

IMAQ™

NI-IMAQ™ User Manual

Image Acquisition Software

Worldwide Technical Support and Product Information

www.natinst.com

National Instruments Corporate Headquarters

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

Worldwide Offices

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

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

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

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

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

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

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

BridgeVIEW™, ComponentWorks™, CVI™, IMAQ™, LabVIEW™, NI-IMAQ™, RTSI™, and StillColor™ are trademarks of National Instruments Corporation.

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

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing for a level of reliability suitable for use in or in connection with surgical implants or as critical components in any life support systems whose failure to perform can reasonably be expected to cause significant injury to a human. Applications of National Instruments products involving medical or clinical treatment can create a potential for death or bodily injury caused by product failure, or by errors on the part of the user or application designer. Because each end-user system is customized and differs from National Instruments testing platforms and because a user or application designer may use National Instruments products in combination with other products in a manner not evaluated or contemplated by National Instruments, the user or application designer is ultimately responsible for verifying and validating the suitability of National Instruments products whenever National Instruments products are incorporated in a system or application, including, without limitation, the appropriate design, process and safety level of such system or application.

Contents

About This Manual

How to Use the NI-IMAQ Manual Set	ix
Conventions Used in This Manual	ix
National Instruments Documentation	x
Related Documentation	x

Chapter 1

Introduction to NI-IMAQ

About the NI-IMAQ Software	1-1
Application Development Environments	1-2
Fundamentals of Building Applications with NI-IMAQ	1-2
The NI-IMAQ Libraries	1-2
Creating an Application	1-3
Sample Programs	1-4

Chapter 2

Software Overview

Introduction	2-1
Generic Functions	2-2
High-Level Functions	2-2
Snap Functions	2-2
Grab Functions	2-2
Ring and Sequence Functions	2-3
Signal I/O Functions	2-3
Miscellaneous Functions	2-4
Low-Level Functions	2-5
Acquisition Functions	2-5
Attribute Functions	2-6
Buffer Management Functions	2-7
Interface Functions	2-7
Utility Functions	2-8

Chapter 3

Programming with NI-IMAQ

Introduction	3-1
High-Level Functions	3-1
Low-Level Functions	3-2
Establishing Interface Connections and Sessions.....	3-2
Interface Functions.....	3-2
Session Functions.....	3-3
Managing Buffers	3-4
Camera Attributes.....	3-4
NI-IMAQ Status Signals	3-5
Line Scan Image Acquisition	3-7
Introductory Programming Examples	3-7
High-Level Snap Functions	3-7
High-Level Grab Functions	3-8
High-Level Sequence Functions	3-10
High-Level Ring Functions.....	3-11
High-Level Signal I/O Functions.....	3-14
Advanced Programming Examples	3-15
Performing a Snap Using Low-Level Functions.....	3-15
Performing a Grab Using Low-Level Functions.....	3-16
Performing a Sequence Acquisition Using Low-Level Functions.....	3-16
Performing a Ring Acquisition Using Low-Level Functions	3-17
StillColor Snap Programming.....	3-17

Appendix A

Color Basics and StillColor

Appendix B

Technical Support Resources

Glossary

Index

Figures

Figure 3-1.	NI-IMAQ Status Signals	3-6
Figure 3-2.	Snap Programming Flowchart	3-8
Figure 3-3.	Grab Programming Flowchart	3-9
Figure 3-4.	Sequence Programming Flowchart	3-11
Figure 3-5.	Ring Programming Flowchart	3-13
Figure 3-6.	Signal I/O Function Programming Flowchart	3-15
Figure 3-7.	Composite StillColor Snap Programming Flowchart.....	3-18
Figure A-1.	White Light and the Visible Spectrum	A-1
Figure A-2.	Traditional Decoding.....	A-7
Figure A-3.	StillColor Decoding	A-7

Tables

Table 1-1.	Import Libraries.....	1-3
Table 3-1.	Interface Naming Convention	3-2

About This Manual

NI-IMAQ software is a powerful application programming interface (API) between your image acquisition application and the National Instruments image acquisition (IMAQ) devices. This manual explains how to use your NI-IMAQ software.

How to Use the NI-IMAQ Manual Set

To install your software and documentation set, you should begin by reading the setup and test document included with your hardware and the NI-IMAQ release notes. 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 this manual, you can use the *NI-IMAQ Function Reference Manual*, which contains detailed descriptions of the NI-IMAQ functions.

Conventions Used in This Manual



The following conventions are used in this manual:

bold

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

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

italic

Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.

monospace

Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.

monospace italic

Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

National Instruments Documentation

The *NI-IMAQ User Manual* is one piece of the documentation set for your system. You could have any of several types of documents, depending on the hardware and software in your system. Use the documents 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 ComponentWorks 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 you may find useful as you read this manual:

- *Microsoft Visual C++ User Guide to Programming*

Introduction to NI-IMAQ

This chapter describes the NI-IMAQ software and lists the application development environments compatible with NI-IMAQ, describes the fundamentals of creating NI-IMAQ applications for Windows NT and Windows 98/95, describes the files used to build these applications, and tells you where to find sample programs.

About the NI-IMAQ Software

Thank you for buying a National Instruments image acquisition (IMAQ) device, which includes NI-IMAQ software. NI-IMAQ is a set of functions that controls the National Instruments plug-in IMAQ devices for image acquisition and Real-Time System Integration (RTSI) bus multiboard synchronization.

NI-IMAQ has both high-level I/O functions for maximum ease of use and low-level I/O functions for maximum flexibility and performance. Examples of high-level functions are snap and grab image acquisition. Examples of low-level functions are buffer setup and video configuration. NI-IMAQ enhances the performance of National Instruments IMAQ devices because it lets multiple devices operate at their peak performance.

NI-IMAQ includes a buffer and data manager that uses sophisticated techniques for handling and managing image acquisition buffers so that you can simultaneously acquire and process data. NI-IMAQ uses direct memory access (DMA) to transfer all data.

NI-IMAQ is a library of routines that work with National Instruments IMAQ devices. NI-IMAQ contains methods for performing tasks ranging from simple device initialization to advanced high-speed real-time image acquisition. The number of services you need for your applications depends on the types of IMAQ devices you have and the complexity of your applications.

Application Development Environments

This release of NI-IMAQ supports the following Application Development Environments (ADEs) for Windows NT and Windows 98/95:

- LabVIEW version 4.x and higher
- LabWindows/CVI version 4.x and higher
- BridgeVIEW version 1.x and higher
- Borland C/C++ version 4.0 and higher
- Borland C++ Builder 3.0 and higher
- Metrowerks CodeWarrior 4.0 and higher
- Microsoft Visual C/C++ version 2.0 and higher
- Microsoft Visual Basic version 4.0 and higher



Note Although NI-IMAQ has been tested and found to work with these ADEs, other ADEs or higher versions of the ADEs listed above may also work.

If you are using Visual Basic, NI-IMAQ support is provided by the ComponentWorks IMAQ hardware interface control. Please consult the ComponentWorks IMAQ Vision documentation for more information.

Files on the NI-IMAQ software media may be compressed. Always run the NI-IMAQ installation utility to extract the files you want. For a brief description of the directories produced by the install programs and the names and purposes of the uncompressed files, consult the `readme.txt` file on your installation CD or diskettes.

Fundamentals of Building Applications with NI-IMAQ

The NI-IMAQ Libraries

The NI-IMAQ for Windows NT/98/95 function libraries are dynamic link libraries (DLLs), which means that NI-IMAQ routines are not linked into the executable files of applications. Only the information about the NI-IMAQ routines in the NI-IMAQ import libraries is stored in the executable files.

Import libraries contain information about their DLL-exported functions. They indicate the presence and location of the DLL routines. Depending on the development tools you are using, you may give the DLL routines information through import libraries or through function declarations. Your NI-IMAQ software kit contains function prototypes for all routines.

Creating an Application

This section outlines the process for developing NI-IMAQ applications using C for Windows NT and Windows 98/95. Detailed instructions on creating project and source files are not included. For information on creating and managing project files, consult the documentation included with your particular development environment.

When programming, use the following guidelines:

- All C source files that use NI-IMAQ functions must include the `NIIMAQ.H` header file. Add this file to the top of your source files.
- You must add the `IMAQ.LIB` import library to your project. Some environments allow you to add import libraries simply by inserting them into your list of project files. Other environments allow you to specify import libraries under the linker settings portion of the project file.
- When compiling, you will need to indicate where the compiler can find the NI-IMAQ header files and shared libraries. Most of the files you need for development are located under the NI-IMAQ target installation directory. If you choose the default directory during installation, the target installation directory is `C:\Program Files\National Instruments\NI-IMAQ`. The include files are located under the `include` subdirectory. The import libraries are located under the `lib\<environment>` subdirectory for the following platforms:

Table 1-1. Import Libraries

Development Environment	Directory
Microsoft Visual C++	<code>lib\msvc</code>
Borland C++	<code>lib\Borland</code>
Metrowerks CodeWarrior	<code>lib\CodeWarrior</code>

Sample Programs

Please refer to the `readme.txt` file located in your target installation directory for the latest details on NI-IMAQ sample programs. These programs are installed in the `sample` subdirectory under the target installation folder if you elected to install the sample files.

Software Overview

This chapter describes the classes of NI-IMAQ functions and briefly describes each function.

Introduction

NI-IMAQ functions are grouped according to the following classes:

- Generic functions
- High-level functions
 - Snap functions
 - Grab functions
 - Ring and sequence functions
 - Signal I/O functions
 - Miscellaneous functions
- Low-level functions
 - Acquisition functions
 - Attribute functions
 - Buffer management functions
 - Interface functions
 - Utility functions

The generic and high-level functions appear within each function class in the logical order you might need to use them. The low-level functions appear within each function class in alphabetical order.

Generic Functions

Use generic functions in both high-level and low-level applications.

<code>imgInterfaceOpen</code>	Opens by name an interface as specified in the Measurement & Automation Explorer for IMAQ.
<code>imgSessionOpen</code>	Opens a session of an unknown type and returns a session ID.
<code>imgClose</code>	Closes a session or interface and unlocks and releases all buffers associated with the data type.

High-Level Functions

Use high-level functions to quickly and easily capture images. If you need more advanced functionality, you can mix high-level functions with low-level functions.

Snap Functions

Snap functions program the session to capture all or a portion of a single frame or field to the user buffer.

<code>imgSnap</code>	Performs a single frame and field acquisition.
<code>imgSnapArea</code>	Performs an area-specific frame or field acquisition.

Grab Functions

Grab functions start a continuous image acquisition to a user buffer. Any frame or field can be copied from the grab buffer to another user buffer.

<code>imgGrabSetup</code>	Configures and optionally starts a continuous acquisition.
<code>imgGrab</code>	Performs a transfer from a continuous acquisition session. Call this function only after calling <code>imgGrabSetup</code> .

`imgGrabArea` Performs a transfer from a continuous acquisition. Call this function only after calling `imgGrabSetup`.

Ring and Sequence Functions

Ring and sequence functions start and stop a continuous acquisition of multiple fields or frames.

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

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

`imgSessionStartAcquisition` Starts a session acquisition identified by the session ID.

`imgSessionStopAcquisition` Stops a session acquisition identified by the session ID.

Signal I/O Functions

Signal I/O functions control the trigger lines on IMAQ devices.

`imgSessionTriggerConfigure` Configures an acquisition to start based on an external trigger.

`imgSessionLineTrigSource` Configures triggering per line for acquisition from a line scan camera.

`imgSessionTriggerClear` Disables all triggers on the session.

`imgSessionTriggerDrive` Configures the specified trigger line to drive a signal out.

`imgSessionTriggerRead` Reads the current value of the specified trigger line.

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

<code>imgSessionWaitSignalAsync</code>	Monitors for a signal to be asserted and invokes a user-defined callback when the signal is asserted.
<code>imgPulseCreate</code>	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.
<code>imgPulseDispose</code>	Disposes of a pulse ID.
<code>imgPulseRate</code>	Converts delay and width into delay, width, and timebase values needed by <code>imgPulseCreate</code> .
<code>imgPulseStart</code>	Starts the generation of a pulse. You must call <code>imgPulseCreate</code> first to configure the pulse.
<code>imgPulseStop</code>	Stops the generation of a pulse.

Miscellaneous Functions

Miscellaneous functions set and get the acquisition window's region of interest and return information such as session status and buffer sizes.

<code>imgSessionStatus</code>	Gets the current session status.
<code>imgSessionSetROI</code>	Sets acquisition region of interest.
<code>imgSessionGetROI</code>	Gets acquisition region of interest.
<code>imgSessionGetBufferSize</code>	Gets the minimum buffer size needed for frame buffer allocation.

Low-Level Functions

Use low-level functions when you require more direct hardware control.

Acquisition Functions

Use acquisition functions to configure, start, and abort an image acquisition, or examine a buffer during an acquisition.

<code>imgMemLock</code>	Locks all session-associated image buffers in memory in preparation for an acquisition.
<code>imgMemUnlock</code>	Unlocks all session-associated buffers.
<code>imgSessionAbort</code>	Stops an asynchronous acquisition or synchronous continuous acquisition immediately.
<code>imgSessionAcquire</code>	Starts acquisition synchronously or asynchronously to the frame buffers in the associated session buffer list.
<code>imgSessionConfigure</code>	Specifies the buffer list to use with this session.
<code>imgSessionCopyArea</code>	Copies an area of a session's buffer to a user-specified buffer.
<code>imgSessionCopyBuffer</code>	Copies a session's image data to a user buffer format.
<code>imgSessionExamineBuffer</code>	Extracts a buffer from a live acquisition; lets you lock a buffer out of a continuous loop sequence for processing when you are performing a ring (continuous) acquisition.
<code>imgSessionReleaseBuffer</code>	Releases a buffer that was previously held with <code>imgSessionExamineBuffer</code> .

Attribute Functions

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

`imgGetAttribute` Returns an attribute for an interface or session.

`imgGetCameraAttributeNumeric` Gets the value of numeric camera attributes.

`imgGetCameraAttributeString` Gets the value of camera attributes.

`imgSessionGetLostFramesList` Gets information about frames that were overwritten during a continuous acquisition.

`imgSessionSetUserLUT8bits` Downloads a custom 8-bit LUT to your IMAQ device.

`ImgSessionSetUserLUT16bits` Downloads a custom 16-bit LUT to your IMAQ device.

`imgSetAttribute` Sets an attribute for an interface or session.

`imgSetCameraAttributeNumeric` Sets the value of numeric camera attributes.

`imgSetCameraAttributeString` Sets the value of camera attributes.

Buffer Management Functions

Use buffer management functions to set up objects such as buffer lists and buffers.

<code>imgCreateBuffer</code>	Creates a user frame buffer based on the geometric definitions of the associated session.
<code>imgCreateBufList</code>	Creates a buffer list that is passed to <code>imgSessionConfigure</code> .
<code>imgDisposeBuffer</code>	Disposes of a user frame buffer.
<code>imgDisposeBufList</code>	Purges all image buffers associated with this buffer list.
<code>imgGetBufferElement</code>	Gets an element of a specific type from a buffer list.
<code>imgSessionClearBuffer</code>	Clears a session's image data to the specified pixel value.
<code>imgSetBufferElement</code>	Sets a buffer list element of a given type to a specific value.

Interface Functions

Interface functions load and control the selected IMAQ device and cameras. These functions use information stored by the Measurement & Automation Explorer for IMAQ.

<code>imgInterfaceQueryNames</code>	Returns the interface name identified by the index parameter.
<code>imgInterfaceReset</code>	Performs a hardware reset on the interface type and returns a status, either good or bad.

Utility Functions

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

<code>imgPlot</code>	Plots a buffer to a window given a native window handle.
<code>imgSessionSaveBufferEx</code>	Saves a buffer of a session to disk in a native operating system-specific format such as bitmap or tag image file format (TIFF).
<code>imgShowError</code>	Returns a null terminated string describing the error code.

Programming with NI-IMAQ

This chapter contains an overview of the NI-IMAQ library, a description of the programming flow of NI-IMAQ, and programming examples. Flowcharts are included for the following operations: snap, grab, sequence, ring, and StillColor acquisitions.

Introduction

The NI-IMAQ API is divided into two groups, the high-level functions and the low-level functions. With the high-level functions, you can write programs quickly without having to learn the details of the low-level API and driver. The low-level functions give you finer granularity and control over your image acquisition process, but you must understand the API and driver in greater detail.



Note The high-level functions call low-level functions and use certain attributes that are listed in the high-level function description in the *NI-IMAQ Function Reference Manual*. Changing the value of these attributes while using low-level functions will affect the operation of the high-level functions.

High-Level Functions

The high-level function set supports four basic types of image acquisition:

- *Snap* acquires a single frame or field to a buffer.
- *Grab* performs an acquisition that loops continually on one buffer; you obtain a copy of the acquisition buffer by *grabbing* a copy to a separate buffer that can be used for analysis.
- *Sequence* performs an acquisition that acquires a specified number of buffers, then stops.
- *Ring* performs an acquisition that loops continually on a specified number of buffers.

The high-level function set also allows simple triggered acquisitions and the generation of external signals on the trigger lines.

Low-Level Functions

The low-level function set supports all types of acquisition and can be used to:

- Create a custom acquisition sequence or ring
- Create and manage your own buffers
- Set session and interface attributes to adjust image quality and size
- Start a synchronous or asynchronous acquisition
- Extract buffers out of a live acquisition for analysis
- Set up and control triggered acquisitions

Establishing Interface Connections and Sessions

To acquire images using the high-level or low-level functions, you must first learn how to establish a connection to an interface and create a session. See the *Interface Functions* and *Session Functions* sections in this chapter for information on how to manage interfaces and sessions, then refer to the high-level or low-level samples for information on acquiring images.

Interface Functions

Use interface functions to query the number of available interfaces, establish a connection to, control access to, and initialize hardware such as the PCI/PXI-1408. All interfaces in NI-IMAQ are specified by a name. By default, the system creates default names for the number of boards in your system. These names observe the convention shown in Table 3-1.

Table 3-1. Interface Naming Convention

Interface Name	Board Installed
img0	Board 0
img1	Board 1
...	...
img n	Board n

You can edit existing or create new interfaces by using the Measurement & Automation Explorer for IMAQ. You also can use the Measurement & Automation Explorer for IMAQ to configure the board serial number and the default state of a particular interface.

Before you can acquire image data successfully, you must open an interface by using the `imgInterfaceOpen` function. `imgInterfaceOpen` requires an interface name and returns a handle to this interface. NI-IMAQ then uses this handle to reference this interface when using other NI-IMAQ functions.

To establish a connection to the first board in your system, use the following program example:

```
INTERFACE_ID  interfaceID;
if (imgInterfaceOpen("img0", &interfaceID) == IMG_ERR_GOOD)
{
    ... user code ...
    imgClose(interfaceID, FALSE);
}
```

This example opens an interface to `img0`. When the program is finished with the interface, it closes the interface using the `imgClose` function.

For a complete list of the available interface functions, refer to the *NI-IMAQ Function Reference Manual*.

Session Functions

Use session functions to configure the type of acquisition you want to perform on a particular interface. After you have established a connection to an interface, you need to create a session and configure it to perform the type of acquisition you require.

To create a session, call the `imgSessionOpen` function. This function requires a valid interface handle and returns a handle to a session. NI-IMAQ then uses this session handle to reference this session when using other NI-IMAQ calls.

To create a session, use the following example program:

```
INTERFACE_ID  interfaceID;
SESSION_ID    sessionID;
if (imgInterfaceOpen("img0", &interfaceID) == IMG_ERR_GOOD)
{
    if (imgSessionOpen(interfaceID, &sessionID) == IMG_ERR_GOOD)
    {
        ... user code ...
        imgClose(sessionID, FALSE);
    }
    imgClose(interfaceID, FALSE);
}
```

This example opens an interface to `img0` and then creates a session to acquire images. When the program is finished with the interface and session, it then closes both handles using the `imgClose` function.

For a complete list of the available session functions, refer to the *NI-IMAQ Function Reference Manual*.

Managing Buffers

Buffer management can be performed either by you or automatically by NI-IMAQ. If the high-level acquisition routines (`imgSnap`, `imgGrab`, `imgSequenceSetup`, and `imgRingSetup`) are initiated with `NULL` pointers for buffer addresses, NI-IMAQ will automatically allocate a buffer and return the value of the buffer pointer to you. After you have a buffer pointer, you can use this pointer in successive calls.

For greater control of the acquisition buffers, such as creating buffers larger than the image size for adding borders, you can create them by calling a memory allocation routine (for example, `malloc`) or using the low-level function `imgCreateBuffer`. When creating buffers using either approach, dispose of the buffers using `free` or `imgDisposeBuffer` when applicable to free PC memory for maximum performance.

Camera Attributes

The camera attributes allow you to control camera functions, such as integration time and pixel binning, directly from NI-IMAQ. These camera attributes are camera-specific and can also be set in the Measurement & Automation Explorer for IMAQ on the **Advanced** tab. Information about specific attributes for your camera is contained in `<my camera>.txt`, which can be found in the `ni-imaq/camera info` directory. For more information about camera attributes and their uses, please consult your camera documentation.



Note Currently, camera attributes are supported only by the IMAQ PCI-1424 and the IMAQ PCI/PXI-1422.

The camera attribute file lists all attributes for the camera. Each attribute description contains four fields—**Attribute Name**, **Description**, **Data Type**, and **Possible Values**. The **Attribute Name** field contains the name of the attribute in quotes. The **Data Type** field contains the data type of the attribute—**String**, **Integer**, or **Float**. **String** indicates that there are

several valid values for this attribute that are expressed as strings. The list of valid values is indicated in **Possible Values**. **Integer** indicates that the attribute value is a numeric value of type integer. **Float** indicates that the attribute value is a numeric value of type floating point. The valid integer and float values are indicated in **Possible Values**.

Use the `imgSetCameraAttributeString` and `imgGetCameraAttributeString` functions to set and get the value of **String**, **Float** and **Integer** attributes. Use the `imgSetCameraAttributeNumeric` and `imgGetCameraAttributeNumeric` functions to set and get the value of **Float** and **Integer** attributes.



Note The spelling and syntax of the **Attribute Name** and string values must match the camera attribute file exactly.

NI-IMAQ Status Signals

NI-IMAQ has several status signals that can be used to trigger the generation of a pulse or invoke a callback function. **Acquisition in Progress** indicates that the board is acquiring image data. This signal goes TRUE when the board initiates the acquisition either through a software or hardware triggered start. When the last piece of image data is transferred to memory, this signal goes FALSE. If the acquisition is a sequence, acquisition in progress will stay TRUE throughout the acquisition until the entire sequence is completed. **Acquisition Done** is the reverse of **Acquisition in Progress**. This signal goes TRUE when the last piece of data is transferred to memory indicating that the acquisition has completed.

Frame Start and **Frame Stop** indicate the status of an acquisition on a buffer basis. **Frame Start** indicates that a buffer is being acquired. This signal goes TRUE when the first valid pixel is detected by the board (even if this pixel is not in the current region of interest). The signal goes FALSE when the last valid pixel is detected by the board. If the acquisition is a sequence or a ring, **Frame Start** will pulse for every buffer in the acquisition. **Frame Done** is the reverse of **Frame Start** and indicates when the image is transferred from the camera to the IMAQ board.

Buffer Complete indicates when the image data has been transferred to memory and is available for image processing. **Buffer Complete** will go TRUE when the data in an image buffer has been transferred to memory (either onboard or system memory, depending on the acquisition).

Figure 3-1 illustrates the values of the signals during a three-buffer sequence acquisition.

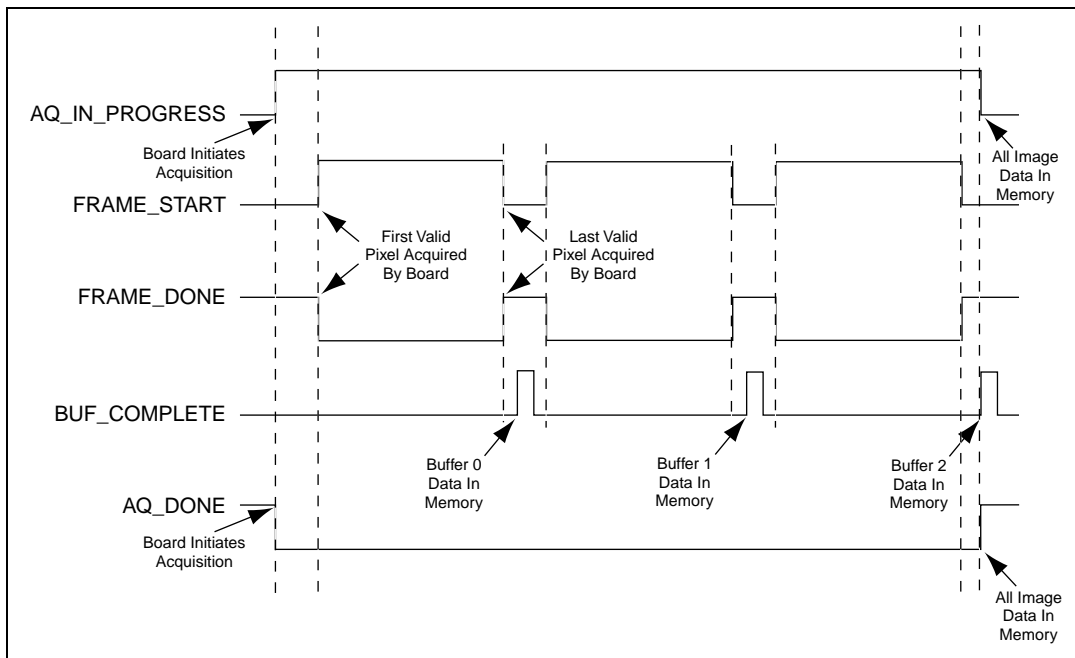


Figure 3-1. NI-IMAQ Status Signals

You can use the NI-IMAQ status signals for many purposes. Pulses can be generated based on the assertion of any of these signals. This allows you to generate specific timing pulses based on acquisitions to control other aspects of your system, such as a strobe light. Furthermore, you can configure callback functions that are invoked based on any of these signals. For example, you may want to initiate an image processing routine as soon as an image is in memory. You can configure a callback containing image processing code to be invoked when **Buffer Complete** is asserted.

Line Scan Image Acquisition

Unlike area scan cameras, line scan cameras output only one line at a time. However, the programming interface for area scan and line scan cameras is identical. The driver builds up the lines acquired into a 2D image. The height of this image is set in the Measurement & Automation Explorer for IMAQ and can be changed with the region of interest height attribute.

Triggering line scan images is identical to triggering area scan images. Using the `imgSessionTriggerConfigure` function, you can trigger the start of each buffer. The `imgSessionLineTrigSource` function allows you to trigger each line of the image, not just the start of the image. For example, if you are using a conveyor belt, you can use an encoder to trigger each line of the image and synchronize the movement of the conveyor belt and the image acquisition.

Introductory Programming Examples

This section introduces some examples for performing the different types of image acquisition. The error codes that NI-IMAQ returns are not included in the examples. In your programs, always check the return code for errors.



Note You can find these code examples in the `ni-imaq\samples` directory.

High-Level Snap Functions

A *snap* acquires a single image into a memory buffer. Snap functions include `imgSnap` and `imgSnapArea`. Use these functions to acquire a single frame or field to a buffer. To use these functions, you must have a valid session handle.

When you invoke a snap, it initializes the board and acquires the next incoming video frame (or field) to a buffer. A snap is appropriate for low-speed or single-capture applications where ease of programming is essential. Figure 3-2 illustrates a typical snap programming order.

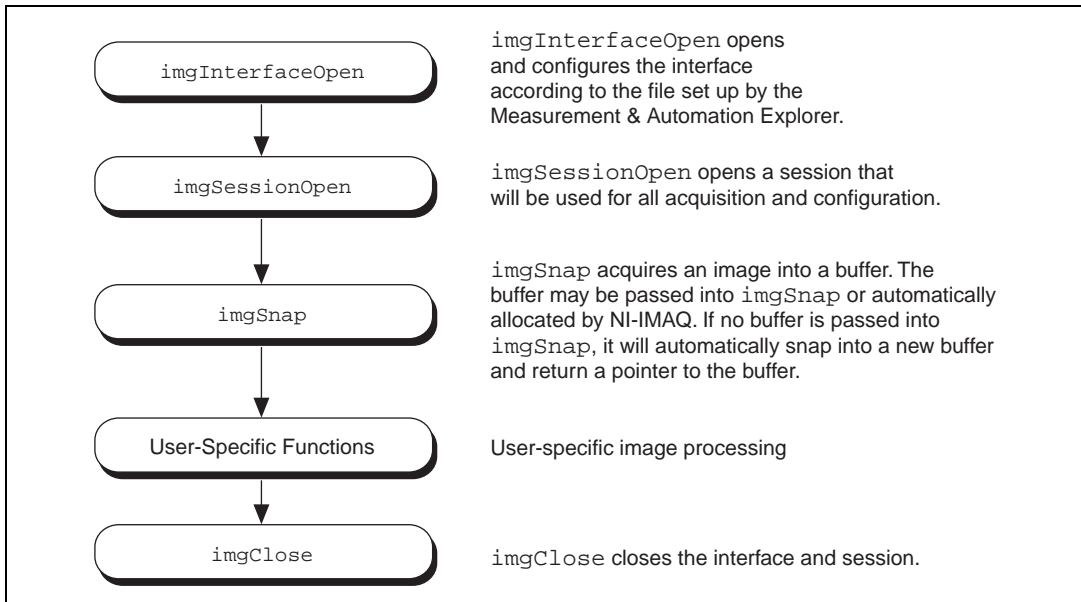


Figure 3-2. Snap Programming Flowchart

The `hlSnap.c` example demonstrates how to perform a single snap using `imgSnap`. The example opens an interface and a session and then performs a single snap. The buffer pointer that is passed to `imgSnap` is initialized to `NULL`, which instructs `imgSnap` to automatically allocate a buffer for the image. The size of the buffer is calculated based on the region of interest (ROI) and the `rowPixel` attributes: ROI height multiplied by `rowPixel` multiplied by the number of bytes per pixel. When you open a session, the ROI values are initialized from the acquisition window (ACQWINDOW) dimensions that are configured in the Measurement & Automation Explorer for IMAQ. The ACQWINDOW dimensions will vary depending on the type of camera you are using.

The sample then calls a process function to analyze the image. When the program is finished, it calls `imgClose` with the interface handle and sets the `freeResources` flag to `TRUE`. This instructs NI-IMAQ to free all of the resources associated with this interface, which releases the session as well as the memory buffer allocated by `imgSnap`.

High-Level Grab Functions

A *grab* is a continuous high-speed acquisition of data to a single buffer in host memory. Grab functions include `imgGrabSetup`, `imgGrab` and `imgGrabArea`. You can use these functions to perform an acquisition that

loops continually on one buffer. A copy of the acquisition buffer is obtained by grabbing a copy to a separate buffer. To use these functions, you must have a valid session handle.

Calling `imgGrabSetup` initializes a session for a grab acquisition. After `imgGrabSetup`, each successive grab will copy the last acquired buffer into a user buffer where you can perform processing on the image. A grab is appropriate for high-speed applications where you need processing performed on only one image at a time. Figure 3-3 illustrates a typical grab programming order.

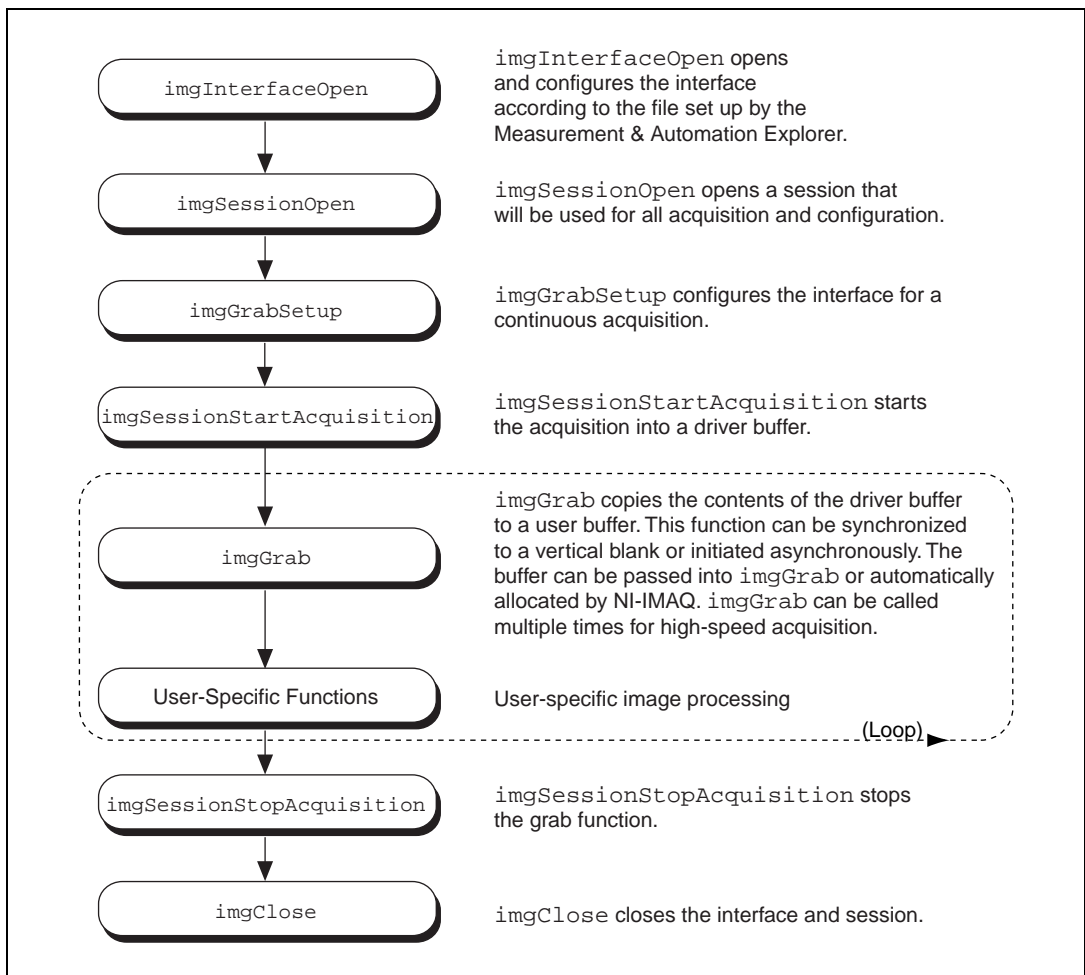


Figure 3-3. Grab Programming Flowchart

The `hlgrab.c` example demonstrates how to perform a grab using `imgGrabArea`. The example performs multiple grabs until an appropriate condition is met. The program configures the session to perform a grab operation by calling the `imgGrabSetup` function. The program then calculates the area to grab using the current ROI, `rowPixels`, and `BYTESPERPIXEL`, and the acquisition is started by calling `imgSessionStartAcquisition`. In this example, we allocate our own user buffer for grabbing and pass this buffer to `imgGrabArea`. When the acquisition is complete, it stops. The program then frees the user buffer and all of the resources associated with this interface by calling `imgClose`.

High-Level Sequence Functions

Sequence functions include `imgSequenceSetup`, `imgSessionStartAcquisition` and `imgStopAcquisition`. A *sequence* initiates a variable-length and variable-delay transfer to multiple buffers. You can configure the delay between acquisitions with `SequenceSetup` and specify both the buffer list that will be used for transfers and the number of buffers. After `imgSequenceSetup`, you can monitor the status of the transfer and perform processing on any of the buffers in the sequence or you can wait until the acquisition completes and process all buffers simultaneously.

A sequence is appropriate for applications where you need to perform processing on multiple images. You can configure a sequence to acquire every frame or skip a variable number of frames between each image. Figure 3-4 illustrates a typical sequence programming order.

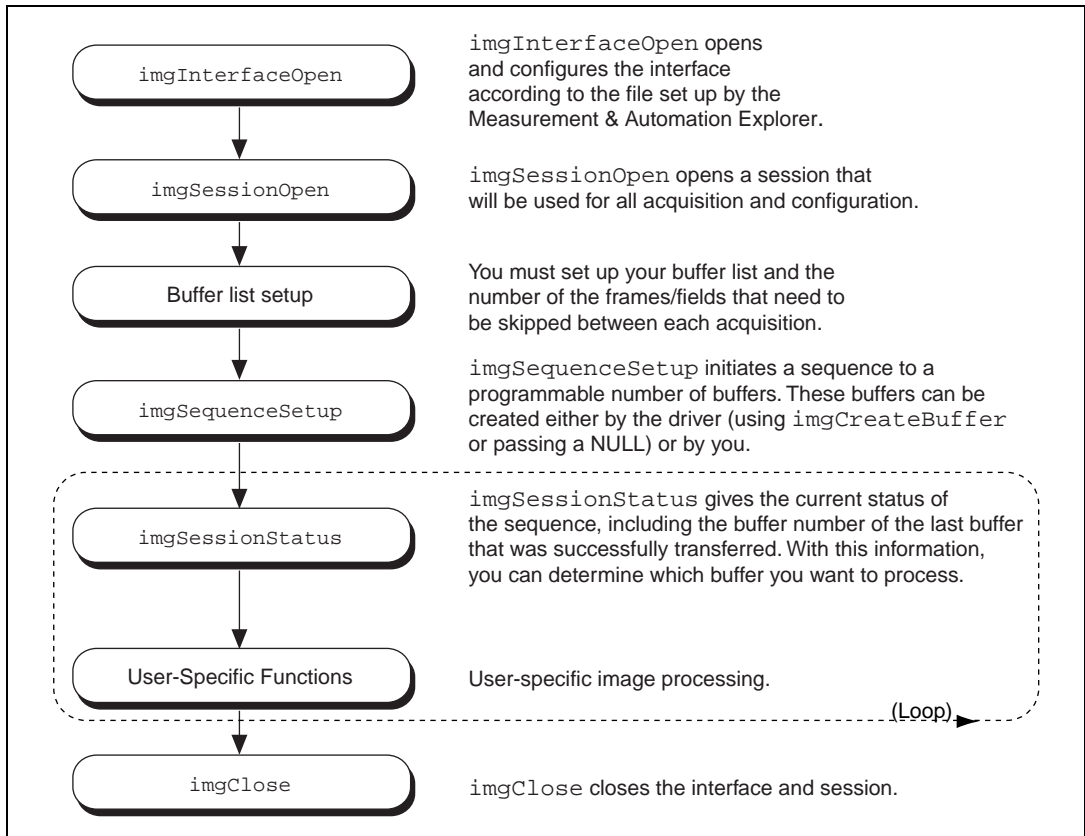


Figure 3-4. Sequence Programming Flowchart

The `HLSeq.c` example demonstrates how to perform a sequence acquisition using `imgSequenceSetup`. The example sets up a sequence that uses 10 user-allocated buffers. Each buffer in the sequence has its own skip count associated with it. The skip count is the number of frames to skip prior to acquiring the next image. The acquisition is started at setup time and the setup call is synchronous.

High-Level Ring Functions

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

A *ring* initiates a continuous high-speed acquisition to multiple buffers. Calling `imgRingSetup` initiates a ring. `imgRingSetup` specifies both the

buffer list that will be used for transfers and the number of buffers. After `imgRingSetup` is called, you can monitor the status of the transfer and perform processing on any of the buffers in the ring. A ring is appropriate for high-speed applications where you need to perform processing on every image. You must use multiple buffers because processing times may vary depending on other applications and processing results. You can configure a ring to acquire every frame or to skip a fixed number of frames between each acquisition.

For certain applications, you can temporarily extract a buffer from the ring to prevent it from being overwritten during the ring's next pass. Use the `imgSessionExamineBuffer` and `imgSessionReleaseBuffer` functions to do this. Figure 3-5 illustrates a typical ring programming order.

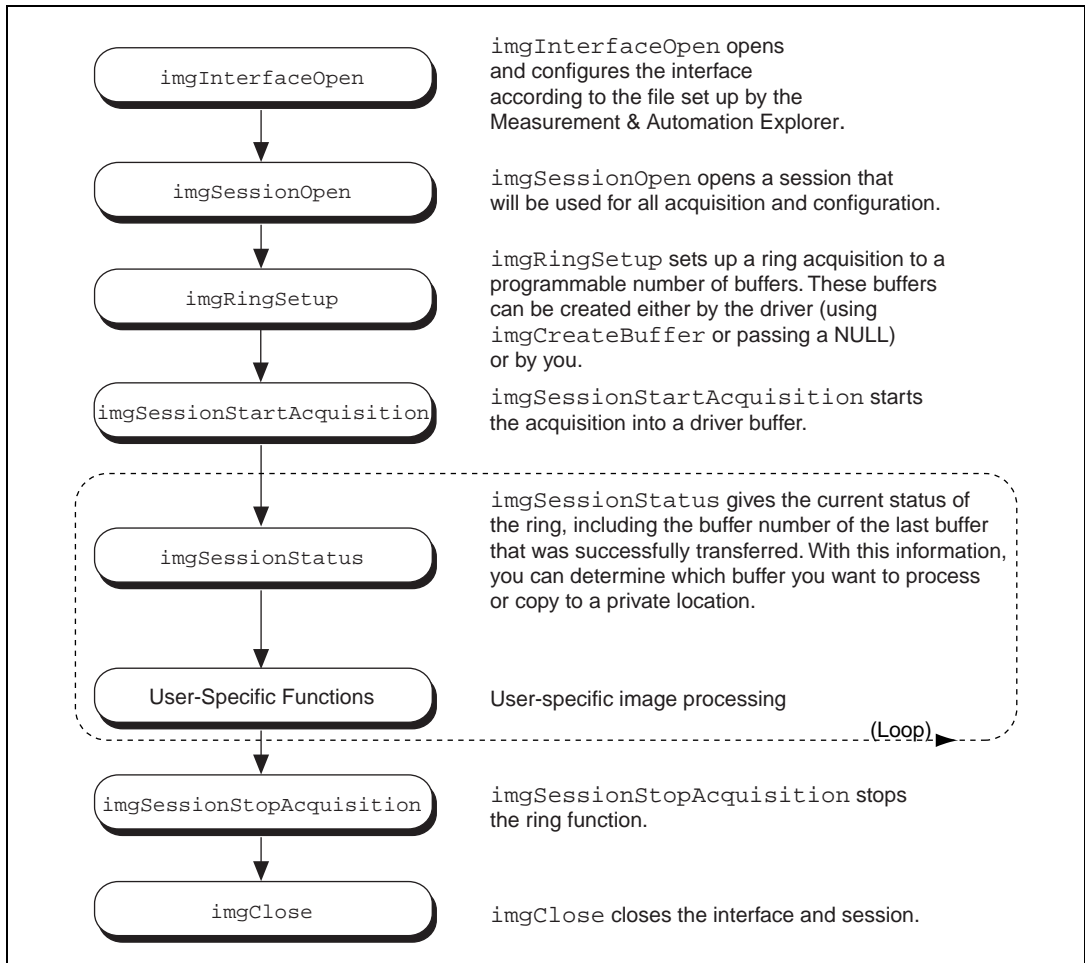


Figure 3-5. Ring Programming Flowchart

The `HLRing.c` example demonstrates how to perform a ring acquisition using `imgRingSetup`. The example sets up a ring containing six buffers and sets the skip count to three, which causes the program to acquire on every third frame. Unlike the sequence example, the skip count is set to the same value for every buffer in the ring. A *skip count* is the number of frames skipped prior to acquiring an image to a buffer. The program then loops, waiting for the next buffer to be acquired. The `imgSessionStatus` function queries NI-IMAQ for the buffer number of the last valid buffer that has been acquired. The last valid buffer is defined as the buffer that contains the most recent video image. This process will continue until a designated condition is met and then the acquisition stops.

High-Level Signal I/O Functions

The signal I/O functions fall into two categories, triggering acquisitions and driving the external trigger lines. Triggered acquisitions allow images to be acquired precisely when an external event occurs, such as a sensor activating. The driving of external trigger lines allows external devices to be controlled in sync with the image acquisition. For example, a strobe light could be fired when a sequence acquisition begins.

Any of the four types of acquisitions can be initiated from an external trigger source by using `imgSessionTriggerConfigure`. For sequence and ring, just the first buffer in the list can be triggered or each buffer in the list can be triggered. After using this function to set up the trigger, any acquisition performed on the session will wait for a trigger. Use `imgSessionTriggerClear` to remove the trigger settings from the session.

Some applications need to send signals out from the IMAQ hardware to an external device. Many types of signals can be driven out of the trigger lines by using `imgSessionTriggerDrive`. This function takes a trigger line number, the polarity the line should be driven, and what to drive on the line. This can be a steady state value of high or low or it can be one of the internal state signals of the hardware, such as acquisition in progress. When specific pulses need to be generated, `imgPulseCreate` and `imgPulseStart` can be used.

Figure 3-6 shows the outline of a program that waits for an external trigger on line 1 before acquiring a single image. It also configures the driver to assert RTSI trigger line 3 when the acquisition is finished. The `trigsnap.c` example contains C code that implements this program.

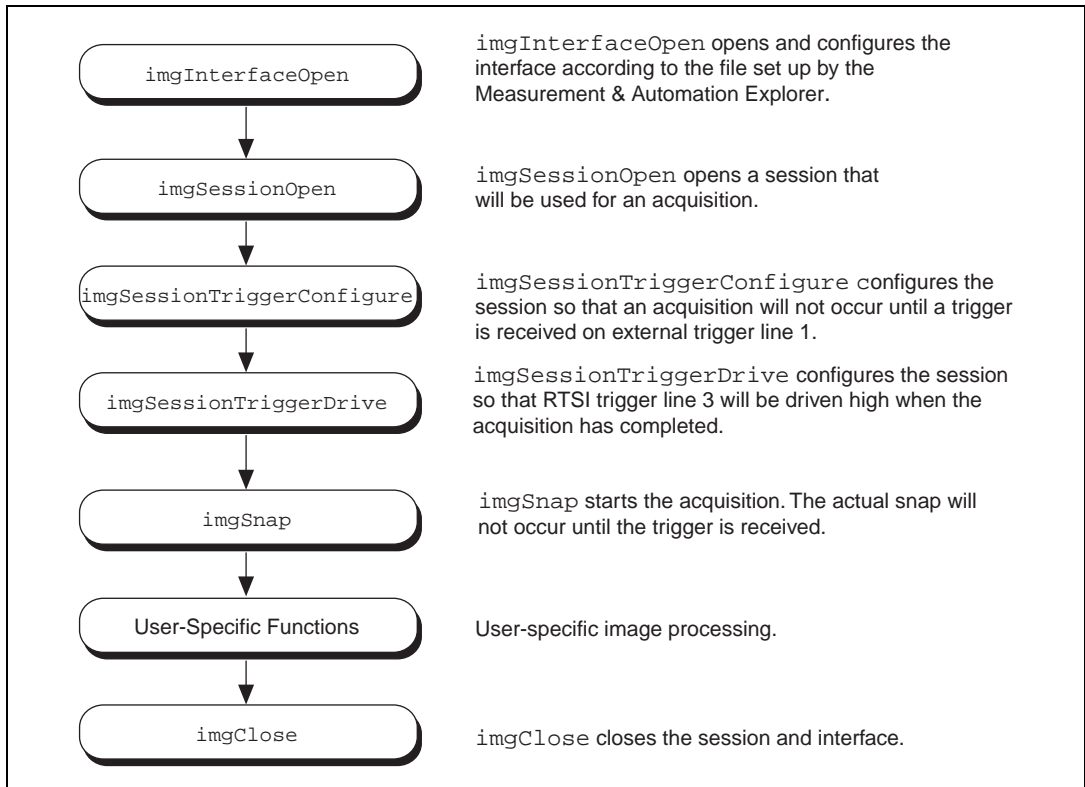


Figure 3-6. Signal I/O Function Programming Flowchart

Advanced Programming Examples

You can use low-level functions or combine high-and low-level functions for more advanced programming techniques, including snap, grab, sequence, ring, and color image acquisitions.

Performing a Snap Using Low-Level Functions

The `LLSnap.c` example demonstrates how to perform a snap acquisition using low-level calls. The example sets up a single-frame acquisition to a buffer allocated by NI-IMAQ. The program retrieves the acquisition window width of the selected camera and aligns it on a 32-bit boundary. You must align both the acquisition window width and `rowPixels` on a 32-bit boundary to ensure that your image is acquired properly. The software does not perform this alignment for you unless you select a scaling option. Although the Measurement & Automation Explorer for IMAQ

performs this alignment for you when you acquire an image with it, you must perform the alignment yourself if you use window widths not aligned on a 32-bit boundary.

After the program sets the ROI, it locks the memory and acquires the image. If you choose to plot the image using the `imgPlot` function, you must align the image width on a 32-bit boundary as well.

Performing a Grab Using Low-Level Functions

The `LLGrab.c` example demonstrates how to perform a grab acquisition using low-level calls. The example sets up a continuous acquisition to a single user-allocated buffer.

As described in the low-level snap example, the program retrieves the acquisition window width of the selected camera and aligns it on a 32-bit boundary. The program creates a buffer list to describe the acquisition buffers. Next, the program sets the ROI to the acquisition window width. The program performs a calculation to determine the correct memory requirements of the user buffer. The program creates the buffer and configures buffer element 0 for a single continuous acquisition. The program then locks the memory and starts the image acquisition asynchronously. The main processing loop of the code shows how to wait for vertical blank and copy the buffer to an analysis buffer.

Keep your analysis code fast to minimize the number of missed frames during analysis. If you need more time to examine a buffer, set up a multiple-buffer ring and call `imgSessionExamineBuffer` to extract the desired buffer from the live sequence.

Performing a Sequence Acquisition Using Low-Level Functions

The `LLSeq.c` example demonstrates how to perform a sequence acquisition using low-level calls. The example sets up a sequence acquisition to multiple buffers allocated by NI-IMAQ. As described in the low-level snap example, the program retrieves the acquisition window width of the selected camera and aligns it on a 32-bit boundary. It creates a buffer list to describe the acquisition buffers. Next, the program sets the ROI to the acquisition window width. The program calculates the correct memory requirements of the frame buffer. However, this is not necessary if you choose to use the default acquisition window width, `rowPixels`, and ROI. In this case, NI-IMAQ will allocate the correct size buffer if you pass a `NULL` as the size parameter to `imgCreateBuffer`. The program creates the buffer and configures the buffer list for each buffer element in the ring.

The program locks the memory and starts the image acquisition asynchronously.

The main processing loop of the code shows how to process each buffer acquired in sequential order.

Performing a Ring Acquisition Using Low-Level Functions

The `LLRing.c` example demonstrates how to perform a ring acquisition using low-level calls. The example sets up a continuous acquisition to multiple buffers allocated by NI-IMAQ.

As described in the low-level snap example, the program retrieves the acquisition window width of the selected camera and aligns it on a 32-bit boundary. It then creates a buffer list to describe the acquisition buffers. Next, the program sets the ROI to the acquisition window width. The program calculates the correct memory requirements of the frame buffer. However, this is not necessary if you choose to use the default acquisition window width, `rowPixels`, and ROI. In this case, NI-IMAQ will allocate the correct size buffer if you pass a `NULL` as the size parameter to `imgCreateBuffer`. The buffer is created and the buffer list is configured for each buffer element in the ring. The memory is locked and the image acquisition is started asynchronously.

The main processing loop of the code shows how to wait for the first buffer to be filled and subsequently processed. NI-IMAQ returns a value of `0xFFFFFFFF` as the `IMG_ATTR_LAST_VALID_BUFFER` attribute until the successful acquisition of the first buffer. To guarantee that you wait for the acquisition of a new buffer in a ring with more than one buffer, you can loop on the attribute `IMG_ATTR_LAST_VALID_BUFFER` until it changes. If your buffer analysis requires many computations, call `imgSessionExamineBuffer` to extract the desired buffer from the live sequence. When using `imgSessionExamineBuffer`, the buffer requested is literally pulled from the looping sequence for the duration of the analysis. Use `imgSessionReleaseBuffer` to return the buffer to the continuous sequence.

StillColor Snap Programming

You can use the high-level snap function to acquire StillColor images from either composite or RGB video sources. As shown in Figure 3-7, acquiring a StillColor image is identical to acquiring a monochrome image except for two session attribute settings. For more information on StillColor, see Appendix A, *Color Basics and StillColor*.

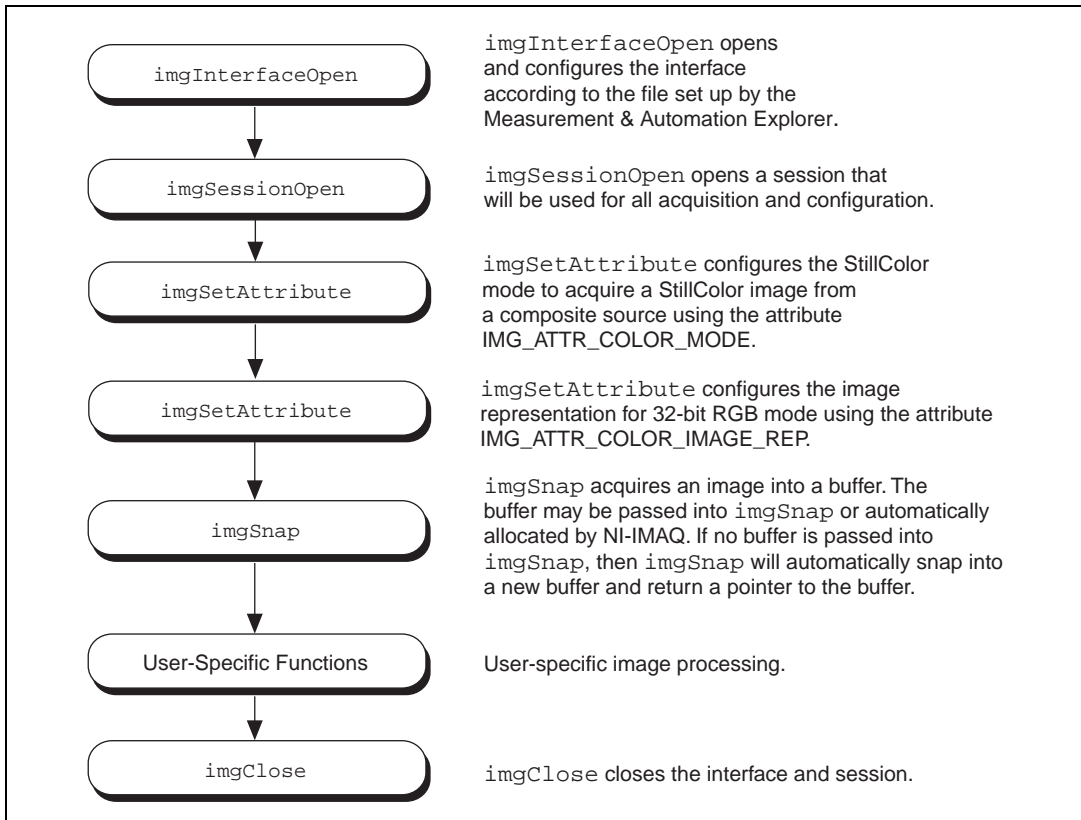


Figure 3-7. Composite StillColor Snap Programming Flowchart

The `SCSnap.c` example demonstrates how to perform a single, composite StillColor snap. The example first opens an interface and a session. The example then uses `imgSetAttribute` to enable and configure StillColor mode to acquire a composite image. The example also configures the image data representation to 32-bit RGB mode. `imgSnap` acquires a StillColor image and returns the image data in the buffer. After the example processes the image, it calls `imgClose` to close the handles and free all of the resources associated with the interface.

Color Basics and StillColor

This appendix explains basic color theories, describes the different color image output options, and describes the different methods you can use to acquire a color image using the IMAQ PCI/PXI-1408 and National Instruments StillColor technology.

Introduction to Color

Color is the wavelength of the light we receive in our eye when we look at an object. In theory, the color spectrum is infinite. Humans, however, can see only a small portion of this spectrum—the portion that goes from the red edge of infrared light (the longest wavelength) to the blue edge of ultraviolet light (the shortest wavelength). This continuous spectrum is called the visible spectrum, as shown in Figure A-1.

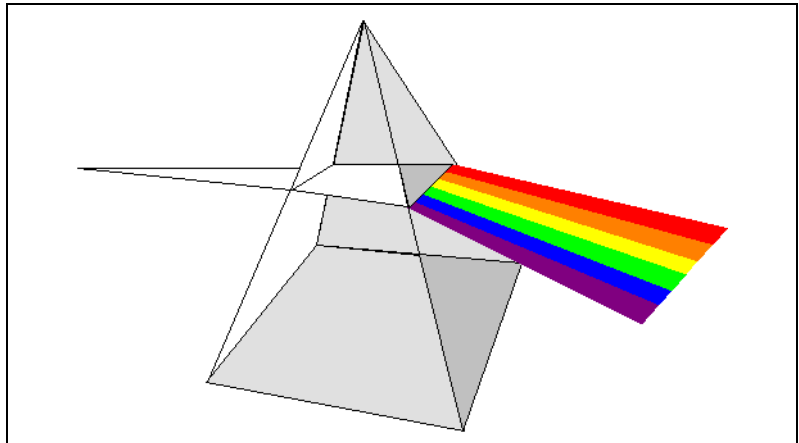


Figure A-1. White Light and the Visible Spectrum

White light is a combination of all colors at once. The spectrum of white light is continuous and goes from ultraviolet to infrared in a smooth transition. You can represent a good approximation of white light by selecting a few reference colors and weighting them appropriately. The most common way to represent white light is to use three reference components, such as red, green, and blue (R, G, and B primaries). You can

simulate most colors of the visible spectrum using these primaries. For example, video projectors use red, green, and blue light generators, and an RGB camera uses red, green, and blue sensors.

The perception of a color depends on many factors, such as:

- *Hue*, which is the perceived dominant color. Hue depends directly on the wavelength of a color.
- *Saturation*, which is dependent on the amount of white light present in a color. Pastels typically have a low saturation while very rich colors have a high saturation. For example, pink typically has a red hue but has a low saturation.
- *Luminance*, which is 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.
- *Intensity*, which is the brightness of a color and which is usually expressed as light or dark. For example, orange and brown may have the same hue and saturation; however, orange has a greater intensity than brown. Intensity is used only in StillColor images. For more information on StillColor, see the *StillColor Basics* section in this appendix.

Image Representations

Color images can be represented in several different formats. These formats can contain all color information from the image or they can consist of just one aspect of the color information, such as hue or luminance. The following image representations can be produced using NI-IMAQ and StillColor or color IMAQ devices.

RGB

The most common image representation is 32-bit RGB format. In this representation, the three 8-bit color planes—red, green and blue—are packed into an array of 32-bit integers. This representation is useful for displaying the image on your monitor. The 32-bit integer organized as:

0	RED	GREEN	BLUE
---	-----	-------	------

where the high-order byte is not used and blue is the low-order byte.

StillColor also supports a 24-bit and a 16-bit representation of the RGB image. The 24-bit representation is equivalent to the 32-bit representation; however, there is no unused byte. For the 16-bit representation, the image is packed into an array of 16-bit integers where each 16-bit pixel contains red, green, and blue, encoded with only five bits each. The most significant bit of the integer is always 0.

Color Planes

Each color plane can be returned individually. The red, green, or blue plane is extracted from the RGB image and represented as an array of 8-bit integers.

Hue, Saturation, Luminance, and Intensity Planes

The hue, saturation, luminance, and intensity planes can also be returned individually if you want to analyze the image. You can retrieve the data in 8-bit format to reduce the amount of data to be processed, or, if you are using StillColor, you can retrieve the data in 16-bit format to take advantage of the higher precision available when using averaging.

The 16-bit image representation available in StillColor is scaled so that the pixel values are always positive. The value range is 0 to +32,767, so it is compatible with both 16-bit signed and 16-bit unsigned integers. On average, the 16-bit representation of a plane is equal to 128 times the 8-bit representation of the plane from the same image. The 16-bit representation is generally only used if you are performing averaging on your image. For example, averaging an image 16 times requires four extra bits ($16 = 2^4$) to represent the increased dynamic range. In this case, using the 16-bit representation may increase the dynamic range of your image.

Luminance, Intensity, Hue, and Saturation are defined using the Red, Green, and Blue values in the following formulas:

$$\text{Luminance} = 0.299 \times \text{Red} + 0.587 \times \text{Green} + 0.114 \times \text{Blue}$$

$$\text{Hue} = \text{ATN2}(Y, X)$$

where

$$Y = (\text{Green} - \text{Blue}) / \sqrt{2} \quad \text{and}$$

$$X = (2 \times \text{Red} - \text{Green} - \text{Blue}) / \sqrt{6}$$

$$\text{Saturation} = 255 \times \left(1 - \frac{3 \times \text{Min}(\text{R}, \text{G}, \text{B})}{\text{R} + \text{G} + \text{B}} \right)$$

For StillColor images:

$$\text{Intensity} = (\text{Red} + \text{Green} + \text{Blue}) / 3$$

$$\text{Saturation} = \sqrt{X^2 + Y^2}$$

where

$$Y = (\text{Green} - \text{Blue}) / \sqrt{2} \text{ and}$$

$$X = (2 \times \text{Red} - \text{Green} - \text{Blue}) / \sqrt{6}$$

32-Bit HSL and HSI

You can also pack the three 8-bit Hue, Saturation, and Luminance planes (HSL). In StillColor, you can pack the three Hue, Saturation, and Intensity planes (HSI) in one array of 32-bit integers, which is equivalent to the 32-bit RGB representation.

Camera Types

You can use two basic video camera types for color acquisition—RGB cameras and composite color video cameras.

An RGB camera delivers the three basic color components—red, green and blue—on three different wires. This type of camera often uses three independent CCD sensors to acquire the three color signals. RGB cameras are used for very accurate color acquisition.

A composite color camera transmits the video signal on a single wire. The signal is composed of two components that are added together. These components are:

- A monochrome video signal that contains the gray level information from the image and the composite synchronization signals. This signal is the same as a standard monochrome video signal, such as RS-170 (NTSC) or CCIR-601 (PAL).
- A modulated signal that contains the color information from the image. The format of this signal depends on your camera. The three main color standards are as follows:
 - M-NTSC (also called NTSC), which is used mainly in the U.S. and Japan
 - B/G-PAL (also called PAL), which is used mainly in Europe, India, and Australia

- SECAM, which is used mainly in France and the former Soviet Republics. SECAM is only used for broadcasting, so SECAM countries often use PAL as the local color image format.

StillColor Basics

StillColor is a technique you can use to acquire color images from composite color video or RGB cameras using the PCI/PXI-1408 monochrome device. Use StillColor Composite mode to acquire color images from a composite color video camera. Use StillColor RGB mode to acquire color images from an RGB camera. StillColor composite acquisition results in an image of much higher quality than the traditional color decoding that can be obtained with a color image acquisition board.

To acquire a color image, the PCI/PXI-1408 acquires multiple frames from the camera. Your computer CPU then processes the frames using the StillColor algorithm and creates a single color image. Because StillColor uses your computer CPU to process the image, the acquisition time for a single image depends on your system performance. You can acquire StillColor composite images at rates of up to 2 frames/s and StillColor RGB images at rates of up to 10 frames/s.

You can use StillColor in applications that require high-quality images of still or very slowly moving objects. StillColor supports many different image representations used in scientific or industrial applications, such as RGB bitmap and single plane hue, saturation, luminance, and intensity. StillColor also supports image averaging of up to 128 frames to increase the dynamic range of the StillColor image. See the [Introduction to Color](#) section in this appendix for more information on image representations.

StillColor Composite

In a composite color video signal, the color information (chroma) is modulated in phase and amplitude around a sub-carrier frequency of 3.58 MHz (NTSC) or 4.43 MHz (PAL). The modulated signal is then added to the luminance information and the entire signal, including synchronization pulses, is transmitted on a single line.

Traditional Color Decoding

On the receiver side or in your IMAQ board, the luminance and the chroma signals must be separated before the color image can be decoded and rebuilt. However, the modulated color information and some of the high-frequency luminance information share the same frequency range

around the sub-carrier frequency. This sharing makes it impossible to separate the two signals perfectly and, therefore, perfect reconstruction of the original color image is not possible.

All of the traditional ways to separate the two signals result in visual artifacts on the final picture. Techniques such as frequency-band filtering or comb filtering can minimize some of these artifacts, but most techniques are optimized to obtain the best picture for visualization of a continuous acquisition. The composite color formats are designed so that artifacts resulting from one frame are almost cancelled by artifacts in following frames. This system takes advantage of the slow response time of the human eye to obscure most of these problems.

The situation is different in a single frame acquisition where a single image is needed. A single image usually clearly shows the result of a bad color/luminance separation. Typical weakness of traditional separation techniques are:

- Reduced luminance bandwidth, resulting in a blurry image
- Cross-color modulation where rapidly changing colors affect the luminance of the image, as shown on the edges of the parrot's head in Figure A-2
- Cross-luminance modulation where rapidly changing luminance (stripes) results in irritating random color patterns, as shown on the black and white stripes around the parrot's eye in Figure A-2



Figure A-2. Traditional Decoding

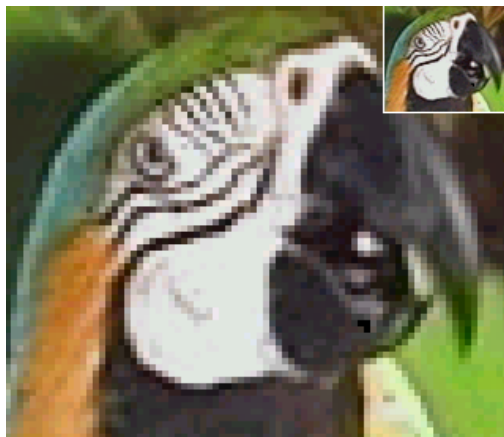


Figure A-3. StillColor Decoding



Note You can find color versions of the illustrations in this appendix in the online version of this document included with your NI-IMAQ software kit.

Both pictures are approximately 80×80 pixels in size and are acquired using an NTSC composite video signal. Figure A-2 uses classic decoding algorithms including bandpass/bandstop and comb filtering. Figure A-3 was acquired using the PCI/PXI-1408 and the StillColor algorithm.

Why StillColor?

StillColor is optimized for single-frame acquisition. A StillColor Composite acquisition acquires multiple consecutive frames. Assuming that all frames represent the same scene of still objects, the algorithm then uses knowledge about the composite color format to perfectly separate the color and the luminance information.

In an NTSC video signal, two consecutive frames representing the same object will contain the same luminance information but will have chroma signals that are opposite in phase. By adding the two frames together, the chroma information is cancelled, and by subtracting the two frames from each other, the luminance signal is cancelled. The resulting separation is now perfect, as shown in Figure A-3.

Color and luminance separation is more complex in a PAL video signal. The IMAQ device must acquire three consecutive frames, but the same perfect separation of the color and luminance information can be achieved after manipulation of these images.

After separating the color and luminance signals, the StillColor algorithm then decodes and rebuilds the color image. As shown in Figure A-3, the result does not show any of the artifacts encountered in traditional color decoding methods.

Composite Color Acquisition

The PCI/PXI-1408, in conjunction with NI-IMAQ, supports acquisition of color images from an NTSC or PAL composite color video camera.

NI-IMAQ can acquire the multiple frames, decode the color information, and rebuild the image automatically. The output image can be a simple RGB color image or one of many image representations supported by NI-IMAQ. See the [Introduction to Color](#) section in this appendix for more information on image representations.

You can connect the composite video signal to any of the four input channels on the PCI/PXI-1408. Since StillColor is used for still scenes, you can perform only a snap (a single-image acquisition).

StillColor RGB

RGB cameras output a color image using three lines. StillColor RGB will acquire the three signals and construct a color image. The three lines are connected to three channels on the PCI/PXI-1408. One frame is acquired from each of the three channels, which represent the red, green, and blue planes of the image. StillColor combines these frames to construct the color image.

RGB Color Acquisition

The PCI/PXI-1408, in conjunction with NI-IMAQ, supports acquisition of color images from an RGB camera.

The NI-IMAQ driver can acquire the three frames and rebuild the image automatically. The output image can be a simple RGB color image or one of many image representations supported by the driver. See the [Introduction to Color](#) section in this appendix for more information on image representations.

For a StillColor RGB snap, connect the three camera channels—red, green, and blue—to Video 1, Video 2, and Video 3, respectively, on the PCI/PXI-1408 device. Specify a channel for the video synchronization signal by selecting that channel as the sync source using the **Operating Mode** tab in the Measurement & Automation Explorer for IMAQ. A typical RGB camera includes the composite video synchronization signal in the green signal. You can also use other synchronization sources, such as an external composite video signal that can be connected to Video 0 or an external TTL composite synchronization signal that can be connected to the CSYNCIN pin of the DSUB connector. (See Chapter 4, *Signal Connections*, of your hardware user manual for signal connection information.)

Technical Support Resources

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

NI Web Support

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

Online Problem-Solving and Diagnostic Resources

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

Software-Related Resources

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

Worldwide Support

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

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

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

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

Glossary

Prefix	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

– negative of, or minus

Ω ohm

/ per

% percent

\pm plus or minus

+ positive of, or plus

A

A amperes

AC alternating current

acquisition window the image size specific to a video standard or camera resolution

active line region the region of lines actively being stored; defined by a line start (relative to VSYNC) and a line count

active pixel region	the region of pixels actively being stored; defined by a pixel start (relative to HSYNC) and a pixel count
A/D	analog-to-digital
ADC	analog-to-digital converter—an electronic device, often an integrated circuit, that converts an analog voltage to a digital number
address	character code that identifies a specific location (or series of locations) in memory
ANSI	American National Standards Institute
antichrominance filter	removes the color information from the video signal
API	application programming interface
AQ_DONE	signals that the acquisition of a frame or field is completed
AQ_IN_PROGRESS	signals that the acquisition of video data is in progress
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. <i>See also</i> white reference level.

buffer	temporary storage for acquired data
bus	the group of conductors that interconnect individual circuitry in a computer, such as the PCI bus; typically the expansion vehicle to which I/O or other devices are connected
C	
C	Celsius
cache	high-speed processor memory that buffers commonly used instructions or data to increase processing throughput
CCIR	Comite Consultatif International des Radiocommunications—a committee that developed standards for color video signals
chrominance	the color information in a video signal
CMOS	complementary metal-oxide semiconductor
compiler	a software utility that converts a source program in a high-level programming language, such as Basic, C or Pascal, into an object or compiled program in machine language. Compiled programs run 10 to 1,000 times faster than interpreted programs. <i>See also</i> Interpreter.
conversion device	device that transforms a signal from one form to another; for example, analog-to-digital converters (ADCs) for analog input and digital-to-analog converters (DACs) for analog output
CPU	central processing unit
CSYNC	composite sync signal; a combination of the horizontal and vertical sync pulses
D	
D/A	digital-to-analog
DAC	digital-to-analog converter; an electronic device, often an integrated circuit, that converts a digital number into a corresponding analog voltage or current

DAQ	data acquisition—(1) collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing; (2) collecting and measuring the same kinds of electrical signals with A/D or DIO 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: $\text{dB} = 20\log_{10} V_1/V_2$, for signals in volts
DC	direct current
default setting	a default parameter value recorded in the driver; in many cases, the default input of a control is a certain value (often 0) that means <i>use the current default setting</i>
DIN	Deutsche Industrie Norme
DLL	dynamic link library—a software module in Microsoft Windows containing executable code and data that can be called or used by Windows applications or other DLLs; functions and data in a DLL are loaded and linked at run time when they are referenced by a Windows application or other DLLs
DMA	direct memory access—a method by which data can be transferred to and from computer memory from and to a device or memory on the bus while the processor does something else; DMA is the fastest method of transferring data to/from computer memory
DRAM	dynamic RAM
drivers	software that controls a specific hardware device such as an IMAQ or DAQ device
dynamic range	the ratio of the largest signal level a circuit can handle to the smallest signal level it can handle (usually taken to be the noise level), normally expressed in decibels
E	
EEPROM	electrically erasable programmable read-only memory—ROM that can be erased with an electrical signal and reprogrammed

external trigger a voltage pulse from an external source that triggers an event such as A/D conversion

F

field For an interlaced video signal, a field is half the number of horizontal lines needed to represent a frame of video; the first field of a frame contains all the odd-numbered lines, the second field contains all of the even-numbered lines.

FIFO first-in first-out memory buffer—the first data stored is the first data sent to the acceptor; FIFOs are used on IMAQ devices to temporarily store incoming data until that data can be retrieved.

flash ADC an ADC whose output code is determined in a single step by a bank of comparators and encoding logic

frame a complete image; in interlaced formats, a frame is composed of two fields

front porch the area of a video signal between the start of the horizontal blank and the start of the horizontal sync

ft feet

function a set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed; examples of functions are:

$$y = \text{COS}(x)$$

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

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

HSYNC horizontal sync signal—the synchronization pulse signal produced at the beginning of each video scan line that keeps a video monitor's horizontal scan rate in step with the transmission of each new line

hue represents the dominant color of a pixel. The hue function is a continuous function that covers all the possible colors generated using the R, G, and B primaries. *See also* RGB.

Hz hertz—the number of scans read or updates written per second

I

IC integrated circuit

ID identification

IEEE Institute of Electrical and Electronics Engineers

in. inches

INL integral nonlinearity—A measure in LSB of the worst-case deviation from the ideal A/D or D/A transfer characteristic of the analog I/O circuitry

instrument driver a set of high-level software functions, such as NI-IMAQ, that controls specific plug-in computer boards; instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW

interlaced a video frame composed of two interleaved fields; the number of lines in a field are half the number of lines in an interlaced frame

interpreter	a software utility that executes source code from a high-level language such as Basic, C or Pascal, by reading one line at a time and executing the specified operation. <i>See also</i> compiler.
interrupt	a computer signal indicating that the CPU should suspend its current task to service a designated activity
interrupt level	the relative priority at which a device can interrupt
I/O	input/output—the transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces
IRE	a relative unit of measure (named for the Institute of Radio Engineers). 0 IRE corresponds to the blanking level of a video signal, 100 IRE to the white level. Note that for CIR/PAL video the black level is equal to the blanking level or 0 IRE, while for RS-170/NTSC video the black level is at 7.5 IRE.
IRQ	interrupt request
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 Measurement & Automation Explorer for IMAQ that contains formulas that let you implement simple imaging operations such as contrast enhancement, data inversion, gamma manipulation, or other nonlinear transfer functions

M

m	meters
M	(1) Mega, the standard metric prefix for 1 million or 10^6 , when used with units of measure such as volts and hertz; (2) mega, the prefix for 1,048,576, or 2^{20} , when used with B to quantify data or computer memory
MB	megabytes of memory
Mbytes/s	a unit for data transfer that means 1 million or 10^6 bytes/s
memory buffer	<i>See</i> buffer.
memory window	continuous blocks of memory that can be accessed quickly by changing addresses on the local processor
MSB	most significant bit
MTBF	mean time between failure
mux	multiplexer—a switching device with multiple inputs that selectively connects one of its inputs to its output

N

NI-IMAQ	driver software for National Instruments IMAQ hardware
noninterlaced	a video frame where all the lines are scanned sequentially, instead of divided into two frames as in an interlaced video frame
NTSC	National Television Standards Committee—the committee that developed the color video standard used primarily in North America, which uses 525 lines per frame. <i>See also</i> PAL.

NVRAM nonvolatile RAM—RAM that is not erased when a device loses power or is turned off

0

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

PCLK pixel clock signal—times the sampling of pixels on a video line

PCLKIN pixel clock in signal

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
TRIG	trigger signal
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

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

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

W

white reference level the level that defines what is white for a particular video system
See also black reference level.

Y

YUV a representation of a color image used for the coding of NTSC or PAL video signals. The luminance information is called Y, while the chrominance information is represented by two components, U and V representing the coordinates in a color plane.

Index

A

- acquisition functions, 2-5
- application development, 1-2 to 1-4
 - creating applications, 1-3
 - NI-IMAQ libraries, 1-2
- attribute functions, 2-6

B

- buffer management functions
 - list of functions, 2-7
 - programming considerations, 3-4

C

- camera attributes, 3-4 to 3-5
- color, A-1 to A-4. *See also* StillColor.
 - definition, A-1
 - hue, A-2
 - image representations, A-2 to A-4
 - 32-bit HSL and HSI, A-4
 - color planes, A-3
 - hue, saturation, luminance, and intensity planes, A-3 to A-4
 - RGB, A-2
 - intensity, A-2
 - luminance, A-2
 - perception of color, A-2
 - saturation, A-2
 - white light and visible spectrum (figure), A-1
- composite color cameras, A-4

D

- documentation
 - conventions used in manual, *ix*
 - how to use NI-IMAQ manual set, *ix*
 - National Instruments documentation, *x*
 - related documentation, *x*
- dynamic link libraries (DLLs), 1-2

E

- e-mail support, B-2
- example programs
 - advanced programming examples, 3-15 to 3-18
 - low-level grab functions, 3-16
 - low-level ring functions, 3-17
 - low-level sequence functions, 3-16 to 3-17
 - low-level snap functions, 3-15 to 3-16
 - StillColor snap programming, 3-17 to 3-18
 - introductory programming examples, 3-7 to 3-15
 - high-level grab functions, 3-8 to 3-10
 - high-level ring functions, 3-11 to 3-13
 - high-level sequence functions, 3-10 to 3-11
 - high-level signal I/O functions, 3-14 to 3-15
 - high-level snap functions, 3-7 to 3-8
 - location (note), 3-7
- location of files, 1-4

F

- fax and telephone support numbers, B-2
- Fax-on-Demand support, B-1
- files required for application development, 1-3
- FTP support, B-1
- functions
 - generic functions, 2-2
 - high-level functions, 2-2 to 2-4
 - grab functions, 2-2 to 2-3
 - miscellaneous functions, 2-4
 - programming considerations, 3-1
 - programming examples, 3-7 to 3-15
 - ring and sequence functions, 2-3
 - session functions, 3-3 to 3-4
 - signal I/O functions, 2-3 to 2-4
 - snap functions, 2-2
 - low-level functions, 2-5 to 2-7
 - acquisition functions, 2-5
 - attribute functions, 2-6
 - buffer management functions, 2-7 to 2-8, 3-4
 - interface functions, 2-7, 3-2 to 3-3
 - programming considerations, 3-2
 - programming examples, 3-15 to 3-18
 - utility functions, 2-8
- overview, 2-1

G

- generic functions, 2-2
- grab acquisition, 3-1
- grab functions
 - list of functions, 2-2 to 2-3
 - programming examples
 - high-level functions, 3-8 to 3-9
 - low-level functions, 3-16

H

- header files, 1-3
- high-level functions, 2-2 to 2-4
 - grab functions, 2-2 to 2-3
 - introductory programming examples, 3-7 to 3-15
 - grab functions, 3-8 to 3-10
 - ring functions, 3-11 to 3-13
 - sequence functions, 3-10 to 3-11
 - signal I/O functions, 3-14 to 3-15
 - snap functions, 3-7 to 3-8
 - miscellaneous functions, 2-4
 - programming considerations, 3-1
 - ring and sequence functions, 2-3
 - signal I/O functions, 2-3 to 2-4
 - snap functions, 2-2
- hue, A-2
- hue planes, A-3 to A-4

I

- import libraries
 - IMAQ.LIB required for application development, 1-3
 - location (table), 1-3
 - purpose and use, 1-2
- intensity, A-7
- intensity planes, A-3 to A-4
- interface functions
 - interface naming convention (table), 3-2
 - list of functions, 2-7
 - programming considerations, 3-2 to 3-3

L

line scan image acquisition, 3-7

low-level functions, 2-5 to 2-7

- acquisition functions, 2-5
- advanced programming examples, 3-14 to 3-18
 - grab functions, 3-16
 - ring functions, 3-17
 - sequence functions, 3-16 to 3-17
 - snap functions, 3-15 to 3-16
- attribute functions, 2-6
- buffer management functions, 2-7 to 2-8, 3-4
- interface functions, 2-7, 3-2 to 3-3
- programming considerations, 3-2
- utility functions, 2-8

luminance, A-2

luminance planes, A-3 to A-4

M

manual. *See* documentation.

miscellaneous high-level functions, 2-4

N

NI-IMAQ header files, 1-3

NI-IMAQ libraries, 1-2

NI-IMAQ software. *See also* functions.

- application development, 1-2 to 1-4
- application development environments, 1-2
- features and overview, 1-1
- NI-IMAQ libraries, 1-2
- sample programs, 1-4

_NIWIN constant, 1-3

P

programming

- advanced examples, 3-14 to 3-17
 - low-level grab functions, 3-16
 - low-level ring functions, 3-17
 - low-level sequence functions, 3-16 to 3-17
 - low-level snap functions, 3-15 to 3-16
 - StillColor snap programming, 3-17 to 3-18
- buffer management, 3-4
- camera attributes, 3-4 to 3-5
- high-level functions, 3-1
- interface functions, 3-2 to 3-3
- introductory examples, 3-7 to 3-15
 - high-level grab functions, 3-7 to 3-10
 - high-level ring functions, 3-11 to 3-13
 - high-level sequence functions, 3-10 to 3-11
 - high-level signal I/O functions, 3-14 to 3-15
 - high-level snap functions, 3-7 to 3-8
 - location (note), 3-7
 - low-level functions, 3-2
 - session functions, 3-3 to 3-4
 - status signals, 3-5 to 3-6
- programming environments supported by NI-IMAQ software, 1-2

R

RGB cameras, A-4

RGB image representations, A-2 to A-3

ring functions. *See also* sequence functions.

- list of functions, 2-3
- programming example
 - high-level functions, 3-11 to 3-13
 - low-level functions, 3-17

S

- sample programs. *See* example programs.
- saturation, A-2
- saturation planes, A-3 to A-4
- sequence acquisition, 3-1
- sequence functions. *See also* ring functions.
 - list of functions, 2-3
 - programming example
 - high-level functions, 3-10 to 3-11
 - low-level functions, 3-16 to 3-17
- session functions
 - programming considerations, 3-3 to 3-4
- signal I/O functions
 - list of functions, 2-3 to 2-4
 - programming example, 3-14 to 3-15
- snap acquisition, 3-1
- snap functions
 - list of functions, 2-2
 - programming example
 - high-level functions, 3-7 to 3-8
 - low-level functions, 3-15 to 3-16
- status signals, 3-5 to 3-6
- StillColor, A-5 to A-9
 - advantages, A-8
 - composite color acquisition, A-8
 - composite color video signals, A-5 to A-9
 - overview, A-5
 - purpose and use, A-5
 - RGB color acquisition, A-9
 - snap programming example, 3-17 to 3-18
 - traditional color decoding, A-5 to A-7

T

- technical support, B-1 to B-2
- telephone and fax support numbers, B-2

U

- utility functions, 2-8

V

- video cameras for color acquisition,
 - A-4 to A-5
- visible spectrum, A-1 to A-2

W

- white light, A-1