IMAQ™

NI-IMAQ™ Function Reference Manual

NATIONAL INSTRUMENTS™

June 1998 Edition
Part Number 371447A-01

Internet Support
E-mail: support@natinst.com
FTP Site: ftp.natinst.com
Web Address: http://www.natinst.com

Bulletin Board Support
BBS United States: 512 794 5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

Fax-on-Demand Support
512 418 1111

Telephone Support (USA)
Tel: 512 795 8248
Fax: 512 794 5678

International Offices
Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 288 3336,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, 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, Israel 03 6120092, Italy 02 413091,
Japan 03 5472 2970, Korea 02 596 7456, Mexico 5 520 2635, Netherlands 0348 433466, Norway 32 84 84 00,
Singapore 2265886, Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
United Kingdom 01635 523545

National Instruments Corporate Headquarters
6504 Bridge Point Parkway Austin, Texas 78730-5039 USA Tel: 512 794 0100

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 manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Contents

About This Manual

Chapter 1
Introduction

Chapter 2
Generic Functions

Chapter 3
High-Level Functions

Chapter 4
Low-Level Functions

Figures

Tables

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.

How to Use the NI-IMAQ Manual Set

Before using this manual, you should begin by reading the setup and test document included with your hardware, the NI-IMAQ release notes, and the *NI-IMAQ User Manual*. These documents contain information about how to install your software and hardware. Then read Chapter 1, *Introduction*, of your hardware user manual, which contains a flowchart that illustrates the sequence of steps you should take to learn about and get started with NI-IMAQ.

When you are familiar with the material in the *NI-IMAQ User Manual*, use this manual, which contains detailed descriptions of the NI-IMAQ functions.

Organization of This Manual

The *NI-IMAQ Function Reference Manual* is organized as follows:

- Chapter 1, *Introduction*, contains important information about how to apply the function descriptions in this manual to your programming language and environment.
- Chapter 2, *Generic Functions*, contains a detailed explanation of each generic NI-IMAQ function. The functions are arranged according to the order in which you will use them.
- Chapter 3, *High-Level Functions*, 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.

- Chapter 4, *Low-Level Functions*, 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.
- Appendix A, *Attributes and Constants*, describes the attributes and constants used by NI-IMAQ.
- Appendix *B*, *Status Codes*, describes the status codes returned by NI-IMAQ.
- Appendix C, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used 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» Substitute Fonts** directs you to pull down the **File** menu, select the **Page Setup** item, select **Options**, and finally select the **Substitute Fonts** options from the last dialog box.
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.
bold	Bold text denotes parameters, menus, menu items, or dialog box buttons or options.
bold italic	Bold italic text denotes a note, caution, or warning.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept. This font also denotes text for which you supply the appropriate word or value, such as in Windows 3.*x*.

`monospace`	Lowercase text in this font denotes text or characters that are to be literally 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, variables, filenames, and extensions, and for statements and comments taken from program code.
`monospace italic`	Italic text in this font denotes that you must enter the appropriate words or values in the place of these items.

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 document contains information that you may find helpful as you read this manual:

- Your computer technical reference manual

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix C, *Customer Communication*, at the end of this manual.

1

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 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 95 and Windows NT. 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`
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»Image Acquisition**. 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

LabWindows/CVI Function Panel	NI-IMAQ Function
Generic Functions	
Close Object	`imgClose`
Interface Open	`imgInterfaceOpen`
Session Open	`imgSessionOpen`
High-Level Snap Functions	
Snap	`imgSnap`
Snap Area	`imgSnapArea`
High-Level Grab Functions	
Grab	`imgGrab`
Grab Area	`imgGrabArea`
Grab Setup	`imgGrabSetup`
High-Level Ring and Sequence Functions	
Ring Setup	`imgRingSetup`
Sequence Setup	`imgSequenceSetup`
Session Start Acquisition	`imgSessionStartAcquisition`
Session Stop Acquisition	`imgSessionStopAcquisition`

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

LabWindows/CVI Function Panel	NI-IMAQ Function
High-Level Signal I/O Functions	
Pulse Create	`imgPulseCreate`
Pulse Dispose	`imgPulseDispose`
Pulse Rate	`imgPulseRate`
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`
Line Trigger Source	`imgSessionLineTrigSource`
High-Level Miscellaneous Functions	
Session Get Buffer Size	`imgSessionGetBufferSize`
Session Get ROI	`imgSessionGetROI`
Session Set ROI	`imgSessionSetROI`
Session Status	`imgSessionStatus`
Low-Level Acquisition Functions	
Lock Buffer List Memory	`imgMemLock`
Session Abort	`imgSessionAbort`
Session Acquire	`imgSessionAcquire`
Session Configure	`imgSessionConfigure`
Session Copy Area	`imgSessionCopyArea`

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

LabWindows/CVI Function Panel	NI-IMAQ Function
Session Copy Buffer	imgSessionCopyBuffer
Session Examine Buffer	imgSessionExamineBuffer
Session Release Buffer	imgSessionReleaseBuffer
Unlock Buffer List Memory	imgMemUnlock
Low-Level Attribute Functions	
Get Attribute	imgGetAttribute
Get Camera Attribute Numeric	imgGetCameraAttributeNumeric
Get Camera Attribute String	imgGetCameraAttributeString
Set Attribute	imgSetAttribute
Set Camera Attribute Numeric	imgSetCameraAttributeNumeric
Set Camera Attribute String	imgSetCameraAttributeString
Get Lost Frames List	imgSessionGetLostFramesList
Low-Level Buffer Management Functions	
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
Low-Level Interface Functions	
Interface Lock	imgInterfaceLock
Interface Query Names	imgInterfaceQueryNames

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

LabWindows/CVI Function Panel	NI-IMAQ Function
Interface Reset	`imgInterfaceReset`
Interface Unlock	`imgInterfaceUnlock`
Low-Level Utility Functions	
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.

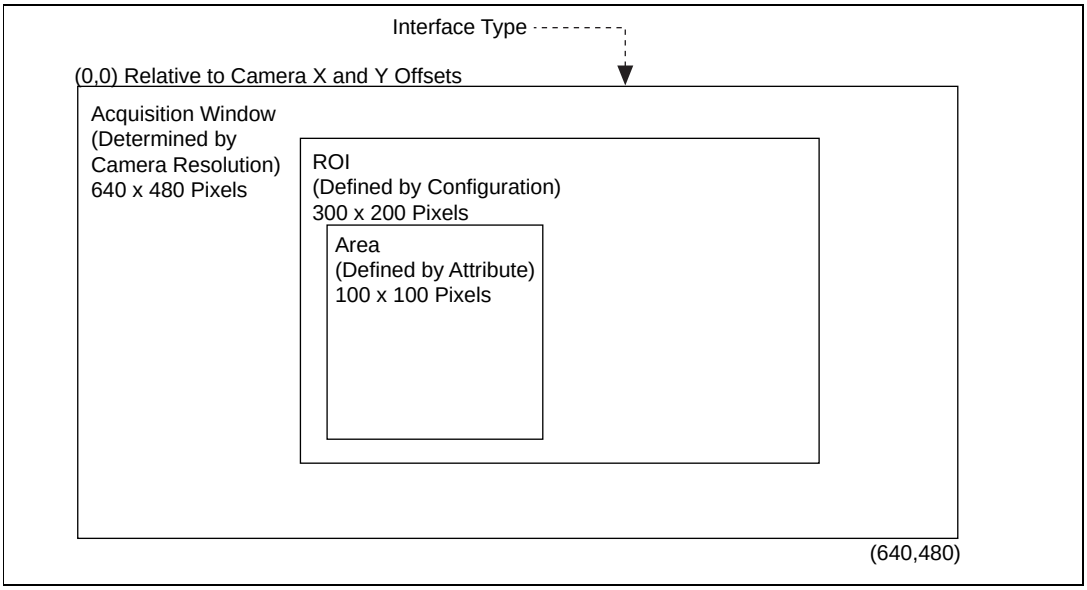

Figure 1-1. Geometric Relationship

Architecture

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

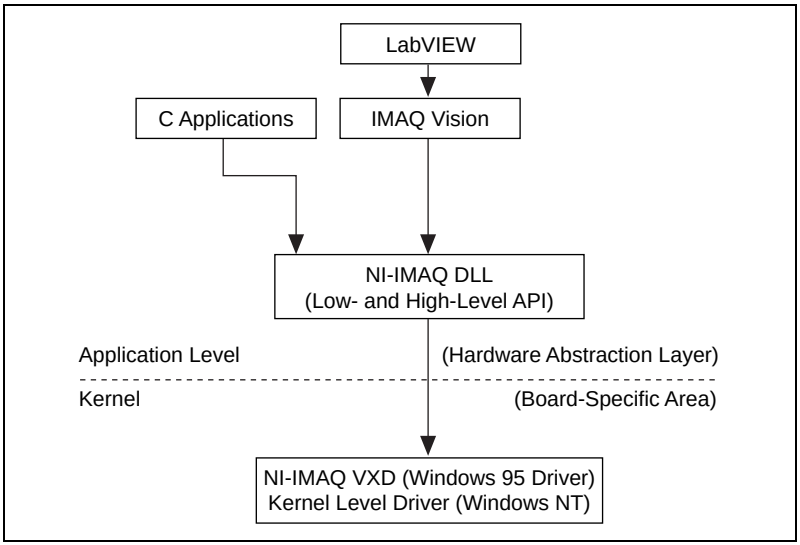

Figure 1-2. NI-IMAQ Architecture

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.

2

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`**(Int8* interface_name, INTERFACE_ID* pifid)**

Purpose

Opens by name an interface as specified in the IMAQ Configuration Utility. If it is successful, this function returns an INTERFACE_ID.

Parameters

Name	Type	Direction	Description
interface_name	Int8*	input	null terminates name of interface to open
pifid	INTERFACE_ID*	output	pointer to INTERFACE_ID type variable
rval	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
ifid	INTERFACE_ID	input	interface ID to open session
psid	SESSION_ID*	output	pointer to a session ID
rval	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
void_id	uInt32	input	session or interface ID
freeResources	uInt32	input	cleanup flag
rval	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.***

3

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
sid	SESSION_ID	input	session ID
bufAddr	void*	input	pointer to the buffer address
rval	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
sid	SESSION_ID	input	session ID
bufAddr	void*	input	pointer to the buffer address
top	uInt32	input	top ordinate of the first pixel transferred
left	uInt32	input	left ordinate of the first pixel transferred
height	uInt32	input	height of rectangle to transfer
width	uInt32	input	width of the rectangle to transfer
rowPixels	uInt32	input	used in calculating the address of the next line
rval	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
sid	SESSION_ID	input	session ID
startNow	uInt32	input	start acquisition after setup completes
rval	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
sid	SESSION_ID	input	session ID
bufAddr	void*	input	pointer to the buffer address
syncOnVB	uInt32	input	vertical blank flag
rval	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
sid	SESSION_ID	input	session ID
bufAddr	void*	input	pointer to the buffer address
syncOnVB	uInt32	input	vertical blank flag
top	uInt32	input	top ordinate of the first pixel transferred
left	uInt32	input	left ordinate of the first pixel transferred
height	uInt32	input	height of rectangle to transfer
width	uInt32	input	width of the rectangle to transfer
rowPixels	uInt32	input	used in calculating the address of the next line
rval	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.

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
sid	SESSION_ID	input	session ID
numberBuffer	uInt32	input	number of buffers used in the ring session
bufferList[]	void*	input	array of buffer pointers
skipCount	uInt32	input	number of frames or field to skip before each acquisition
startnow	uInt32	input	start acquisition after setup completes
rval	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
sid	SESSION_ID	input	session ID
numberBuffer	uInt32	input	number of buffers used in the ring session
bufferList[]	void*	input	array of buffer pointers
skipCount[]	uInt32	input	array containing the number of frames or field to skip before each acquisition
startNow	uInt32	input	start acquisition after setup completes
async	uInt32	input	indicates either a synchronous or an asynchronous acquisition
rval	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
sid	SESSION_ID	input	session ID
rval	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
sid	SESSION_ID	input	session ID
rval	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
sid	SESSION_ID	input	session ID
trig_num	uInt32	input	trigger line on which pulse is generated
trig_polarity	uInt32	input	polarity
time_out	uInt32	input	time to wait for image
trig_action	uInt32	input	start acquisition based on trigger, start each buffer list based on trigger, trigger each buffer
rval	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_TRIG1
IMG_EXT_TRIG2
IMG_EXT_TRIG3
IMG_EXT_RTSI0
IMG_EXT_RTSI1
IMG_EXT_RTSI2
IMG_EXT_RTSI3
IMG_EXT_RTSI4
IMG_EXT_RTSI5
IMG_EXT_RTSI6

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
sid	SESSION_ID	input	session ID
trig_source	uInt32	input	source of the trigger signal
trig_polarity	uInt32	input	polarity of the trigger line
skip_number	uInt32	input	number of triggers to skip
rval	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_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	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 skip between lines. 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 10.

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

Name	Type	Direction	Description
sid	SESSION_ID	input	session ID
rval	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
sid	SESSION_ID	input	session ID
trig_num	uInt32	input	trigger line on which pulse is generated
trig_polarity	uInt32	input	polarity
trig_drive	uInt32	input	HIGH, LOW, internal status signals
rval	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_TRIG1
IMG_EXT_TRIG2
IMG_EXT_TRIG3
IMG_EXT_RTSI0
IMG_EXT_RTSI1
IMG_EXT_RTSI2
IMG_EXT_RTSI3
IMG_EXT_RTSI4
IMG_EXT_RTSI5
IMG_EXT_RTSI6

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 specifies 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
sid	SESSION_ID	input	session ID
trig_num	uInt32	input	trigger line on which pulse is generated
trig_polarity	uInt32	input	polarity
status	uInt32*	output	value of trigger line
rval	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_TRIG1
IMG_EXT_TRIG2
IMG_EXT_TRIG3
IMG_EXT_RTSI0
IMG_EXT_RTSI1
IMG_EXT_RTSI2
IMG_EXT_RTSI3
IMG_EXT_RTSI4
IMG_EXT_RTSI5
IMG_EXT_RTSI6

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
sid	SESSION_ID	input	session ID
signal	uInt32	input	signal which will cause the pulse to be generated
signal_pol	uInt32	input	polarity of signal
timeout	uInt32	input	time to wait for image
rval	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_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 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
sid	SESSION_ID	input	session ID
signal	uInt32	input	signal which will cause the pulse to be generated
signal_pol	uInt32	input	polarity of signal
funcptr	CALL_BACK_PTR	input	callback function
callback_data	void*	input	four-byte value passed to the callback function
rval	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 valued 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:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BPMD	bad pulse mode
IMG_ERR_EMEM	not enough memory to create status thread
IMG_ERR_OSER	bad thread creation
IMG_ERR_PAR1	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
timebase	uInt32	input	timebase value
delay	uInt32	input	interval before the pulse
width	uInt32	input	interval of the pulse
signal_source	uInt32	input	signal that triggers the pulse generation
signal_polarity	uInt32	input	polarity of the signal
output	uInt32	input	trigger line on which the pulse is generated
output_polarity	uInt32	input	the polarity of the pulse output
pulse_mode	uInt32	input	indicates if the pulse is repeated
plsID	PULSE_ID*	input	pulse ID created and configured with pulse functions
rval	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:

delay × (**timebase** 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:

width × (**timebase** 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_RTSI0
IMG_AQ_DONE	IMG_EXT_RTSI1
IMG_FRAME_START	IMG_EXT_RTSI2
IMG_FRAME_DONE	IMG_EXT_RTSI3
IMG_EXT_TRIG0	IMG_EXT_RTSI4
IMG_EXT_TRIG1	IMG_EXT_RTSI5
IMG_EXT_TRIG2	IMG_EXT_RTSI6

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_RTSI2
IMG_EXT_TRIG1	IMG_EXT_RTSI3
IMG_EXT_TRIG2	IMG_EXT_RTSI4
IMG_EXT_TRIG2	IMG_EXT_RTSI5

IMG_EXT_RTSI0 IMG_EXT_RTSI6
IMG_EXT_RTSI1

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
plsID	PULSE_ID	input	pulse ID
rval	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
delaytime	uInt32	input	interval before the pulse in microseconds
widthtime	uInt32	input	interval of the pulse in microseconds
delay	uInt32*	output	interval before the pulse in cycles
width	uInt32*	output	interval of the pulse in cycles
timebase	uInt32*	output	timebase source
rval	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
pid	PULSE_ID	input	pulse ID
sid	SESSION_ID	input	session ID
rval	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
pid	PULSE_ID	input	pulse ID
rval	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
sid	SESSION_ID	input	session ID
status	uInt32*	input	current session status
bufferNumber	uInt32*	input	pointer to an area of memory reserved for the current buffer number
rval	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 origin and dimension. 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
sid	SESSION_ID	input	session ID
top	uInt32	output	top ordinate of the first pixel transferred
left	uInt32	output	left ordinate of the first pixel transferred
height	uInt32	output	height of rectangle to transfer
width	uInt32	output	width of the rectangle to transfer
rval	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 origin and dimensions.

Parameters

Name	Type	Direction	Description
sid	SESSION_ID	input	session ID
top	uInt32*	output	current value of IMG_ATTR_ROI_TOP
left	uInt32*	output	current value of IMG_ATTR_ROI_LEFT
height	uInt32*	output	current value of IMG_ATTR_ROI_HEIGHT
width	uInt32*	output	current value of IMG_ATTR_ROI_WIDTH
rval	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
sid	SESSION_ID	input	session ID
sizeNeeded	uInt32*	output	buffer size
rval	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.

4

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
bid	BUFLIST_ID	input	buffer list ID
rval	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
bid	BUFLIST_ID	input	buffer list ID
rval	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
sid	SESSION_ID	input	session ID
buf_num	uInt32*	output	pointer to the last valid buffer number
rval	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
sid	SESSION_ID	input	session ID
async	uInt32	input	asynchronous flag
callback	CALL_BACK_PTR	input	user completion routine
rval	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
sid	SESSION_ID	input	session ID
buflist	BUFLIST_ID	input	a valid BUFLIST_ID
rval	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:

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

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
sid	SESSION_ID	input	session ID
buf_num	uInt32	input	element number of buffer to copy
top	uInt32	input	pixel value to set data to top vertical coordinate of area
left	uInt32	input	left horizontal coordinate of area
width	uInt32	input	width of area
height	uInt32	input	height of area
buffer	Ptr	input	pointer to user image memory
rowPixels	uInt32	input	used in calculating the address of the next line
vsync	uInt32	input	wait until the next vertical blank to perform the copy
rval	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_PAR3	invalid coordinate
IMG_ERR_PAR4	invalid coordinate
IMG_ERR_PAR5	invalid height
IMG_ERR_PAR6	invalid width
IMG_ERR_PAR7	null pointer
IMG_ERR_PAR8	invalid rowPixels

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
sid	SESSION_ID	input	session ID
buf_num	uInt32	input	element number of buffer to copy
buffer	uInt8*	input	pointer to user buffer
vsync	uInt32	input	wait until the next vertical blank to perform
rval	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
sid	SESSION_ID	input	session ID
whichBuffer	uInt32	input	identifies which buffer to get
bufferNumber	uInt32*	input	element number of the returned buffer
bufferAddr	uInt32*	input	a pointer to an address to store the address of the locked buffer
rval	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
sid	SESSION_ID	input	session ID
bufferItem	uInt32	input	buffer number to release
rval	Int32	output	status

Parameter Discussion

sid is a valid SESSION_ID type variable.

bufferItem indicates the buffer number to release as returned by `imgSessionExamineBuffer`.

rval returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_PAR1	invalid SESSION_ID
IMG_ERR_PAR2	invalid element number

Attribute Functions

Attribute functions include `imgGetAttribute`, `imgGetCameraAttributeNumeric`, `imgGetCameraAttributeString`, `imgSessionGetLostFramesList`, `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
void_id	uInt32	input	session or interface ID
type	uInt32	input	attribute type
value	void*	input/ output	pointer to a place attribute value
rval	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
sid	SESSION_ID	input	session ID
attributeString	int8*	input	string containing attribute name
currentValueNumeric	double*	input	pointer to place attribute value
rval	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
sid	SESSION_ID	input	session ID
attributeString	int8*	input	string containing attribute name
currentValueString	int8*	input	string containing attribute value
sizeofCurrentValueString	uInt32	input	size of **currentValueString**
rval	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 **currentValueSring**.

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
sid	SESSION_ID	input	session ID
framelist	uInt32*	input	pointer to cumulative frame number array
numEntries	uInt32	input	size of **framelist** array
rval	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 **framelist**

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
void_id	uInt32	input	session or interface ID
type	uInt32	input	attribute type
value	uInt32	input	new attribute value
rval	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	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
sid	SESSION_ID	input	session ID
attributeString	int8*	input	string containing attribute name
newValueNumeric	double	input	pointer to place attribute value
rval	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
sid	SESSION_ID	input	session ID
attributeString	int8*	input	string containing attribute name
newValueString	int8*	input	string containing attribute value
rval	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`, `imgSetArrayPointerValue`, 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 you do not pass a buffer size, 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
sid	SESSION_ID	input	session ID
where	uInt32	input	indicates where the buffer should be stored
bufferSize	uInt32	input	size of buffer to create
bufPtrAddr	void*	output	pointer to return the buffer address
rval	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`.

Parameters

Name	Type	Direction	Description
numElements	uInt32	input	number of entries in the buffer list
bid	BUFLIST_ID*	output	pointer to the new buffer list
rval	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
buffPtrAddr	void*	input	pointer to a buffer
rval	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
bid	BUFLIST_ID	input	valid BUFLIST_ID
freeResources	uInt32	input	determines whether buffers and buffer list will be disposed or just the buffer list
rval	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
bid	BUFLIST_ID	input	valid BUFLIST_ID
element	uInt32	input	element number the buffer list
itemType	uInt32	input	element item to get
itemValue	void*	input	new value for item
rval	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_COMMAND
IMG_BUFF_SIZE
IMG_BUFF_SKIPCOUNT
IMG_BUFF_CHANNEL

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
sid	SESSION_ID	input	session ID
buf_num	uInt32	input	element number of the buffer to clear
pixel_value	uInt8	input	pixel value to set data to
rval	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

imgSetArrayPointerValue

Format

rval = `imgSetArrayPointerValue`**(uInt32* pl, uInt32 index, uInt32* mglPointers)**

Purpose

A Visual Basic helper function for constructing an array of 32-bit pointers.

Parameters

Name	Type	Direction	Description
pl	uInt32*	input	32-bit pointer
index	uInt32	input	element number to set
mglPointers	uInt32*	input	array of 32-bit pointers
rval	Int32	output	status

Parameter Discussion:

pl is a 32-bit pointer.

index is the array element in **mglPointers** to set with **pl**.

mglPointers is a pointer to an array of 32-bit pointer values.

rval returns IMG_ERR_GOOD if no error occurs.

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
bid	BUFLIST_ID	input	a valid BUFLIST_ID
element	uInt32	input	element number the buffer list
itemType	uInt32	input	element item to set
itemValue	uInt32	input	new value for item
rval	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_CHANNEL
IMG_BUFF_COMMAND
IMG_BUFF_TRIGGER
IMG_BUFF_SIZE
IMG_BUFF_NUMBUFS
IMG_BUFF_SKIPCOUNT

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 `imgInterfaceLock`, `imgInterfaceQueryNames`, `imgInterfaceReset`, and `imgInterfaceUnlock`.

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.

imgInterfaceLock

Format

rval = `imgInterfaceLock`**(INTERFACE_ID ifid)**

Purpose

Locks a logical interface so that another process cannot use it. To unlock an interface, call `imgInterfaceUnlock`. Each call to `imgInterfaceLock` causes the lock count associated with this interface to be incremented by one.

Parameters

Name	Type	Direction	Description
ifid	INTERFACE_ID	input	interface ID
rval	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_ILCK	interface locked by another process
IMG_ERR_PAR1	bad INTERFACE_ID
IMG_WRN_ILCK	warning, interface still locked

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
index	uInt32	input	interface number to obtain
queryName	Int8*	input	pointer to an array of strings
rval	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
ifid	INTERFACE_ID	output	interface ID
rval	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

imgInterfaceUnlock

Format

rval = `imgInterfaceUnlock`**(INTERFACE_ID ifid)**

Purpose

Unlocks a logical interface, allowing another process to use it. Each call to `imgInterfaceUnlock` causes the lock count associated with this interface to be decremented by one. If the interface is still locked after this call is made, the function will return the warning, IMG_WRN_ILCK.

Parameters

Name	Type	Direction	Description
ifid	INTERFACE_ID	input	interface ID
rval	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_ILCK	interface locked by another process
IMG_ERR_PAR1	bad INTERFACE_ID
IMG_WRN_ILCK	warning, interface still locked

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
window	GUIHNDL	input	window handle
buffer	void*	input	pointer to video data
leftBufOffset	uInt32	input	x-offset into the buffer to start plotting
topBufOffset	uInt32	input	y-offset into the buffer to start plotting
xsize	uInt32	input	width of the image
ysize	uInt32	input	height of the image
xpos	uInt32	input	x-position to start plotting in the window
ypos	uInt32	input	y-position to start plotting in the window
flags	uInt32	input	set display property
rval	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
sid	SESSION_ID	input	session ID
buffer	void*	input	buffer to save
file_name	Int8*	input	null terminated string describing a file name (optional path)
rval	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.

rval returns the following status codes:

IMG_ERR_GOOD	no error occurred
IMG_ERR_BADFILEXT	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, Int8* text)**

Purpose

Returns a null terminated string describing the error code.

Parameters

Name	Type	Direction	Description
error	IMG_ERR	input	a valid error code
text	Int8*	input	pointer to a character array of no less than 255 bytes
rval	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

A

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* 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	Imd.	R/W	Description
IMG_ATTR_ACQ_IN_PROGRESS	S	Yes	R	Returns TRUE if an acquisition is in progress on the camera associated with this session
IMG_ATTR_ACQUIRE_FIELD	S	Yes	R/W	Sets the field acquired when IMG_ATTR_FRAME_FIELD is set to FIELD_MODE. Possible values are: IMG_ACQUIRE_ODD IMG_ACQUIRE_EVEN IMG_ACQUIRE_ALL The IMAQ PCI-1408 supports only IMG_ACQUIRE_ALL.
IMG_ATTR_ACQWINDOW_HEIGHT	S	No	R/W	Get/set the acquisition window height of the camera/channel associated with this session
IMG_ATTR_ACQWINDOW_LEFT	S	No	R/W	Get/set the acquisition window left of the camera/channel associated with this session
IMG_ATTR_ACQWINDOW_TOP	S	No	R/W	Get/set the acquisition window top of the camera/channel associated with this session

Table A-1. Attribute Summary (Continued)

Attribute	Type	Imd.	R/W	Description
IMG_ATTR_ACQWINDOW_WIDTH	S	No	R/W	Get/set the acquisition window width of the camera/channel associated with this session
IMG_ATTR_BITSPERPIXEL	S	Yes	R	Returns the bits per pixel value of the camera/channel associated with this session
IMG_ATTR_BLACK_REF_VOLT	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	S	Yes	R	Returns the bytes per pixel value of the camera/channel associated with this session
IMG_ATTR_CHROMA_FILTER	S	Yes	R/W	Set/get the antichrominance filter to be used. Values are: IMG_FILTER_NONE IMG_FILTER_NTSC IMG_FILTER_PAL
IMG_ATTR_COLOR	I	Yes	R	Returns TRUE if the interface board is color-capable
IMG_ATTR_COLOR_AVG_COUNT	S	Yes	R/W	Set/get the number of color images to be acquired and averaged for one output image (1–128). Default value is 1.
IMG_ATTR_COLOR_BRIGHTNESS	S	Yes	R/W	Adjusts the brightness of the image. The unit is IRE (percentage of the white level). Default value is 0.
IMG_ATTR_COLOR_CONTRAST	S	Yes	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. (For example, an 8-bit image would be centered around 128.) Default value is 1.
IMG_ATTR_COLOR_GAIN	S	Yes	R/W	Set/get 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	S	Yes	R/W	Set/get the hardware black reference of your IMAQ device when StillColor is selected. Values are 0 V to 1.26 V.

Table A-1. Attribute Summary (Continued)

Attribute	Type	Imd.	R/W	Description
IMG_ATTR_COLOR_HUE_OFFS_ANGLE	S	Yes	R/W	Set/get the offset angle for the hue calculation. A value of 0 (default) results in a red color to toggle between 0 and max (255 or 32,767). Changing this value will move the toggling point to other colors. The unit is degrees and corresponds to the rotation angle in the chromaticity space. Color hue offset angle
IMG_ATTR_COLOR_IMAGE_REP	S	Yes	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_RGB48 IMG_COLOR_REP_RGB24 IMG_COLOR_REP_RGB16 IMG_COLOR_REP_HSL32 IMG_COLOR_REP_HSI32
IMG_ATTR_COLOR_IMAGE_X_SHIFT	S	Yes	R/W	Color acquisition left shift
IMG_ATTR_COLOR_IMAGE_X_SHIFT_REF	S	Yes	R/W	Color acquisition left shift reference
IMG_ATTR_COLOR_LOW_REF_VOLT	S	Yes	R/W	Set/get the hardware white reference of your IMAQ device when StillColor is selected. Values are 0 V to 1.26 V.
IMG_ATTR_COLOR_MODE	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	S	Yes	R/W	Set/get the enabling of the NTSC setup compensation (StillColor NTSC only): Disabled (0) Enabled (1)

Table A-1. Attribute Summary (Continued)

Attribute	Type	Imd.	R/W	Description
IMG_ATTR_COLOR_NTSC_SETUP_VALUE	S	Yes	R/W	Set/get the NTSC setup compensation value. The unit is IRE (percentage of white level). Default value is 7.5%. (StillColor NTSC only)
IMG_ATTR_COLOR_SATURATION	S	Yes	R/W	Set/get the color saturation of the image. Saturation of 0 corresponds to a monochrome image. Default value is 1.
IMG_ATTR_COLOR_SW_CHROMA_FILTER	S	Yes	R/W	Set/get the software filter to clean the chroma signal (StillColor NTSC only): Disabled (0) Enabled (1)
IMG_ATTR_COLOR_TINT	S	Yes	R/W	Set/get 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	S	Yes	R	Returns the BUFLIST_ID of the buffer list associated with this session
IMG_ATTR_FRAME_COUNT	S	Yes	R	Returns the number of frames acquired since the start of an acquisition
IMG_ATTR_FRAME_FIELD	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.
IMG_ATTR_FRAMEWAIT_MSEC	S	No	R/W	Get/Set 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	S	Yes	R	Returns the number of reserved driver buffers currently left

Table A-1. Attribute Summary (Continued)

Attribute	Type	Imd.	R/W	Description
IMG_ATTR_GAIN	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_GENLOCK_SWITCH_CHAN	S	Yes	R/W	Determines if the board will relock to the video source when switching channels. If FALSE, the board will always relock to the video source when switching channels. If TRUE, the board will not relock to the video source when switching channels. Use this attribute if you have two genlocked cameras.
IMG_ATTR_HASRAM	I	Yes	R	Returns TRUE if the interface board has onboard memory
IMG_ATTR_HORZ_RESOLUTION	I	Yes	R	Returns the maximum horizontal resolution of the interface
IMG_ATTR_HSCALE	S	No	R/W	Set/get 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	S	No	R/W	Set/get the invert image mode. Beneficial when calling `imgPlot`. 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	S	Yes	R	Returns a buffer element number of the last received frame buffer
IMG_ATTR_LAST_VALID_FRAME	S	Yes	R	Returns the cumulative buffer index (frame#)
IMG_ATTR_LINE_COUNT	S	Yes	R	Returns the current line count of the frame being acquired

Table A-1. Attribute Summary (Continued)

Attribute	Type	Imd.	R/W	Description
IMG_ATTR_LUT	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	S	Yes	R	Returns TRUE if the session's buffer list is locked in memory
IMG_ATTR_NUM_BUFFERS	S	Yes	R	Returns the number of buffers in the buffer list associated with the session
IMG_ATTR_PCLK_DETECT	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	I	Yes	R	Returns the maximum pixel depth of the interface board in bytes
IMG_ATTR_RAMSIZE	I	Yes	R	Returns the size of the RAM on the interface board
IMG_ATTR_ROI_HEIGHT	S	No	R/W	Get/set the region of interest height of the camera/channel associated with this session
IMG_ATTR_ROI_LEFT	S	Yes	R/W	Get/set the region of interest left of the camera/channel associated with this session
IMG_ATTR_ROI_TOP	S	Yes	R/W	Get/set the region of interest top of the camera/channel associated with this session
IMG_ATTR_ROI_WIDTH	S	No	R/W	Get/set the region of interest width of the camera/channel associated with this session
IMG_ATTR_ROWBYTES	S	No	R/W	Set the true width of a horizontal line in memory. Used to calculate the next lines memory offset

Table A-1. Attribute Summary (Continued)

Attribute	Type	Imd.	R/W	Description
IMG_ATTR_START_FIELD	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	S	No	R/W	Get/set 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	I	Yes	R	Returns the maximum vertical resolution of the interface
IMG_ATTR_VSCALE	S	No	R	Set/get 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	S	Yes	R/W	Set the white reference value of the channel associated with this session. Values are 0–63.
IMG_ATTR_WHITE_REF_VOLT	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.
IMG_ATTR_XOFF_BUFFER	S	No	R/W	Set/get 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	S	No	R/W	Set/get 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	`imgGetAttribute` `imgSetAttribute`	Acquire all fields
IMG_ACQUIRE_EVEN	`imgGetAttribute` `imgSetAttribute`	Acquire only even fields
IMG_ACQUIRE_ODD	`imgGetAttribute` `imgSetAttribute`	Acquire only odd fields

IMG_AQ_DONE	`imgSessionWaitSignal`
	`imgSessionWaitSignalAsync`
	`imgPulseCreate`
		Asserted at the end of an acquisition when the last piece of data has been transferred to memory

IMG_AQ_IN_PROGRESS	`imgSessionWaitSignal`
	`imgSessionWaitSignalAsync`
	`imgPulseCreate`
		Asserted when the board initiates an acquisition either through a software- or hardware-triggered start

IMG_BOARD_INTERFACE	`imgGetAttribute`	Specifies the type of interface is a plug-in board

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

Table A-2. Constants Summary (Continued)

Constant	Use With	Description
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 private 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.

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_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

Table A-2. Constants Summary (Continued)

Constant	Use With	Description
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 Light color output
IMG_COLOR_REP_LUM8	imgGetAttribute imgSetAttribute	Specifies 8-bit Light 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)
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

Table A-2. Constants Summary (Continued)

Constant	Use With	Description
IMG_COLOR_REP_SAT8	`imgGetAttribute` `imgSetAttribute`	Specifies 8-bit Saturation color output
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_RTSI0	`imgPulseCreate`
	`imgSessionWaitSignal`
	`imgSessionWaitSignalAsync`
	`imgSessionTriggerConfigure`
	`imgSessionTriggerDrive`
	`imgSessionTriggerRead`
		Specifies the RTSI line 0

IMG_EXT_RTSI1	`imgSessionTriggerConfigure`
	`imgSessionTriggerDrive`
	`imgSessionTriggerRead`
		Specifies the RTSI line 1

IMG_EXT_RTSI2	`imgSessionTriggerConfigure`
	`imgSessionTriggerDrive`
	`imgSessionTriggerRead`
		Specifies the RTSI line 2

IMG_EXT_RTSI3	`imgSessionTriggerConfigure`
	`imgSessionTriggerDrive`
	`imgSessionTriggerRead`
		Specifies the RTSI line 3

IMG_EXT_RTSI4	`imgSessionTriggerConfigure`
	`imgSessionTriggerDrive`
	`imgSessionTriggerRead`
		Specifies the RTSI line 4

IMG_EXT_RTSI5	`imgSessionTriggerConfigure`
	`imgSessionTriggerDrive`
	`imgSessionTriggerRead`
		Specifies the RTSI line 5

IMG_EXT_RTSI6	`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

Table A-2. Constants Summary (Continued)

Constant	Use With	Description

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

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

Table A-2. Constants Summary (Continued)

Constant	Use With	Description
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_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

Table A-2. Constants Summary (Continued)

Constant	Use With	Description
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_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_START	`imgSessionTriggerDrive`	Specifies that the trigger line is driven with frame start
IMG_TRIG_DRIVE_FRAME_DONE	`imgSessionTriggerDrive`	Specifies that the trigger line is driven with frame done
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

B

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

Status Name	Description
IMG_ERR_AIOP	Cannot perform request, acquisition in progress
IMG_ERR_ALLOC	Error during large buffer allocation
IMG_ERR_BADCAMPARAM	Bad parameter from camera file
IMG_ERR_BADCAMTYPE	Bad camera type (not NTSC or PAL)
IMG_ERR_BADFILEXT	Unknown file extension
IMG_ERR_BADFILFMT	Bad file format
IMG_ERR_BADPIXTYPE	Camera not supported (not 8 bits)
IMG_ERR_BBLB	A buffer list, buffer is null
IMG_ERR_BBLE	Buffer list does contains an invalid command
IMG_ERR_BBLF	Buffer list does not contain a valid final command
IMG_ERR_BBUF	Bad buffer pointer in list
IMG_ERR_BCMF	Bad camera file (check syntax)
IMG_ERR_BDMA	Bad DMA transfer
IMG_ERR_BFRQ	Bad frequency values
IMG_ERR_BINT	Bad interface
IMG_ERR_BITP	Bad interface type
IMG_ERR_BPID	Bad pulse ID
IMG_ERR_BPMD	Bad pulse mode

Table B-1. Status Code Summary (Continued)

Status Name	Description
IMG_ERR_BROI	ROI width is less than rowbytes
IMG_ERR_BROW	Rowbytes is less than region of interest
IMG_ERR_BSIZ	Buffer size used is too small for attributes
IMG_ERR_BTAC	No trigger action—acquisition will time out
IMG_ERR_BTRG	Trigger loopback problem—cannot drive trigger with action enabled
IMG_ERR_DISE	Error releasing the image buffer
IMG_ERR_DLLE	DLL internal error, bad logic state
IMG_ERR_ELCK	Cannot lock buffers down, no more memory
IMG_ERR_EMEM	Not enough memory to perform the operation
IMG_ERR_FIFO	FIFO overflow caused acquisition to halt
IMG_ERR_GOOD	No error occurred
IMG_ERR_HLPR	Bad parameter to low-level—check attributes and high-level arguments
IMG_ERR_HWNC	Hardware not capable of supporting this
IMG_ERR_ILCK	Interface locked
IMG_ERR_MLCK	Memory lock error, cannot perform acquisition
IMG_ERR_MXBF	Too many buffers already allocated
IMG_ERR_MXBI	Exhausted buffer IDs
IMG_ERR_MXPI	Exhausted pulse IDs
IMG_ERR_NAIP	No acquisition in progress
IMG_ERR_NBUF	No buffers available, too early in acquisition
IMG_ERR_NCAM	No camera defined for this channel
IMG_ERR_NCAP	Function not implemented
IMG_ERR_NCFG	Invalid action, no buffers configured for session
IMG_ERR_NDLL	Unable to load DLL (LabWindows/CVI only)

Table B-1. Status Code Summary (Continued)

Status Name	Description
IMG_ERR_NEPK	No external pixel clock present
IMG_ERR_NFNC	Unable to find API function in DLL (LabWindows/CVI only)
IMG_ERR_NINF	No interface found
IMG_ERR_NLCK	Buffer list is not locked
IMG_ERR_NOSR	Unable to allocate system resources (LabWindows/CVI only)
IMG_ERR_NSAT	Nonsettable attribute
IMG_ERR_NVBL	Not successful because of hardware limitations
IMG_ERR_OSER	Operating system error occurred
IMG_ERR_OVRN	Too many interfaces open
IMG_ERR_PALKEYDTCT	PAL key detection error
IMG_ERR_PAR1	Function-specific, see function description
IMG_ERR_PAR10	Function-specific, see function description
IMG_ERR_PAR2	Function-specific, see function description
IMG_ERR_PAR3	Function-specific, see function description
IMG_ERR_PAR4	Function-specific, see function description
IMG_ERR_PAR5	Function-specific, see function description
IMG_ERR_PAR6	Function-specific, see function description
IMG_ERR_PAR7	Function-specific, see function description
IMG_ERR_PAR8	Function-specific, see function description
IMG_ERR_PAR9	Function-specific, see function description
IMG_ERR_PG_BAD_TRANS	Bad pattern generation transition time
IMG_ERR_PG_TOO_MANY	Too many pattern generation transitions defined
IMG_ERR_PLCK	Partial lock—cannot perform acquisition
IMG_ERR_PLNS	Pulse not started error
IMG_ERR_SCC1	RGB Color channel not set to 1
IMG_ERR_SCLM	Field scaling mode not supported

Table B-1. Status Code Summary (Continued)

Status Name	Description
IMG_ERR_SERIAL	Serial port error
IMG_ERR_SERIAL_TIMO	Serial transmit/receive timeout
IMG_ERR_SMALLALLOC	Error during small buffer allocation
IMG_ERR_TIMO	Wait timed out, acquisition not complete
IMG_ERR_VLCK	Cannot get lock on video source
IMG_ERR_ZBUF	Zero buffer size, no bytes filled
IMG_WRN_BCAM	Warning, corrupt camera file detected
IMG_WRN_BLKG	Unstable blanking reference (StillColor mode)
IMG_WRN_BRST	Bad quality colorburst (StillColor mode)
IMG_WRN_CONF	Warning, change requires reconfiguration to take effect
IMG_WRN_ILCK	Warning, interface still locked

C

Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a fax-on-demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call 512 795 6990. You can access these services at:

United States: 512 794 5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at 512 418 1111.

E-Mail Support (Currently USA Only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

`support@natinst.com`

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

Country	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Brazil	011 288 3336	011 288 8528
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Québec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 6120092	03 6120095
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
United Kingdom	01635 523545	01635 523154
United States	512 795 8248	512 794 5678

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___) _____Phone (___) _____

Computer brand_____ Model _____Processor_____

Operating system (include version number) _____

Clock speed _____MHz RAM _____MB Display adapter _____

Mouse ___yes ___no Other adapters installed_____

Hard disk capacity _____MB Brand_____

Instruments used _____

_____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is: _____

_____

_____

_____

_____

List any error messages: _____

_____

_____

The following steps reproduce the problem: _____

_____

_____

_____

_____

IMAQ Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

IMAQ hardware _____

Interrupt level of hardware _____

DMA channels of hardware _____

Base I/O address of hardware _____

Programming choice _____

NI-IMAQ, IMAQ Vision, LabVIEW, LabWindows/CVI, or ComponentWorks version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Other Products

Computer make and model _____

Microprocessor _____

Clock frequency or speed _____

Type of video board installed _____

Operating system version _____

Operating system mode _____

Programming language _____

Programming language version _____

Other boards in system _____

Base I/O address of other boards _____

DMA channels of other boards _____

Interrupt level of other boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: *NI-IMAQ Function Reference Manual*

Edition Date: June 1998

Part Number: 371447A-01

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

E-Mail Address _____

Phone (___) _____ Fax (___) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, Texas 78730-5039

Fax to: Technical Publications
National Instruments Corporation
512 794 5678

Glossary

Prefix	Meanings	Value
p-	pico	10^{-12}
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}
G-	giga-	10^{9}

Numbers/Symbols

+5V	5 V signal
–	negative of, or minus
Ω	ohm
/	per
%	percent
±	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	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. *See also* 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. *See also* Interpreter.
conversion device	device that transforms a signal from one form to another; for example, analog-to-digital converters (ADCs) for analog input and digital-to-analog converters (DACs) for analog output
CPU	central processing unit

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 = 20log10 V1/V2, 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 *use the current default setting*
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 = COS (x)
status = `AO_config`(**board**, **channel**, **range**)

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
GND	ground signal
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. *See also* 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. *See also* compiler.
interrupt	a computer signal indicating that the CPU should suspend its current task to service a designated activity
interrupt level	the relative priority at which a device can interrupt
I/O	input/output—the transfer of data to/from a computer system involving communications channels, operator interface devices, 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

K

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 of 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 IMAQ Configuration Utility 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	*See* 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. *See also* 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. *See also* NTSC.
PCI	Peripheral Component Interconnect—a high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA; it is achieving widespread acceptance as a standard for PCs and workstations and offers a theoretical maximum transfer rate of 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

R

RAM	random-access memory
real time	a property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time
relative accuracy	a measure in LSB of the accuracy of an ADC; it includes all nonlinearity and quantization errors but does not include offset and gain errors of the circuitry feeding the ADC
resolution	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
VSYNCIN	vertical sync in signal

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

G

H

I

L

M

N

P

R

S

T

U

V