EXT_RTISI3	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 3
IMG_EXT_RTISI4	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 4
IMG_EXT_RTISI5	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 5
IMG_EXT_RTISI6	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the RTSI line 6
IMG_EXT_TRIG0	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the external trigger 0
IMG_EXT_TRIG1	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the external trigger 1
IMG_EXT_TRIG2	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the external trigger 2
IMG_EXT_TRIG3	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the external trigger 3
IMG_FIELD_EVEN	imgGetAttribute imgSetAttribute	Specifies the start field of the acquisition as even
IMG_FIELD_MODE	imgGetAttribute imgSetAttribute	Specifies the acquisition mode as field
IMG_FIELD_ODD	imgGetAttribute imgSetAttribute	Specifies the start field of the acquisition as odd
IMG_FILTER_NONE	imgGetAttribute imgSetAttribute	Specifies no video filter
IMG_FILTER_NTSC	imgGetAttribute imgSetAttribute	Specifies the video filter is NTSC
IMG_FILTER_PAL	imgGetAttribute imgSetAttribute	Specifies the video filter is PAL
IMG_FRAME_DONE	imgSessionWaitSignal imgSessionWaitSignalAsync imgPulseCreate	Asserted at the end of acquisition into each image buffer

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_FRAME_MODE	imgGetAttribute imgSetAttribute	Specifies the acquisition mode as interlaced
IMG_FRAME_START	imgSessionWaitSignal imgSessionWaitSignalAsync imgPulseCreate	Asserted at the start of acquisition into each image buffer
IMG_FRAMETIME_10MINUTES	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 10 minutes
IMG_FRAMETIME_10SECONDS	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 10 s
IMG_FRAMETIME_1MINUTE	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 1 minute
IMG_FRAMETIME_1SECOND	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 1 s
IMG_FRAMETIME_2MINUTES	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 2 minutes
IMG_FRAMETIME_2SECONDS	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 2 s
IMG_FRAMETIME_5MINUTES	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 5 minutes
IMG_FRAMETIME_5SECONDS	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 5 s
IMG_FRAMETIME_STANDARD	imgGetAttribute imgSetAttribute	Specifies a frame timeout value of 100 ms
IMG_GAIN_0DB	imgGetAttribute imgSetAttribute	Specifies the gain is +0 dB
IMG_GAIN_3DB	imgGetAttribute imgSetAttribute	Specifies the gain is +3 dB
IMG_GAIN_6DB	imgGetAttribute imgSetAttribute	Specifies the gain is +6 dB
IMG_HOST_FRAME	imgCreateBuffer	Specifies the new buffer is created in host (computer) memory
IMG_IMMEDIATE	imgPulseCreate	Causes the pulse to be generated immediately when the function is executed

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_LAST_BUFFER	imgSessionExamineBuffer	Specifies to examine the last valid buffer in a live acquisition
IMG_LUT_BINARY	imgGetAttribute imgSetAttribute	Specifies a binary LUT, which converts the sampled data to a binary image of black and white
IMG_LUT_INVERSE	imgGetAttribute imgSetAttribute	Specifies an inverse LUT, which inverts the gray levels
IMG_LUT_INVERSE_BINARY	imgGetAttribute imgSetAttribute	Specifies an inverse binary LUT, which converts the sampled data into a binary image of white and black
IMG_LUT_INVERSE_LOG	imgGetAttribute imgSetAttribute	Specifies an inverse log LUT, which converts the sampled data to a logarithmic form that produces greater contrast in the white region
IMG_LUT_LOG	imgGetAttribute imgSetAttribute	Specifies a log LUT, which converts the sampled data to a logarithmic form that produces greater contrast in the black region
IMG_LUT_NORMAL	imgGetAttribute imgSetAttribute	Specifies a normal LUT
IMG_LUT_TYPE_DEFAULT	imgSessionSetUserLUT8bits imgSessionSetUserLUT16bits	Default LUT used to initialize all LUTs
IMG_LUT_TYPE_RED	imgSessionSetUserLUT8bits	Red channel LUT for RGB digital cameras or the PCI/PXI-1411
IMG_LUT_TYPE_GREEN	imgSessionSetUserLUT8bits	Green channel LUT for RGB digital cameras or the PCI/PXI-1411
IMG_LUT_TYPE_BLUE	imgSessionSetUserLUT8bits	Blue channel LUT for RGB digital cameras or the PCI/PXI-1411
IMG_LUT_TYPE_TAP0	imgSessionSetUserLUT8bits imgSessionSetUserLUT16bits	Tap 0 LUT for digital boards

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_LUT_TYPE_TAP1	imgSessionSetUserLUT8bits imgSessionSetUserLUT16bits	Tap 1 LUT for digital boards
IMG_LUT_TYPE_TAP2	imgSessionSetUserLUT8bits	Tap 2 LUT for digital boards
IMG_LUT_TYPE_TAP3	imgSessionSetUserLUT8bits	Tap 3 LUT for digital boards
IMG_OLDEST_BUFFER	imgSessionExamineBuffer	Specifies to examine the oldest buffer in a live acquisition
IMG_OTHER_INTERFACE	imgGetAttribute	Specifies the type of interface is not a plug-in board
IMG_PULSE_POLAR_ACTIVEH	imgPulseCreate	Specifies the polarity of a pulse as active high
IMG_PULSE_POLAR_ACTIVEL	imgPulseCreate	Specifies the polarity of a pulse as active low
IMG_SCALE_DIV2	imgGetAttribute imgSetAttribute	Specifies scaling by divisions of 2
IMG_SCALE_DIV4	imgGetAttribute imgSetAttribute	Specifies scaling by divisions of 4
IMG_SCALE_DIV8	imgGetAttribute imgSetAttribute	Specifies scaling by divisions of 8
IMG_SCALE_NONE	imgGetAttribute imgSetAttribute	Specifies no scaling
IMG_TRIG_ACTION_BUFFER	imgSessionTriggerDrive	Specifies each buffer is acquired based on a trigger
IMG_TRIG_ACTION_BUFLIST	imgSessionTriggerDrive	Specifies acquisition of buffer list starts on a trigger
IMG_TRIG_ACTION_CAPTURE	imgSessionTriggerDrive	Specifies trigger starts acquisition
IMG_TRIG_ACTION_NONE	imgSessionTriggerDrive	Specifies triggering is disabled
IMG_TRIG_DRIVE_AQ_DONE	imgSessionTriggerDrive	Specifies that the trigger line is driven on AQ_DONE

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_TRIG_DRIVE_AQ_IN_PROGRESS	imgSessionTriggerDrive	Specifies that the trigger line is driven on AQ_IN_PROGRESS
IMG_TRIG_DRIVE_ASSERTED	imgSessionTriggerDrive	Specifies to immediately drive the trigger line asserted
IMG_TRIG_DRIVE_DISABLED	imgSessionTriggerDrive	Specifies that the trigger line is not driven
IMG_TRIG_DRIVE_FRAME_DONE	imgSessionTriggerDrive	Specifies that the trigger line is driven with frame done
IMG_TRIG_DRIVE_FRAME_START	imgSessionTriggerDrive	Specifies that the trigger line is driven with frame start
IMG_TRIG_DRIVE_HSYNC	imgSessionTriggerDrive	Specifies that the trigger line is driven on HSYNC
IMG_TRIG_DRIVE_PIXEL_CLK	imgSessionTriggerDrive	Specifies that the trigger line is driven with pixel clock
IMG_TRIG_DRIVE_UNASSERTED	imgSessionTriggerDrive	Specifies to immediately drive the trigger line unasserted
IMG_TRIG_DRIVE_VSYNC	imgSessionTriggerDrive	Specifies that the trigger line is driven on VSYNC
IMG_TRIG_NONE	imgGetAttribute imgSetAttribute	Specifies no trigger for the session
IMG_TRIG_POLAR_ACTIVEH	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the polarity of a trigger as active HIGH
IMG_TRIG_POLAR_ACTIVEL	imgSessionTriggerConfigure imgSessionTriggerDrive imgSessionTriggerRead	Specifies the polarity of a trigger as active LOW
IMG_TRIGMODE_NONE	imgGetAttribute imgSetAttribute	Specifies a trigger mode of disabled. Triggers will not cause an acquisition to occur.

**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
IMG_TRIGMODE_NOREPEAT	imgGetAttribute imgSetAttribute	Specifies a trigger mode of no-repeat. Enabled triggers will cause an acquisition to occur on the buffers that have their buffer trigger attribute set. The sequence will not re-arm after the last buffer.
IMG_TRIGMODE_REPEAT	imgGetAttribute imgSetAttribute	Specifies a trigger mode of repeat. Enabled triggers will cause an acquisition to occur on the specified buffers, and the sequence automatically will re-arm to acquire the sequence again after the last buffer.
IMGPLOT_COLOR_RGB24	imgPlot	Specifies a 24-bit color RGB image
IMGPLOT_COLOR_RGB32	imgPlot	Specifies a 32-bit color RGB image
IMGPLOT_INVERT	imgPlot	Specifies to invert the image when plotted
IMGPLOT_MONO_10	imgPlot	Specifies a 10-bit monochrome image
IMGPLOT_MONO_12	imgPlot	Specifies a 12-bit monochrome image
IMGPLOT_MONO_14	imgPlot	Specifies a 14-bit monochrome image
IMGPLOT_MONO_16	imgPlot	Specifies a 16-bit monochrome image
IMGPLOT_MONO_32	imgPlot	Specifies a 32-bit monochrome image
IMGPLOT_MONO_8	imgPlot	Specifies an 8-bit monochrome image
INTERFACE_NAME_SIZE	imgInterfaceQueryNames	Specifies the size of each array element in the interface names array



**Table A-2.** Constants Summary (Continued)

Constant	Use With	Description
PULSE_MODE_SINGLE	imgPulseCreate	Specifies a single pulse that will generate one pulse on the assertion edge of the specified signal
PULSE_MODE_SINGLE_REARM	imgPulseCreate	Specifies a pulse that will generate a pulse on all assertion edges of the specified signal until imgPulseStop is called
PULSE_MODE_TRAIN	imgPulseCreate	Specifies a continuous pulse train that will generate a pulse until imgPulseStop is called
PULSE_TIMEBASE_100KHZ	imgPulseCreate	Specifies a 100 kHz timebase to use for pulse generation
PULSE_TIMEBASE_50MHZ	imgPulseCreate	Specifies a 50 MHz timebase to use for pulse generation
PULSE_TIMEBASE_PIXELCLK	imgPulseCreate	Specifies the incoming pixel clock from the camera to use as a timebase for pulse generation

# Status Codes

This appendix describes the status codes returned by NI-IMAQ.

Each NI-IMAQ function returns a status code that indicates whether the function was performed successfully. A summary of the status codes is listed in Table B-1.

**Table B-1.** Status Code Summary

Error Code	Status Name	Description
-1074397183	IMG_ERR_NCAP	Function not implemented
-1074397182	IMG_ERR_OVRN	Too many interfaces open
-1074397181	IMG_ERR_EMEM	Not enough memory to perform the operation
-1074397180	IMG_ERR_OSER	Operating system error occurred
-1074397179	IMG_ERR_PAR1	Function-specific, see function description
-1074397178	IMG_ERR_PAR2	Function-specific, see function description
-1074397177	IMG_ERR_PAR3	Function-specific, see function description
-1074397176	IMG_ERR_PAR4	Function-specific, see function description
-1074397175	IMG_ERR_PAR5	Function-specific, see function description
-1074397174	IMG_ERR_PAR6	Function-specific, see function description
-1074397173	IMG_ERR_PAR7	Function-specific, see function description
-1074397172	IMG_ERR_MXBF	Too many buffers already allocated
-1074397171	IMG_ERR_DLLE	DLL internal error, bad logic state
-1074397170	IMG_ERR_BSIZ	Buffer size used is too small for minimum acquisition frame
-1074397169	IMG_ERR_MXBI	Exhausted buffer IDs
-1074397168	IMG_ERR_ELCK	Cannot lock buffers down, no more memory

**Table B-1.** Status Code Summary (Continued)

<b>Error Code</b>	<b>Status Name</b>	<b>Description</b>
-1074397167	IMG_ERR_DISE	Error releasing the image buffer
-1074397166	IMG_ERR_BBUF	Bad buffer pointer in list
-1074397165	IMG_ERR_NLCK	Buffer list is not locked
-1074397164	IMG_ERR_NCAM	No camera defined for this channel
-1074397163	IMG_ERR_BINT	Bad interface
-1074397162	IMG_ERR_BROW	Rowbytes is less than region of interest
-1074397161	IMG_ERR_BROI	ROI width is greater than rowbytes
-1074397160	IMG_ERR_BCMF	Bad camera file (check syntax)
-1074397159	IMG_ERR_NVBL	Not successful because of hardware limitations
-1074397158	IMG_ERR_NCFG	Invalid action, no buffers configured for session
-1074397157	IMG_ERR_BBLF	Buffer list does not contain a valid final command
-1074397156	IMG_ERR_BBLE	Buffer list contains an invalid command
-1074397155	IMG_ERR_BBLB	A buffer list buffer is null
-1074397154	IMG_ERR_NAIP	No acquisition in progress
-1074397153	IMG_ERR_VLCK	Cannot get lock on video source
-1074397152	IMG_ERR_BDMA	Bad DMA transfer
-1074397151	IMG_ERR_AIOP	Cannot perform request, acquisition in progress
-1074397150	IMG_ERR_TIMO	Wait timed out, acquisition not complete
-1074397149	IMG_ERR_NBUF	No buffers available, too early in acquisition
-1074397148	IMG_ERR_ZBUF	Zero buffer size, no bytes filled
-1074397147	IMG_ERR_HLPR	Bad parameter to low-level—check attributes and high-level arguments

**Table B-1.** Status Code Summary (Continued)

<b>Error Code</b>	<b>Status Name</b>	<b>Description</b>
-1074397146	IMG_ERR_BTRG	Trigger loopback problem—cannot drive trigger with action enabled
-1074397145	IMG_ERR_NINF	No interface found
-1074397144	IMG_ERR_NDLL	Unable to load DLL (LabWindows/CVI only)
-1074397143	IMG_ERR_NFNC	Unable to find API function in DLL (LabWindows/CVI only)
-1074397142	IMG_ERR_NOSR	Unable to allocate system resources (LabWindows/CVI only)
-1074397141	IMG_ERR_BTAC	No trigger action—acquisition will time out
-1074397140	IMG_ERR_FIFO	FIFO overflow caused acquisition to halt
-1074397139	IMG_ERR_MLCK	Memory lock error, cannot perform acquisition
-1074397138	IMG_ERR_ILCK	Interface locked
-1074397137	IMG_ERR_NEPK	No external pixel clock
-1074397136	IMG_ERR_SCLM	Field scaling mode not supported
-1074397135	IMG_ERR_SCC1	Channel not set to 1 when using StillColor RGB acquisition
-1074397134	IMG_ERR_SMALLALLOC	Error during small buffer allocation
-1074397133	IMG_ERR_ALLOC	Error during large buffer allocation
-1074397132	IMG_ERR_BADCAMTYPE	Bad camera type; camera needs to be of type NTSC or PAL
-1074397131	IMG_ERR_BADPIXTYPE	Camera not supported; must be an 8-bit camera
-1074397130	IMG_ERR_BADCAMPARAM	Bad camera parameter in configuration file
-1074397129	IMG_ERR_PALKEYDTCT	PAL key detection error
-1074397128	IMG_ERR_BFRQ	Bad frequency values
-1074397127	IMG_ERR_BITP	Bad interface type

**Table B-1.** Status Code Summary (Continued)

<b>Error Code</b>	<b>Status Name</b>	<b>Description</b>
-1074397126	IMG_ERR_HWNC	Hardware not capable of supporting this function
-1074397125	IMG_ERR_SERIAL	Serial port error
-1074397124	IMG_ERR_MXPI	Exhausted pulse IDs
-1074397123	IMG_ERR_BPID	Bad pulse ID
-1074397121	IMG_ERR_SERIAL_TIMO	Serial transmit/receive timeout
-1074397120	IMG_ERR_PG_TOO_MANY	Too many pattern generation transitions defined
-1074397119	IMG_ERR_PG_BAD_TRANS	Bad pattern generation transition time
-1074397118	IMG_ERR_PLNS	Pulse not started
-1074397117	IMG_ERR_BPMD	Bad pulse mode
-1074397116	IMG_ERR_NSAT	Nonsettable attribute
-1074397115	IMG_ERR_HYBRID	Cannot mix system and onboard memory buffers
-1074397114	IMG_ERR_BADFILFMT	Pixel depth not supported with this file format
-1074397113	IMG_ERR_BADFILEXT	File extension not supported
-1074397112	IMG_ERR_NRTSI	Too many RTSI triggers mapped; you can use only four RTSI lines at once
-1074397111	IMG_ERR_MXTRG	Exhausted trigger resources
-1074397110	IMG_ERR_MXRC	Exhausted resource (general)
-1074397109	IMG_ERR_OOR	Parameter out of range
-1074397108	IMG_ERR_NPROG	FPGA not programmed
-1074397107	IMG_ERR_NEOM	Not enough onboard memory to perform the operation
-1074397106	IMG_ERR_BDTYPE	Bad display type; buffer cannot be displayed with <code>imgPlot</code>
-1074397105	IMG_ERR_THRDACC DEN	Thread denied access to function

**Table B-1.** Status Code Summary (Continued)

<b>Error Code</b>	<b>Status Name</b>	<b>Description</b>
-1074397104	IMG_ERR_BADFILWRT	Could not write the file
1073086470	IMG_WRN_OATTR	Warning, old attribute used
1073086469	IMG_WRN_BRST	Warning, bad quality colorburst (StillColor mode)
1073086468	IMG_WRN_BLK_G	Warning, unstable blanking reference (StillColor mode)
1073086467	IMG_WRN_ILCK	Warning, interface still locked
1073086466	IMG_WRN_CONF	Warning, change requires reconfiguration to take effect
1073086465	IMG_WRN_BCAM	Warning, corrupt camera file detected



---

# Technical Support Resources

This appendix describes the comprehensive resources available to you in the Technical Support section of the National Instruments Web site and provides technical support telephone numbers for you to use if you have trouble connecting to our Web site or if you do not have internet access.

## NI Web Support

---

To provide you with immediate answers and solutions 24 hours a day, 365 days a year, National Instruments maintains extensive online technical support resources. They are available to you at no cost, are updated daily, and can be found in the Technical Support section of our Web site at [www.natinst.com/support](http://www.natinst.com/support).

### Online Problem-Solving and Diagnostic Resources

- **KnowledgeBase**—A searchable database containing thousands of frequently asked questions (FAQs) and their corresponding answers or solutions, including special sections devoted to our newest products. The database is updated daily in response to new customer experiences and feedback.
- **Troubleshooting Wizards**—Step-by-step guides lead you through common problems and answer questions about our entire product line. Wizards include screen shots that illustrate the steps being described and provide detailed information ranging from simple getting started instructions to advanced topics.
- **Product Manuals**—A comprehensive, searchable library of the latest editions of National Instruments hardware and software product manuals.
- **Hardware Reference Database**—A searchable database containing brief hardware descriptions, mechanical drawings, and helpful images of jumper settings and connector pinouts.
- **Application Notes**—A library with more than 100 short papers addressing specific topics such as creating and calling DLLs, developing your own instrument driver software, and porting applications between platforms and operating systems.

## Software-Related Resources

- **Instrument Driver Network**—A library with hundreds of instrument drivers for control of standalone instruments via GPIB, VXI, or serial interfaces. You also can submit a request for a particular instrument driver if it does not already appear in the library.
- **Example Programs Database**—A database with numerous, non-shipping example programs for National Instruments programming environments. You can use them to complement the example programs that are already included with National Instruments products.
- **Software Library**—A library with updates and patches to application software, links to the latest versions of driver software for National Instruments hardware products, and utility routines.

## Worldwide Support

---

National Instruments has offices located around the globe. Many branch offices maintain a Web site to provide information on local services. You can access these Web sites from [www.natinst.com/worldwide](http://www.natinst.com/worldwide).

If you have trouble connecting to our Web site, please contact your local National Instruments office or the source from which you purchased your National Instruments product(s) to obtain support.

For telephone support in the United States, dial 512 795 8248. For telephone support outside the United States, contact your local branch office:

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20,  
Brazil 011 284 5011, Canada (Ontario) 905 785 0085,  
Canada (Québec) 514 694 8521, China 0755 3904939,  
Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24,  
Germany 089 741 31 30, Hong Kong 2645 3186, India 91805275406,  
Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970,  
Korea 02 596 7456, Mexico (D.F.) 5 280 7625,  
Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466,  
Norway 32 27 73 00, Singapore 2265886, Spain (Madrid) 91 640 0085,  
Spain (Barcelona) 93 582 0251, Sweden 08 587 895 00,  
Switzerland 056 200 51 51, Taiwan 02 2377 1200,  
United Kingdom 01635 523545



# Glossary

---

Prefix	Meaning	Value
p-	pico-	$10^{-12}$
n-	nano-	$10^{-9}$
$\mu$ -	micro-	$10^{-6}$
m-	milli-	$10^{-3}$
k-	kilo-	$10^3$
M-	mega-	$10^6$
G-	giga-	$10^9$
t-	tera-	$10^{12}$

## Numbers/Symbols

– negative of, or minus

$\Omega$  ohm

/ per

% percent

$\pm$  plus or minus

+ positive of, or plus

## A

A amperes

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 VSYNC) and a line count
active pixel region	the region of pixels actively being stored; defined by a pixel start (relative to HSYNC) and a pixel count
A/D	analog-to-digital
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>B</b>	
b	bit—one binary digit, either 0 or 1
B	byte—eight related bits of data, an eight-bit binary number; also used to denote the amount of memory required to store one byte of data
back porch	the area of the video signal between the rising edge of the horizontal sync signal and the active video information
black reference level	the level that represents the darkest an image can get. <i>See also</i> white reference level.

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

C	Celsius
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
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. <i>See also</i> 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

## 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 boards plugged into a computer, and possibly generating control signals with D/A and/or DIO boards in the same computer
dB	decibel—the unit for expressing a logarithmic measure of the ratio of two signal levels: $dB = 20 \log_{10} V_1/V_2$ , for signals in volts
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 <i>use the current default setting</i>
DIN	Deutsche Industrie Norme
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
drivers	software that controls a specific hardware device such as an IMAQ or DAQ device.
dynamic range	the ratio of the largest signal level a circuit can handle to the smallest signal level it can handle (usually taken to be the noise level), normally expressed in decibels

**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 the odd-numbered lines, 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; FIFOs are used on IMAQ devices to temporarily store incoming data until that data can be retrieved. For example, an analog input FIFO stores the results of A/D conversions until the data can be retrieved into system memory, a process that requires the servicing of interrupts and often the programming of the DMA controller. This process can take several milliseconds in some cases. During this time, data accumulates in the FIFO for future retrieval.
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
ft	feet
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; examples of functions are: $y = \text{COS}(x)$ <b>status</b> = AO_config( <b>board</b> , <b>channel</b> , <b>range</b> )

## G

gamma	the nonlinear change in the difference between the video signal's brightness level and the voltage level needed to produce that brightness
genlock	circuitry that aligns the video timing signals by locking together the horizontal, vertical, and color subcarrier frequencies and phases and generates a pixel clock to clock pixel data into memory for display or into another circuit for processing
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

h	hour
hardware	the physical components of a computer system, such as the circuit boards, plug-in boards, chassis, enclosures, peripherals, cables, and so on
horizontal sync	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. <i>See also</i> RGB.
Hz	hertz—the number of scans read or updates written per second

## I

IC	integrated circuit
ID	identification
IEEE	Institute of Electrical and Electronics Engineers
in.	inches

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 Basic, C or Pascal, by reading one line at a time and executing the specified operation. <i>See also</i> 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, and/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
<b>K</b>	
k	kilo—the standard metric prefix for 1,000, or $10^3$ , used with units of measure such as volts, hertz, and meters
K	kilo—the prefix for 1,024, or $2^{10}$ , used with B in quantifying data or computer memory
kbytes/s	a unit for data transfer that means 1,000 or $10^3$ bytes/s
Kword	1,024 words of memory

## 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	look-up table—a selection in the Measurement & Automation Explorer for IMAQ that contains formulas that let you implement simple imaging operations such as contrast enhancement, data inversion, gamma manipulation, or other nonlinear transfer functions

## M

m	meters
M	(1) Mega, the standard metric prefix for 1 million or $10^6$ , when used with units of measure such as volts and hertz; (2) mega, the prefix for 1,048,576, or $2^{20}$ , when used with B to quantify data or computer memory
MB	megabytes of memory
Mbytes/s	a unit for data transfer that means 1 million or $10^6$ bytes/s
memory buffer	<i>See</i> buffer.
memory window	continuous blocks of memory that can be accessed quickly 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 IMAQ hardware
noninterlaced	a video frame where all the lines are scanned sequentially, instead of 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. <i>See also</i> PAL.
NVRAM	nonvolatile RAM—RAM that is not erased when a device loses power or is turned off

**O**

operating system	base-level software that controls a computer, runs programs, interacts with users, and communicates with installed hardware or peripheral devices
------------------	---

**P**

PAL	Phase Alternation Line—one of the European video color standards; uses 625 lines per frame. <i>See also</i> 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 132 Mbytes/s
PFI	programmable function input
PGIA	programmable gain instrumentation amplifier
picture aspect ratio	the ratio of the active pixel region to the active line region; for standard video signals like RS-170 or CCIR, the full-size picture aspect ratio normally 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, but 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 HYSNCs; 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 incoming HSYNC signal
protocol	the exact sequence of bits, characters, and control codes used to transfer data between computers and peripherals through a communications channel
pts	points
<b>R</b>	
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	the smallest signal increment that can be detected by a measurement system; resolution can be expressed in bits, in proportions, or in percent 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	red, green, and blue—the three primary colors used to represent a color picture. An RGB camera is a camera that deliver three signals, one for each primary.
ribbon cable	a flat cable in which the conductors are side by side
ROI	region-of-interest—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 IMAQ and DAQ boards directly, by means of connectors on top of the boards, for precise synchronization of functions

## S

s	seconds
saturation	the richness of a color. A saturation of zero corresponds to no color, that is, a gray pixel. 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
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 board.
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
syntax	the set of rules to which statements must conform in a particular programming language
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 set up 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

## U

UV plane      *See* YUV.

## V

V      volts

VCO      voltage-controlled oscillator—an oscillator that changes frequency depending on a control signal; used in a PLL to generate a stable pixel clock

vertical sync      the synchronization pulse generated at the beginning of each video field that tells the video monitor when to start a new field

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, back porch, active pixel region, and a front porch

## 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 representing the coordinates in a color plane.

# Index

---

## A

acquisition functions, 4-1 to 4-12  
    imgMemLock, 4-2  
    imgMemUnlock, 4-3  
    imgSessionAbort, 4-4  
    imgSessionAcquire, 4-5  
    imgSessionConfigure, 4-6  
    imgSessionCopyArea, 4-7 to 4-8  
    imgSessionCopyBuffer, 4-9  
    imgSessionExamineBuffer, 4-10 to 4-11  
    imgSessionReleaseBuffer, 4-12  
acquisition window, 1-6  
architecture of NI-IMAQ (figure), 1-8  
area, 1-6  
arrays, 1-2  
attribute functions, 4-13 to 4-22  
    imgGetAttribute, 4-14  
    imgGetCameraAttributeNumeric, 4-15  
    imgGetCameraAttributeString, 4-16  
    imgSessionGetLostFramesList, 4-17  
    imgSessionSetUserLUT8bits, 4-18  
    imgSessionSetUserLUT16bits, 4-19  
    imgSetAttribute, 4-20  
    imgSetCameraAttributeNumeric, 4-21  
    imgSetCameraAttributeString, 4-22  
attribute summary (table), A-1 to A-7

## B

block diagram of NI-IMAQ architecture, 1-8  
buffer management functions, 4-23 to 4-33  
    imgCreateBuffer, 4-24 to 4-25  
    imgCreateBufList, 4-26  
    imgDisposeBuffer, 4-27  
    imgDisposeBufList, 4-28

imgGetBufferElement, 4-29 to 4-30  
imgSessionClearBuffer, 4-31  
imgSetBufferElement, 4-32 to 4-33

## C

code examples, 1-6  
constants summary (table), A-8 to A-18

## D

data types. *See* variable data types.  
documentation  
    conventions used in manual, *ix*  
    National Instruments documentation, *x*  
    related documentation, *x*

## E

examples of code, 1-6

## F

telephone support numbers, C-2  
functions  
    generic, 2-1 to 2-4  
    high-level, 3-1 to 3-40  
        grab functions, 3-5 to 3-9  
        miscellaneous functions, 3-35 to 3-40  
        ring and sequence functions,  
            3-10 to 3-15  
        signal I/O functions, 3-16 to 3-34  
        snap functions, 3-1 to 3-4  
    LabWindows/CVI function tree (table),  
        1-3 to 1-6

- low-level, 4-1 to 4-42
  - acquisition functions, 4-1 to 4-12
  - attribute functions, 4-13 to 4-22
  - buffer management functions, 4-23 to 4-33
  - interface functions, 4-34 to 4-36
  - utility functions, 4-37 to 4-42

## G

- generic functions, 2-1 to 2-4
  - imgClose, 2-4
  - imgInterfaceOpen, 2-2
  - imgSessionOpen, 2-3
- geometric definitions, 1-6 to 1-7
  - acquisition window, 1-6
  - area, 1-6
  - region of interest, 1-6
  - relationship between terms (figure), 1-7
- grab functions, 3-5 to 3-9
  - imgGrab, 3-7
  - imgGrabArea, 3-8 to 3-9
  - imgGrabSetup, 3-6

## H

- hardware abstraction layer, 1-8
- high-level functions, 3-1 to 3-40
  - grab functions, 3-5 to 3-9
  - miscellaneous functions, 3-35 to 3-40
  - ring and sequence functions, 3-10 to 3-15
  - signal I/O functions, 3-16 to 3-34
  - snap functions, 3-1 to 3-4
    - imgSnap, 3-2
    - imgSnapArea, 3-3 to 3-4

## I

- imgClose function, 2-4
- imgCreateBuffer function, 4-24 to 4-25
- imgCreateBufList function, 4-26
- imgDisposeBuffer function, 4-27
- imgDisposeBufList function, 4-28
- imgGetAttribute function, 4-14
- imgGetBufferElement function, 4-29 to 4-30
- imgGetCameraAttributeNumeric function, 4-15
- imgGetCameraAttributeString function, 4-16
- imgGrab function, 3-7
- imgGrabArea function, 3-8 to 3-9
- imgGrabSetup function, 3-6
- imgInterfaceOpen function, 2-2
- imgInterfaceQueryNames function, 4-35
- imgInterfaceReset function, 4-36
- imgMemLock function, 4-2
- imgMemUnlock function, 4-3
- imgPlot function, 4-38 to 4-39
- imgPulseCreate function, 3-28 to 3-30
- imgPulseDispose function, 3-31
- imgPulseRate function, 3-32
- imgPulseStart function, 3-33
- imgPulseStop function, 3-34
- imgRingSetup function, 3-11
- imgSequenceSetup function, 3-12 to 3-13
- imgSessionAbort function, 4-4
- imgSessionAcquire function, 4-5
- imgSessionClearBuffer function, 4-31
- imgSessionConfigure function, 4-6
- imgSessionCopyArea function, 4-7 to 4-8
- imgSessionCopyBuffer function, 4-9
- imgSessionExamineBuffer function, 4-10 to 4-11
- imgSessionGetBufferSize function, 3-40
- imgSessionGetLostFramesList function, 4-17
- imgSessionGetROI function, 3-39
- imgSessionLineTrigSource function, 3-19
- imgSessionOpen function, 2-3
- imgSessionReleaseBuffer function, 4-12
- imgSessionSaveBufferEx function, 4-40 to 4-41
- imgSessionSetROI function, 3-37 to 3-38
- imgSessionSetUserLUT8bits, 4-18

3-17 to 3-18  

 3-21 to 3-22  



 3-26 to 3-27  



 function, 4-21  




interface functions, 4-33 to 4-37  
 imgInterfaceQueryNames, 4-35  
 imgInterfaceReset, 4-36

## L

LabVIEW software, 1-2  
 LabWindows/CVI software  
 function tree for image acquisition (table),  
 1-3 to 1-6  
 programming considerations, 1-3  
 low-level functions, 4-1 to 4-42  
 acquisition functions, 4-1 to 4-12  
 attribute functions, 4-13 to 4-22  
 buffer management functions,  
 4-23 to 4-33  
 interface functions, 4-34 to 4-36  
 utility functions, 4-37 to 4-42

## M

manual. *See* documentation.  
 miscellaneous high-level functions,  
 3-35 to 3-40  
 imgSessionGetBufferSize, 3-40  
 imgSessionGetROI, 3-39  
 imgSessionSetROI, 3-37 to 3-38  
 imgSessionStatus, 3-36

## N

NI-IMAQ attributes (table), A-1 to A-7

## P

programming language considerations,  
 1-2 to 1-6  
 code examples, 1-6  
 LabVIEW, 1-2  
 LabWindows/CVI, 1-3 to 1-6  
 third-party programming  
 environments, 1-6

## R

region of interest, 1-6  
 ring and sequence functions, 3-10 to 3-15  
 imgRingSetup, 3-11  
 imgSequenceSetup, 3-12 to 3-13  
 imgSessionStartAcquisition, 3-14  
 imgSessionStopAcquisition, 3-15

## S

sequence functions. *See* ring and sequence  
 functions.  
 signal I/O functions, 3-16 to 3-34  
 imgPulseCreate, 3-28 to 3-30  
 imgPulseDispose, 3-31  
 imgPulseRate, 3-32  
 imgPulseStart, 3-33

- imgPulseStop, 3-34
- imgSessionLineTrigSource, 3-19
- imgSessionTriggerClear, 3-20
- imgSessionTriggerConfigure,  
3-17 to 3-18
- imgSessionTriggerDrive, 3-21 to 3-22
- imgSessionTriggerRead, 3-23
- imgSessionWaitSignal, 3-24 to 3-25
- imgSessionWaitSignalAsync,  
3-26 to 3-27
- snap functions, 3-1 to 3-4
  - imgSnap, 3-2
  - imgSnapArea, 3-3 to 3-4
- status codes, B-1 to B-5
  - format, 1-1

## T

- technical support, C-1 to C-2
- telephone support numbers, C-2

## U

- utility functions, 4-37 to 4-42
  - imgPlot, 4-38 to 4-39
  - imgSessionSaveBufferEx, 4-40 to 4-41
  - imgShowError, 4-42

## V

- variable data types
  - arrays, 1-2
  - primary types (table), 1-2
  - Windows 95 and Windows NT, 1-1

## W

- Web site support, C-1