Related Documentation

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

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

- NI Vision Acquisition Software Release Notes—Contains information about new functionality, minimum system requirements, and installation instructions for NI-IMAQ driver software.
- *Measurement & Automation Explorer for NI-IMAQ*—Explains how to configure NI-IMAQ, an image acquisition device, and a camera using Measurement & Automation Explorer (MAX).
- *NI-IMAQ Function Reference Help*—Contains reference information about the LabWindows/CVI functions for NI-IMAQ driver software.
- *NI-IMAQ VI Reference Help*—Contains reference information about the LabVIEW palettes and VIs for NI-IMAQ driver software.
- NI Developer Zone—For more information about developing your image acquisition application, visit the <u>NI Developer Zone</u> at ni.com/zone. NI Developer Zone contains example programs, tutorials, technical presentations, the Instrument Driver Network, a measurement glossary, an online magazine, and a product advisor, as well as a community area where you can share ideas, questions, and source code with developers around the world.

Using Help

<u>Conventions</u> <u>Navigating Help</u> <u>Searching Help</u> <u>Printing Help File Topics</u>

Conventions

This help file uses the following conventions:

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

Navigating Help (Windows Only)

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

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

Searching Help (Windows Only)

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

Wildcards

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

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



Note Wildcard searching will not work on Simplified Chinese, Traditional Chinese, Japanese, and Korean systems.

Nested Expressions

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

Boolean Expressions

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

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

Search Options

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

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

Printing Help File Topics (Windows Only)

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

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

Printing PDF Documents

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

Imaging Fundamentals

This section contains information about the nomenclature and concepts related to image acquisition and machine vision applications.

Digital Images Cameras Color Basics

Quadrature Encoders

Digital Images

This section contains information about the definition and properties of digital images, image types, and file formats.

Definition of a Digital Image

Properties of a Digital Image

Image Files

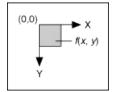
Definition of a Digital Image

An image is a two-dimensional array of values representing light intensity. For the purposes of image processing, the term *image* refers to a digital image. An image is a function of the light intensity

f(x, y)

where f is the brightness of the point (x, y) and x and y represent the spatial coordinates of a picture element (abbreviated *pixel*).

By convention, the spatial reference of the pixel with the coordinates (0,0) is located at the top, left corner of the image. Notice in the following image that the value of *x* increases moving from left to right, and the value of *y* increases from top to bottom.



In digital image processing, an imaging sensor converts an image into a discrete number of pixels. The imaging sensor assigns each pixel a numeric location and a gray level or color value that specifies the brightness or color of the pixel.

Properties of a Digital Image

A digitalized image has three basic properties: resolution, definition, and number of planes.

Spatial Resolution

The spatial resolution of an image is its number of rows and columns of pixels.

Image Definition

The definition of an image indicates the number of shades that you can see in an image. The bit depth of an image is the number of bits used to encode the value of a pixel. For a given bit depth of n, the image has an image definition of 2^n , meaning a pixel can have 2^n different values. For example, for a bit depth of 8 bits, a pixel can take 256 different values ranging from 0 to 255. For a bit depth of 16 bits, a pixel can take 65,536 different values ranging from 0 to 65,535 or from -32,768 to 32,767.

The manner in which you encode your image depends on the nature of the image acquisition device, the type of image processing you need to use, and the type of analysis you need to perform. For example, 8-bit encoding is sufficient if you need to obtain the shape information of objects in an image. However, if you need to precisely measure the light intensity of an image or region in an image, you should use 16-bit or floating-point encoding.

Use color encoded images when your machine vision or image processing application depends on the color content of the objects you are inspecting or analyzing.

Number of Planes

The number of planes in an image corresponds to the number of arrays of pixels that compose the image. A grayscale or pseudo-color image is composed of one plane. A true-color image is composed of three planes —one each for the red component, green component, and blue component.

In true-color images, the color component intensities of a pixel are coded into three different values. The color image is the combination of three arrays of pixels corresponding to the red, green, and blue components in an <u>RGB</u> image. <u>HSL</u> images are defined by their hue, saturation, and luminance values. Refer to <u>Color Basics</u> for more information about color images and when they are used.

Image Files

An image file is composed of a header followed by pixel values. Depending on the file format, the header contains image information about the horizontal and vertical resolution, pixel definition, and the original palette. Image files may also store information about calibration, pattern matching templates, and overlays. The following are common image file formats:

- Bitmap (BMP)
- Tagged image file format (TIFF)
- Portable network graphics (PNG)—Offers the capability of storing image information about spatial calibration, pattern matching templates, and overlays
- Joint Photographic Experts Group format (JPEG)
- National Instruments internal image file format (AIPD)—Used for saving floating-point, complex, and HSL images

Standard formats for 8-bit grayscale and RGB color images are BMP, TIFF, PNG, JPEG, and AIPD. Standard formats for 16-bit grayscale and complex images are PNG and AIPD.

Cameras

There are many different types of cameras available for your image acquisition applications. You should use the parameters of your application to decide which type of camera system is appropriate for your needs.

The National Instruments Industrial Camera Advisor at ni.com/zone offers a one-stop Web resource for selecting an imaging camera. This virtual catalog provides features and specifications for more that 150 cameras so that you can compare cameras by category (such as analog, digital, line scan, area scan, or progressive scan), by feature, or by manufacturer.

The following sections provide information about cameras:

Area Scan vs. Line Scan

<u>Analog</u>

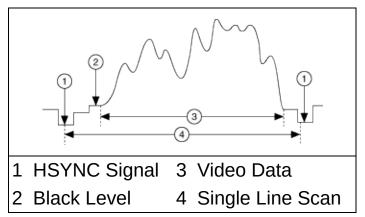
Digital

Area Scan vs. Line Scan

Cameras use different methods of acquiring the pixels of an image. Two commonly used methods are area scan and line scan. An area scan camera acquires an area of pixels at a time until the entire image is acquired. Most applications use this type of camera. A line scan camera scans only one line of pixels at a time, providing faster acquisition. However, the lines must be fit together by your acquisition hardware and software to create a whole image. Line scan cameras are useful in web inspection applications during which the object under inspection moves along a conveyor or stage in a production system. Line scan cameras are also useful in high-resolution applications because you can arbitrarily lengthen the image by fitting a specified number of lines together.

Analog Cameras

Analog cameras output a video signal in an analog format. The horizontal sync (HSYNC) pulse identifies the beginning of each line; multiple lines make up a field. An additional pulse, called the vertical sync (VSYNC), identifies the beginning of a field. For most traditional analog cameras, the odd and even fields are interlaced to increase the perceived image update rate. Two fields make up one frame. Some higher-end cameras expose the entire image at once. The following figure illustrates the analog output of a video signal.



The black level is a reference voltage for measuring pixel intensities. Low voltages typically indicate darker pixels, while higher voltages result in lighter pixels.

Standard Analog Video Formats

The following table describes the standard video formats used by analog imaging cameras. These formats vary in image size, color availability, and frame rate. Most analog cameras adhere to one of these four formats.

Format	Country	Frames per Second	Color	Image Size
RS-170	USA, Japan	30	No	640 × 480
NTSC	USA, Japan	30	Yes	640 × 480
CCIR	Europe	25	No	768 × 576
PAL	Europe	25	Yes	768 × 576

Digital Cameras

Digital cameras use three types of signals—data lines, a pixel clock, and enable lines. Data lines are parallel wires that carry digital signals corresponding to pixel values. Monochrome digital cameras typically represent pixels with 8, 10, 12, or 14 bits. Color digital cameras typically use up to 8 or 10 bits per color for each pixel. Depending on your camera, you may have as many as 30 data lines, or wires, representing one pixel. The number of data lines per pixel determines bit depth.

The pixel clock is a high-frequency pulse train that indicates when the data lines contain valid data. On the active edge of the pixel clock—which can be either the rising edge or the falling edge, depending on the camera—the value of the digital lines is input into your image acquisition device. The pixel clock frequency determines the rate that pixels are acquired.

Enable lines indicate when data lines contain valid data. The Line Valid signal in digital cameras provides the same type of information about where each line is located as the HSYNC signal provides for analog cameras. In digital cameras, the Line Valid signal is usually active for the entire duration of the line, rather than indicating only the start of the line. At the end of that row, the Line Valid signal goes inactive until the next row of pixels begins. The Frame Valid signal in digital cameras provides the same type information about where each line is located as the VSYNC signal provides for analog cameras. In digital cameras, the Frame Valid signal is active during the acquisition of an entire frame. At the end of that frame, the Frame Valid signal goes inactive until the beginning of the next frame.

Digital line scan cameras consist of a single row of pixel elements and require only a Line Valid timing signal. Area scan cameras provide both the Line Valid and Frame Valid signals.

Taps

Increasing the speed of the digital camera pixel clock or acquiring more than one pixel at a time can greatly improve camera acquisition speed. A tap is a group of data lines that together carry one pixel. A camera that latches only one pixel during the active edge of the pixel clock is known as a single tap camera. Other cameras have multiple pixels on separate data lines that are all available during the active edge of the pixel clock. These multi-tap cameras are available with as many as 10 taps. Cameras with multiple taps require more data lines but provide faster data transfer.

Camera Files

Because digital cameras vary in specifications such as speed, image size, pixel depth, number of taps, and modes, NI-IMAQ requires a camera file specific to your camera to define all of these values in order to use that camera with your image acquisition device. You can find a list of camera files that have been tested and approved by National Instruments online at the National Instruments Industrial Camera Advisor at ni.com/zone. You also can create your own camera files using the NI Camera File Generator, which you can download from the National Instruments Machine Vision Web site at ni.com/vision.

Types of Digital Cameras

There are three main types of digital cameras: parallel, Camera Link, and IEEE 1394.

Parallel

Many older or specialized digital cameras use a parallel interface that provides a wide range of acquisition speeds, image sizes, and pixel depths. Parallel cameras often require users to customize their cables and connectors to suit their image acquisition device.

Camera Link

The <u>Camera Link</u> standard was developed to ease the challenges of the custom cable interface between parallel digital cameras and image acquisition devices. National Instruments, as part of the Camera Link Standards Committee consisting of camera and frame grabber manufacturers, developed this standard to offer speed and triggering functionality with the ease of standardized cabling. This standard allows for high-speed image capture.

IEEE 1394

The IEEE 1394 standard offers a simple daisy chain cabling system with a standard interface. IEEE 1394, however, lacks the data throughput capabilities of the parallel and Camera Link interfaces. IEEE 1394 also lacks trigger synchronization capabilities.

The following table describes the advantages and disadvantages of the three main digital interface standards.

Interface	Advantages	Disadvantages
Parallel	High Speed	Bulky cabling
standard		No physical or protocol standards for interfacing with image acquisition devices
IEEE 1394 standard	Simple cabling that allows daisy chaining and use of hubs and repeaters	Slower data transfer rate
	Low cost	
	No camera files required	

	High speed	More costly than IEEE 1394
Link	Uniform cables	10 m cable length limit
standard		

Camera Link

Developed by a consortium of camera and image acquisition device manufacturers, Camera Link is a standard for interfacing digital cameras with image acquisition devices. Camera Link simplifies connectivity between the image acquisition device and the camera by defining a single standard connector for both. This standard ensures compatibility of devices bearing the Camera Link logo.

The basis for the Camera Link standard is a serial data transmission method consisting of a general-purpose transmitter/receiver pair. Each driver takes 28 bits of parallel digital data and a clock and serializes the stream to four LVDS (EIA-644) data streams and an LVDS clock. The Camera Link standard provides high-speed data transmission across 10 wires over distances of up to 10 m.

Camera Link Base configuration uses one transmitter/receiver pair and requires one cable for data transmission. Camera Link Medium configuration uses two transmitter/receiver pairs. Camera Link Full configuration uses three transmitter/receiver pairs. Both Medium and Full configuration require two data cables.

Refer to the *Specifications of the Camera Link Interface Standard for Digital Cameras and Frame Grabbers* manual for more detailed information about Camera Link specifications. This manual is available on several Web sites, including the <u>Automated Imaging Association Web</u> <u>site</u> at www.machinevisiononline.org.

Color Basics

This section explains basic color theories, describes color image output options, and describes methods you can use to acquire a color image.

Introduction to Color Color Spaces Image Representations Color Camera Types

Introduction to Color

Color is the wavelength of the light the human eye receives when we look at an object. In theory, the color spectrum is infinite. Humans, however, can see only a small portion of this spectrumthe 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.

White light is a combination of all colors. 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 (RGB) 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:

- **Hue**—The perceived dominant color. Hue depends directly on the wavelength of a color.
- **Saturation**—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**—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.

Color Spaces

Color spaces allow you to represent a color. A color space is a subspace within a three-dimensional coordinate system where each color is represented by a point. You can use color spaces to facilitate the description of colors between persons, machines, or software programs.

Various industries and applications use a number of different color spaces. Humans perceive color according to parameters such as brightness, hue, and intensity, while computers perceive color as a combination of red, green, and blue. The printing industry uses cyan, magenta, and yellow to specify color.

The following is a list of common color spaces:

- **RGB**—Based on red, green, and blue. Used by computers to display images.
- **HSL**—Based on hue, saturation, and luminance. Used in image processing applications.
- **CIE**—Based on brightness, hue, and colorfulness. Defined by the Commission Internationale de l'Eclairage (International Commission on Illumination) as the different sensations of color that the human brain perceives.
- **CMY**—Based on cyan, magenta, and yellow. Used by the printing industry.
- **YIQ**—Separates the luminance information (Y) from the color information (I and Q). Used for TV broadcasting.

When to Use

You must define a color space every time you process color images. If you expect the lighting conditions to vary considerably during your color machine vision application, use the HSL color space. The HSL color space provides more accurate color information than the RGB space when running color processing functions, such as color matching, color location, and color pattern matching.

If you do not expect the lighting conditions to vary considerably during your application, and you can easily define the colors you are looking for using red, green, and blue, use the RGB space. Also, use the RGB space if you want to display, but not process, color images in your application. The RGB space reproduces an image as you would expect to see it.

Color Image Representations

Color images can be represented in 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 or color image acquisition devices.

32-Bit 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 the monitor. The 32-bit integer is organized as

0 RED GREEN BLUE

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

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.

64-Bit RGB

In the 64-bit RGB representation, the three 16-bit color planes are packed into an array of 64-bit integers, where the high-order byte is not used. This representation allows for more distinct colors per plane.

32-Bit HSL

In the 32-bit HSL representation, the three 8-bit color planes—hue, saturation, and luminance—are packed into an array of 32-bit integers in the same manner as the 32-bit RGB representation.

The hue, saturation, and luminance planes also can 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.

Color Camera Types

You can use three basic camera types for color acquisition—analog or digital RGB cameras, composite color cameras, and Bayer cameras.

RGB Cameras

RGB cameras deliver 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.

Composite Color Cameras

Composite color cameras transmit the video signal on a single wire. The signal is composed of two of the following components, which are added together:

- 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 or CCIR-601.
- A modulated signal that contains the color information from the image. The format of this signal depends on your camera. The three main analog color standards are as follows:
 - M-NTSC (also called NTSC) is used mainly in the U.S. and Japan.
 - B/G-PAL (also called PAL) is used mainly in Europe, India, and Australia.
 - SECAM, which is used mainly in France and the former Soviet Republics, is used only for broadcasting, so SECAM countries often use PAL as the local color image format.

Bayer Cameras

Bayer encoding is a method to produce color images with a single imaging sensor, as opposed to three individual sensors for the red, green, and blue components of light. This technology greatly reduces the cost of cameras.

The Bayer color filter array (CFA) is a primary color mosaic pattern of 50% green, 25% red, and 25% blue pixels. Green pixels comprise half of the total pixels because the human eye gets most of its sharpness information from green light.

Light travels through the camera lens onto an image sensor that provides one value for each sensor cell. The sensor is an array of tiny, lightsensitive diodes called photosites, which converts light into electrical charges. The sensor is covered by the Bayer CFA so that only one color value reaches any given pixel. The raw output is a mosaic of red, green, and blue pixels of different intensities.

When the image is captured, the accumulated charge for each cell is read and analog values are converted to digital pixel values using an analog-to-digital (A/D) converter.

Interpolation, sometimes referred to as demosaicing, fills in the missing colors. A decoding algorithm determines a value for the RGB components for each pixel in the array by averaging the color values of selected neighboring pixels and producing an estimate of color and intensity. This algorithm can be applied for the camera, the image acquisition device, or in the software. This algorithm is included in NI-IMAQ and NI-IMAQ for IEEE 1394 Cameras.

After the interpolation process is complete, the white balancing process further enhances the image by adjusting the red and blue signals to match the green signal in white areas of the image.

Quadrature Encoders

This section contains an overview of quadrature encoders and information about the scaled encoder signal used with NI-IMAQ.

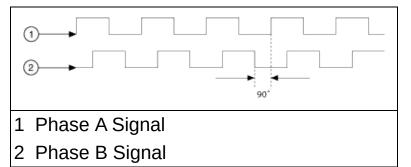
Quadrature Encoder Overview

Scaled Encoder Signal

Quadrature Encoder Overview

A quadrature encoder uses two output channels, Phase A and Phase B, to track the position of a rotary shaft. Generally, this shaft is coupled to a motor drive that controls the movement of an object of interest. By monitoring the encoder Phase A and Phase B signals, you can obtain a precise measurement of the object's position.

To generate Phase A and Phase B signals, the quadrature encoder uses two code tracks with sectors positioned 90 degrees out of phase. The phase difference indicates the position and direction of rotation. If Phase A leads Phase B, the shaft is rotating in a clockwise direction. If Phase B leads Phase A, the shaft is rotating in a counter-clockwise direction. The following figure illustrates the Phase A signal leading the Phase B signal.

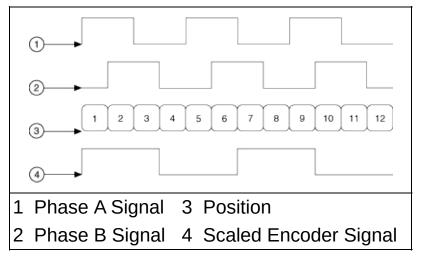


Compatible NI image acquisition devices include hardware that can be used to track both the position and direction of rotation of the Phase A and Phase B signals. For example, this information can be used in conjunction with a line scan camera to acquire lines synchronous to the movement of a conveyor belt. This gives you the ability to specify your line rate in terms of positional units (such as inches or centimeters) rather than time.

Scaled Encoder Signal

The scaled encoder signal is an edge-sensitive signal that is used to track cumulative forward progression of the quadrature encoder Phase A and Phase B signals. The scaled encoder signal is derived by applying a divide factor to the raw positional signal that is encoded between Phase A and Phase B.

All NI image acquisition devices expect the raw positional signal to be encoded with x4 encoding. The scaled encoder signal can be used as a line trigger, as a timebase for pulse generation, and it can be driven out on a trigger line for external usage. The following figure illustrates the scaled encoder signal that is produced when using a divide factor of six.



NI image acquisition devices that support multiple ports have a unique scaler per port. The unique scaler allows you to simultaneously acquire from multiple line scan cameras using different line rates that are all synchronous to the same quadrature encoder. Some NI image acquisition devices also support querying the absolute position counter value. Refer to the image acquisition device documentation to determine if the device supports querying the absolute position counter.

Refer to the <u>Quadrature Encoder Overview</u> for more information about quadrature encoders.

Programming with NI-IMAQ

Using NI-IMAQ you can program your image acquisition device to acquire, display, and save images. You can use NI-IMAQ with other National Instruments software for a complete image acquisition and analysis solution. NI-IMAQ works with LabVIEW, LabWindows/CVI, and other common programming languages. National Instruments Vision software adds powerful image processing and analysis to these programming environments. You also can use Vision Assistant to quickly and easily prototype your image analysis applications.

The following sections describe how to use NI-IMAQ to program your image acquisition device.

Building Applications with NI-IMAQ

Programming with NI-IMAQ Functions

Programming with NI-IMAQ VIs

Programming with ActiveX Controls

Variable Height Acquisition

NI-IMAQ and the LabVIEW Real-Time Module

Building Applications with NI-IMAQ

The following sections describe important fundamentals for developing applications using NI-IMAQ.

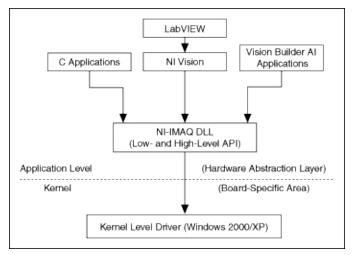
Architecture

NI-IMAQ libraries

Sample Programs

Architecture

The following block diagram of the NI-IMAQ architecture illustrates the low- and mid-level architecture for NI image acquisition devices.



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

NI-IMAQ Libraries

The NI-IMAQ 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, indicating the presence and location of the DLL routines. Depending on the development tools you use, you may give the DLL routines information through import libraries or function declarations. Your NI-IMAQ software kit contains function prototypes for all routines.

Sample Programs

Refer to the readme.rtf 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.

Programming with NI-IMAQ Functions

The following sections describe important fundamentals of programming your image acquisition device using NI-IMAQ.

Function Overview

Establishing Interface Connections and Sessions

Managing Buffers

Camera Attributes

NI-IMAQ Status Signals

Line Scan Image Acquisition

Geometric Definitions

Programming Examples

Refer to the *NI-IMAQ Function Reference Help* for more information about NI-IMAQ programming functions.

Creating an Application Using C

This section outlines the process for developing NI-IMAQ applications using C. Detailed instructions about creating project and source files are not included. Refer to the documentation included with your particular development environment for information about creating and managing project files.

When programming, use the following guidelines:

- Include the NIIMAQ.H header file in all C source files that use NI-IMAQ functions. Add this file to the top of your source files.
- 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, indicate where the compiler can find the NI-IMAQ header files and shared libraries. You can find most of the files you need for development 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. You can find the include files under the include subdirectory. The import libraries for Microsoft Visual C++ are located under the lib\msvc subdirectory.

Function Overview

The NI-IMAQ application programming interface (API) is divided into three groups: high-level functions, low-level functions, and generic 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 the image acquisition process, but you must understand the API and driver in greater detail. Generic functions allow you to set up interfaces and sessions and close both when you are finished.

 $\overline{\mathbb{N}}$

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 Help*. Changing the value of these attributes while using low-level functions affects the operation of the high-level functions.

Interface Functions

Interface functions can be used in combination with both high- and low-level functions to perform the following tasks:

- Set up the interface and session.
- Close the interface and session when you are finished with the application.
- Obtain an interface name.
- Reset the interface.

High-Level Functions

High-level functions support 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

<u>Low-level functions</u> support all types of acquisition. Use low-level functions to perform the following tasks:

- Create a custom acquisition sequence or ring.
- Create and manage 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.

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. Refer to the following sections for information about managing interfaces and sessions. Then refer to the high-level or low-level samples for information about acquiring images.

Interface Functions

Use interface functions to query the number of available interfaces, establish a connection to, control access to, and initialize hardware. All parameters configured in MAX for an image acquisition device are associated with an interface name. You can have one device associated with more than one interface name, which allows you to have several different configurations for one device. Use the interface name to refer to the device in the programming environment.

Interface name information is stored in an interface (.iid) file and includes the device serial number and the camera file associated with each channel or port on the device. NI-IMAQ specifies all interfaces by a name. By default, the system creates default names for the number of devices in the system. These names observe the convention shown in the following table.

Interface Name	Device Installed
img0	Device 0
img1	Device 1
imgn	Device n

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

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

You can edit existing interfaces or create new interfaces by using MAX. You also can use MAX to configure 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 uses this handle to reference this interface when using other NI-IMAQ functions.

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

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

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

Refer to the *NI-IMAQ Function Reference Help* for a complete list of the available interface functions.

Session Functions

Use session functions to configure the type of acquisition you want to perform using a particular interface. After you have established a connection to an interface, 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 named 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.

Refer to the *NI-IMAQ Function Reference Help* for a complete list of the available session functions.

Managing Buffers

NI-IMAQ can automatically perform buffer management, or you can perform buffer management manually. If the high-level acquisition routines (imgSnap, imgGrab, imgSequenceSetup, and imgRingSetup) are initialized with NULL pointers for buffer addresses, NI-IMAQ automatically allocates a buffer and returns the value of the buffer pointer. After you obtain a buffer pointer, you can use this pointer in successive calls.

For greater control of the acquisition buffers, create buffers with a memory allocation routine (for example, malloc) or use 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.

Camera Attributes

Camera attributes allow you to control camera-specific functions, such as integration time and pixel binning, directly from NI-IMAQ. You can set camera attributes in MAX. Information about specific attributes for some cameras is contained in a camera attribute file <my camera>.txt, which is in the <NI-IMAQ>/Camera Info directory, where <NI-IMAQ> is the location to which NI-IMAQ is installed.

All parameters configured for a camera type are stored in a camera (.icd) file. The camera file is then associated with a specific channel or port on an image acquisition device. The camera file includes information about the video signal timing and the input signal range of the video signal.

The camera attribute file lists all attributes for the camera. Each attribute description contains the following fields:

- Attribute Name contains the name of the attribute in quotes.
- **Description** contains a brief description of the camera.
- Data Type 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.
 - **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.
- **Possible Values** contains a list of valid **String**, **Integer**, or **Float** values.

Use the imgSetCameraAttributeString and imgGetCameraAttributeString functions to set and get the value of **String** 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.

Many cameras are configured with serial commands. Many camera files provided by National Instruments are already programmed with the camera serial command set. When you use the imgSetCameraAttributeString or imgSetCameraAttributeNumeric function, any serial commands programmed into the camera file are automatically sent to the camera when imgSessionOpen is called. If you need more low-level control over the serial communication between the camera and your image acquisition device, use the imgSessionSerial functions.

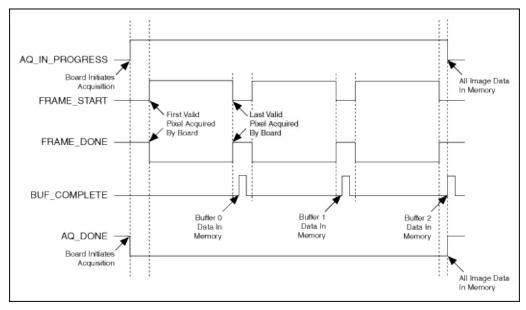
NI-IMAQ Status Signals

NI-IMAQ contains several status signals that you can use to trigger the generation of a pulse or invoke a callback function. **Acquisition in Progress** (AQ_IN_PROGRESS) indicates that the device is acquiring image data. This signal becomes TRUE when the device initiates the acquisition either through a software or hardware triggered start. When the last piece of image data is transferred to memory, this signal becomes FALSE. If the acquisition until the entire sequence is completed. **Acquisition Done** (AQ_DONE) is the inverse of **Acquisition in Progress**. This signal becomes TRUE when the last piece of data is transferred to memory, indicating that the acquisition has completed.

Frame Start (FRAME_START) and Frame Done (FRAME_DONE) indicate the status of an acquisition on a buffer basis. Frame Start indicates that a buffer is being acquired. This signal becomes TRUE when the device detects the first valid pixel in the current region of interest. The signal becomes FALSE when the device detects the last valid pixel of the frame. If the acquisition is a sequence, a ring, or a grab, Frame Start and Frame Done pulse for every buffer in the acquisition. Frame Done is the inverse of Frame Start and indicates when the image is transferred from the camera to the image acquisition device.

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

The following figure illustrates the values of the signals during a threebuffer sequence acquisition.



You can use the NI-IMAQ status signals for many purposes. You can generate pulses based on the assertion of any of these signals. Pulse generation 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 by any of these signals. For example, to initiate an image processing routine as soon as an image is in memory, configure a callback containing image processing code. Invoke the callback when you assert **Buffer Complete**.

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 MAX and can be changed with the region of interest height attribute. You also can configure NI-IMAQ to acquire a variable number of lines. Refer to <u>Variable Height Acquisition</u> for more information.

Triggering line scan images is similar to triggering area scan images. Using the imgSessionTriggerConfigure2 function, you can trigger the start of each buffer. The imgSessionLineTrigSource2 function also 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.

FRAME_START and FRAME_DONE status signals are not valid for a line scan acquisition unless each buffer is triggered. Skip count is not supported for line scan acquisitions. A continuous line scan acquisition into onboard memory is not supported. However, continuous line scan acquisition into system memory is supported.

Geometric Definitions

The following list defines several terms you should be familiar with when performing image acquisition tasks:

- A sync window is the area defined by the horizontal synchronization pulse (HSYNC) and the vertical synchronization pulse (VSYNC).
- An acquisition window is the image size specific to a video standard or camera resolution. The default is set by your specific camera file. The window starting position varies according to camera.
- A region of interest (ROI) is a hardware-programmable rectangular portion of the acquisition window. This is a specific area of the image to acquire.

The following figure illustrates the geometric relationship of these terms.

Sync Window (Determined by	(0,0) Relative to Camera X and Y Offsets
HSYNC and VSYNC)	Acquisition Window (Determined by Camera Resolution) 640 x 480 Pixels
	ROI (Defined by Configuration) 300 x 200 Pixels
	(640,480)

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 the code examples discussed in this section in the <NI-IMAQ>\Sample directory, where <NI-IMAQ> is the location to which NI-IMAQ is installed.

Introductory Examples

High-Level Snap Functions High-Level Grab Functions High-Level Sequence Functions High-Level Ring Functions High-Level Signal I/O Functions

Advanced Examples

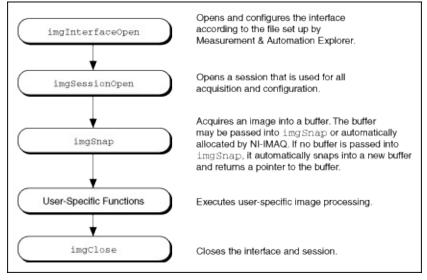
Performing a Snap Using Low-Level Functions Performing a Grab Using Low-Level Functions Performing a Sequence Acquisition Using Low-Level Functions Performing a Ring Acquisition Using Low-Level Functions

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 device and acquires the next incoming video frame (or field) to a buffer. Use a snap for low-speed or single-capture applications in which ease of programming is essential.

The following figure illustrates a typical snap programming order.



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 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 MAX. The ACQWINDOW dimensions vary depending on the type of camera you are using.

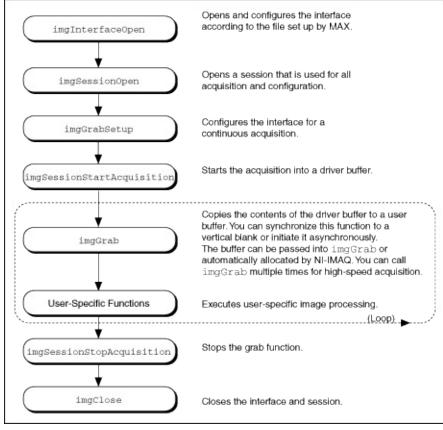
The sample then calls a process function to analyze the image. When the program is finished, it calls imgClose with the interface handle and sets the **freeResources** flag to TRUE. This instructs NI-IMAQ to free all of the resources associated with this interface, which releases the session as well 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. Obtain a copy of the acquisition buffer 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 copies the last acquired buffer into a user buffer where you can perform processing on the image. Use grab for high-speed applications where you need processing performed on only one image at a time.

The following figure illustrates a typical grab programming order.



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

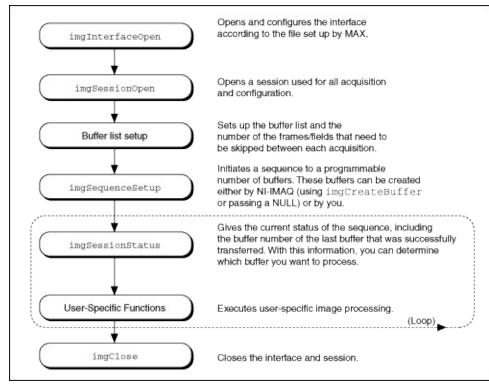
calculates the area to grab using the current ROI, rowPixels, and bytesPerPixel, and starts the acquisition by calling imgSessionStartAcquisition. In this example, the program allocates a user buffer for grabbing and passes this buffer to imgGrabArea. When the acquisition is complete, it stops. The program 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 imgSequenceSetup and specify both the buffer list 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.

Use a sequence in 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.

The following figure illustrates a typical sequence programming order.



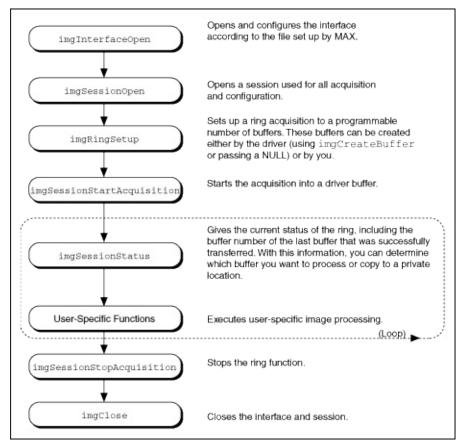
The HLSeq.c example demonstrates how to perform a sequence acquisition using imgSequenceSetup. The example sets up a sequence that uses ten user-allocated buffers. Each buffer in the sequence has its own skip count associated with it. The skip count is the number of frames to skip prior to acquiring 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 highspeed acquisition to multiple buffers. Calling imgRingSetup initiates a ring. imgRingSetup specifies both the buffer list used for transfers and the number of buffers. After you call imgRingSetup, you can monitor the status of the transfer and perform processing on any of the buffers in the ring. Use a ring 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 next pass of the ring. Use the imgSessionExamineBuffer and imgSessionReleaseBuffer functions to do this.

The following figure illustrates a typical ring programming order.



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 fourth 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 continues until a designated condition is met and then the acquisition stops.

High-Level Signal I/O Functions

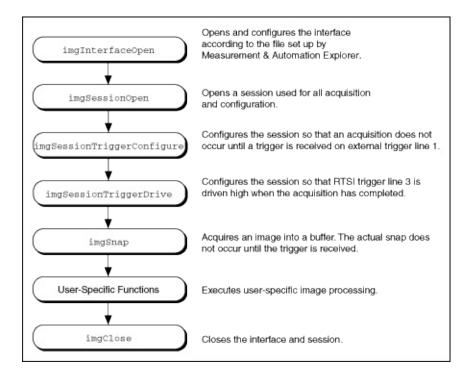
The signal I/O functions fall into two categories—triggering acquisitions and driving the trigger lines. Use triggered acquisitions to acquire images precisely when an external event occurs, such as a sensor activating. Driving trigger lines allows you to control external devices in sync with the image acquisition. For example, you can fire a strobe light before the acquisition of each frame in a sequence.

You can initiate any of the four types of acquisitions from an external trigger source by using imgSessionTriggerConfigure. For sequence and ring acquisitions, you can choose to trigger only the first buffer in the list, or you can choose to trigger each buffer in the list. After you use this function to set up the trigger, any acquisition performed on the session waits for a trigger. Use imgSessionTriggerClear to remove the trigger settings from the session.

Many machine vision cameras support asynchronous reset, which is an immediate frame output after a trigger. Most camera files supplied by National Instruments program the image acquisition device to send this trigger through the camera control lines. For applications using asynchronous reset, the external trigger signal should be routed to the image acquisition device, and imgSessionTriggerConfigure should be called to setup the device to trigger the camera.

Some applications need to send signals out from the image acquisition hardware to an external device. You can drive many types of signals out of the trigger lines by using imgSessionTriggerDrive. The parameters of this function include a trigger line number, the polarity of the line, and what to drive on the line. You can use a steady state value of high or low or one of the internal state signals of the hardware, such as acquisition in progress. When you need to generate specific pulses, use imgPulseCreate and imgPulseStart.

The following figure shows the outline of a program that waits for an external trigger on line 1 before acquiring a single image. The program 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.



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 32bit boundary to ensure that the image is acquired properly. The software does not perform this alignment for you unless you select a scaling option. 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 appropriate 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 memory requirement calculation is not necessary if you choose to use the default acquisition window width, rowPixels, and ROI. In this case, NI-IMAQ allocates the correct size buffer if you pass 0 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. The program creates the buffer and configures the buffer list for each buffer element in the ring. The program then locks the memory and starts the image acquisition 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 0xFFFFFFF 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 appropriate buffer from the live sequence. When you use imgSessionExamineBuffer, the driver does not allow you to write new data into that buffer during the analysis. Use imgSessionReleaseBuffer to return the buffer to the continuous sequence.

Programming with NI-IMAQ VIs

The following sections describe important fundamentals for programming your image acquisition device using NI-IMAQ VIs.

Overview

Location of NI-IMAQ Examples

Common NI-IMAQ VI Parameters

Managing Buffers in LabVIEW

NI-IMAQ Acquisition Types

Acquisition VIs

Triggering

Image Display

Camera Attributes

Error Handling

Refer to <u>NI-IMAQ and the LabVIEW Real-Time Module</u> for information about programming NI-IMAQ with the LabVIEW Real-Time Module.

Overview of NI-IMAQ VIs

LabVIEW is a development environment based on graphical programming. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution. The NI-IMAQ VI Library, a series of virtual instruments (VIs) for using LabVIEW with your image acquisition device, is included with the NI-IMAQ software.

NI Vision for LabVIEW is an image processing and analysis library that consists of more than 300 VIs. Some of the basic NI Vision VIs are shared with NI-IMAQ. If you use these basic functions, you can later upgrade your programs to use NI Vision without any changes to the image acquisition VIs.

Location of NI-IMAQ Examples

The NI-IMAQ VI examples illustrate some common applications. You can find examples through the NI Example Finder. To access the NI Example Finder, select **Find Examples** from the LabVIEW Help menu. Refer to the NI Example Finder Help for more information about finding examples using the NI Example Finder. Click the **Help** button in the NI Example Finder to display the NI Example Finder Help.

You also can find the NI-IMAQ VI examples in the

<LabVIEW>\examples\IMAQ directory, where <LabVIEW> is the location to which you installed LabVIEW. For a brief description of any example, open the example VI and choose **File»VI Properties»Documentation**. You also can display help for the VI by clicking the yellow question mark next to the VI icon in the block diagram or front panel.

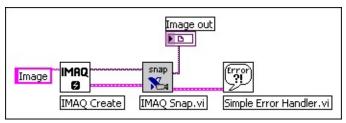
NI-IMAQ Acquisition Types

Four NI-IMAQ image acquisition types are available in LabVIEW: snap, grab, sequence, and ring. The following sections describe each acquisition type and give examples.

Snap

A snap acquires a single image into a memory buffer. Use this acquisition mode to acquire a single frame or field to a buffer. When you invoke a snap, it initializes the device and acquires the next incoming video frame (or field) to a buffer. Use a snap for low-speed or single-capture applications.

Use the IMAQ Snap VI for snap applications. The following diagrams shows a simplified block diagram for using IMAQ Snap.

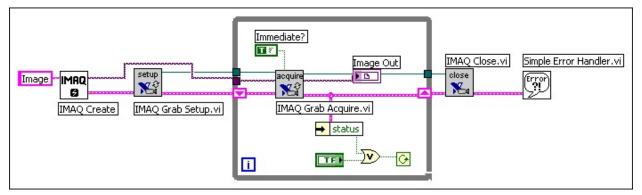


Grab

A grab is a continuous, high-speed acquisition of data to a single buffer in host memory. This function performs an acquisition that loops continually on one buffer. You can get a copy of the acquisition buffer by grabbing a copy to a LabVIEW image buffer.

You must use two VIs—IMAQ Grab Setup and IMAQ Grab Acquire—for a grab acquisition in LabVIEW. IMAQ Grab Setup, which you call only once, initializes the acquisition and starts capturing the image to an internal software buffer. IMAQ Grab Acquire, which you can call multiple times, copies the image currently stored in the internal buffer to a LabVIEW image buffer. The **Immediate?** input to IMAQ Grab Acquire determines if the system returns the image currently being acquired or the last completely acquired image. The default value is FALSE, which causes NI-IMAQ to wait until the current image is completely acquired before returning it. A typical application for an immediate transfer is the acquisition of images of stationary objects. After the program finishes copying images, call IMAQ Close once to shut down the acquisition.

The following figure shows a simplified block diagram for using IMAQ Grab Setup and IMAQ Grab Acquire. In this example, you perform an immediate copy by wiring a TRUE to the **Immediate?** input.

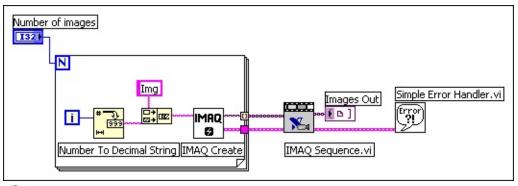


Sequence

A sequence initiates a variable-length and variable-delay transfer to multiple buffers. Use a sequence for applications that process multiple images. You can configure a sequence to acquire every frame or skip a variable number of frames between each image.

Use IMAQ Sequence for sequence applications. IMAQ Sequence starts, acquires, and releases a sequence acquisition. The input **Skip Table** is an array containing the number of frames to skip between images. IMAQ Sequence does not return until the entire sequence is acquired.

The following figure shows a simplified block diagram for using IMAQ Sequence. Place IMAQ Create inside a For Loop to create an array of images for the **Images in** input to IMAQ Sequence. **To Decimal** and **Concatenate** create a unique name for each image in the array.



Note Each image must have a unique name.

Ring

A ring initiates a continuous high-speed acquisition to multiple buffers. Use a ring 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 find an example of a ring acquisition in the <LabVIEW>\examples\IMAQ\IMAQ Low Level.llb file, where <LabVIEW> is the location to which LabVIEW is installed.

You can configure a ring to acquire every frame or to skip a fixed number of frames between acquisitions. In LabVIEW, you must use the NI-IMAQ low-level VIs to perform a ring.

Common NI-IMAQ VI Parameters

IMAQ Session is a unique identifier that specifies the interface file used for the acquisition. This identifier is produced by the IMAQ Init VI and used as an input to all other NI-IMAQ VIs. The NI-IMAQ VIs use **IMAQ Session Out**, which is identical to **IMAQ Session**, to simplify dataflow programming. **IMAQ Session Out** is similar to the duplicate file sessions provided by the file I/O VIs. The high-level acquisition VIs—IMAQ Snap, IMAQ Grab Setup, and IMAQ Sequence—require you to wire **IMAQ Session In** only if you are using an interface other than the default img0, if you are using multiple devices, or if you need to set IMAQ properties before the acquisition.

Many acquisition VIs require that you supply an image buffer to receive the captured image. You can create this image buffer with the IMAQ Create VI. Refer to <u>Managing Buffers in LabVIEW</u> for more information. The input that receives the image buffer is **Image in**. The **Image out** output returns the captured image.

During development, it may be useful to examine the contents of your image at run-time. With LabVIEW 7.0 or later, you can use a LabVIEW image probe to view the contents of your image during execution. Right-click your image wire and select **Probe**.

The acquisition VIs use the **Region of Interest** input to specify a rectangular portion of an image frame to be captured. You can use **Region of Interest** to reduce the size of the image you want to capture. **Region of Interest** is an array of four elements with the elements defined as Left, Top, Right, Bottom. If **Region of Interest** is not wired, the entire image acquisition window is captured. You configure the default acquisition window using MAX.

The acquisition VIs use the **Step x** and **Step y** inputs to specify a horizontal and vertical sampling step. The sampling step causes a reduction in spatial resolution.

Managing Buffers in LabVIEW

IMAQ Create and IMAQ Dispose manage image buffers in LabVIEW. IMAQ Create allocates an image buffer. The following diagram illustrates the IMAQ Create VI.

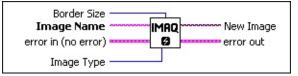


Image Name is a label for the buffer created. Each buffer must have a unique name. **Image Type** specifies the type of image being created. Use **8 bits** for 8-bit monochrome images, **16 bits** for 10-, 12-, and 14-bit monochrome images, **RGB** for RGB color images, and **HSL** for HSL color images. If you do not know the image type at design time, you can get the image type programmatically from the session with the LabVIEW property node.

When you acquire into a buffer, the image type of the buffer is coerced to match the type of the acquired image.

New Image contains information about the buffer, which is initially empty. When you wire **New Image** to the **Image in** input of an image acquisition VI, the image acquisition VI allocates the correct amount of memory for the acquisition. If you are going to process the image, you might need to wire to **Border Size**. **Border Size** is the width, in pixels, created around an image. Some image processing functions, such as labeling and morphology, require a border.

IMAQ Dispose frees the memory allocated for the image buffer. Call this VI only after the image is no longer required for processing. The following diagram illustrates the IMAQ Dispose VI.



Acquisition VIs

Two acquisition VI types are available in LabVIEW—high-level and low-level.

High-Level

You can use the high-level acquisition VIs for basic image acquisition applications. NI-IMAQ includes VIs for snap, grab, and sequence as described in <u>NI-IMAQ Acquisition Types</u>. You can find examples of using the high-level acquisition VIs in the <LabVIEW>\examples\IMAQ\IMAQ High Level.llb file, where <LabVIEW> is the location to which LabVIEW is installed.

Low-Level

Use the low-level acquisition VIs for more advanced image acquisition applications, including ring acquisitions and acquisitions to onboard memory. The low-level VIs configure an acquisition, start an acquisition, retrieve the acquired images, and stop an acquisition. You can use these VIs in conjunction with the event VIs to construct advanced image acquisition applications.

Complete the following steps to perform a low-level acquisition:

- 1. Call IMAQ Init to initialize the device and create an **IMAQ Session**.
- 2. Configure the acquisition with IMAQ Configure List and IMAQ Configure Buffer. IMAQ Configure List configures a buffer list to be used in an acquisition. The buffer list contains a specific number of buffers that contain the acquired images. The buffers can be stored either in system memory or in onboard memory for devices with onboard memory.
- 3. Call IMAQ Configure Buffer once for each buffer in the buffer list. The buffer contains the channel or port from which to acquire and the number of frames to skip before acquiring into the buffer.
- 4. After configuring the buffer list and individual buffers, call IMAQ Start to start the acquisition asynchronously. IMAQ Start returns immediately after the acquisition has started.
- 5. Access the acquired images using either IMAQ Get Buffer or IMAQ Extract Buffer. IMAQ Get Buffer returns acquired images from the buffer list and is normally used for snap and sequence acquisitions. IMAQ Get Buffer waits until the requested buffer has been acquired to return the image. You also can use this VI to return all images in the buffer list. IMAQ Get Buffer can retrieve images from a continuous acquisition only if the acquisition has been stopped.

IMAQ Extract Buffer extracts a buffer from a continuous acquisition and allows for the examination of a buffer during acquisition. This VI removes the buffer from the acquisition. NI-IMAQ does not write new data into the buffer until this VI is called again. Use IMAQ Extract Buffer in ring acquisitions when you must process images during the acquisition. IMAQ Copy Acquired Buffer returns a copy of an acquired image. IMAQ Copy Acquired Buffer allows you to create a copy of any buffer at any time during the acquisition.

6. After an acquisition, release the resources associated with the acquisition using IMAQ Close. IMAQ Close also stops the acquisition if one is in progress. If you want to stop the acquisition without releasing the resources (such as the image buffers), use IMAQ Stop.

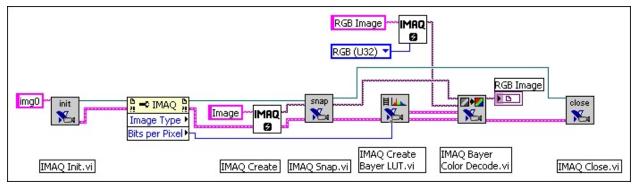
Examples of the low-level acquisition VIs are included in the <LabVIEW>\examples\IMAQ\IMAQ Low Level.llb file, where <LabVIEW> is the location to which LabVIEW is installed.

Bayer Decoded Acquisition

Complete the following steps to perform a Bayer decoded acquisition:

- 1. Call IMAQ Init to generate an **IMAQ Session**.
- 2. Use the IMAQ property node to find the bits per pixel.
- 3. Create an image with IMAQ Create using the image type from the property node.
- 4. Acquire the image with IMAQ Snap.
- 5. Call IMAQ Create Bayer LUT to create a lookup table based on the input gains.
 - Note To find the appropriate values for the gains, use the White Balancing Utility located at Start»All Programs»National Instruments»Vision»White Balancing Utility.
- 6. Create an RGB image using IMAQ Create.
- 7. Call IMAQ Bayer Color Decode to decode the color information from the raw image.

The following figure shows a block diagram for acquiring an IMAQ Bayer decoded image.



Triggering

You can use trigger lines on the image acquisition device to link or coordinate a vision action or function with events external to the computer, such as receiving a strobe pulse for lighting or a pulse from an infrared detector that indicates the position of an item on an assembly line. All TTL trigger lines are bidirectional so that the device can generate or receive the triggers on any line. Isolated inputs and outputs are unidirectional. Isolated inputs may only be used to receive triggers. Isolated outputs may only be used to generate triggers. Use Real-Time System Integration (RTSI) triggers to coordinate your image acquisition device with other National Instruments devices, such as data acquisition (DAQ) devices.

Use IMAQ Configure Trigger2 to configure the trigger conditions for an acquisition. You must call IMAQ Configure Trigger2 before the acquisition VI. The **Trigger Type** input specifies the type of trigger signal. Each trigger line has a programmable polarity that is specified with **Trigger Polarity**. **Frame timeout** specifies the amount of time to wait for the trigger.

The following diagram shows how to use IMAQ Configure Trigger2 to perform a snap acquisition based on a trigger.

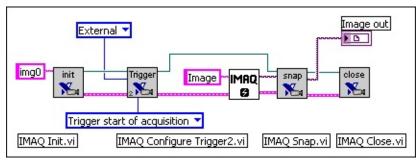
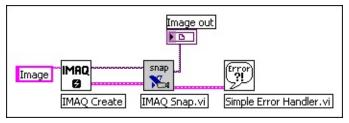


Image Display

Many image acquisition applications require that one or more images be displayed. You have three options for displaying images in LabVIEW.

If you have LabVIEW 7.0 or later, you display an image directly on the front panel using the Image Display control, which is available on the **Vision Controls** palette. To display an image on the Image Display control, place the image control on the front panel of your VI. On the block diagram, wire the **Image Out** from an acquisition VI to the Image Display control terminal. The following diagram illustrates using an Image Display control to display an image. To view examples using the Image Display control in LabVIEW 7.0 and later, select **Help*Find Examples**.



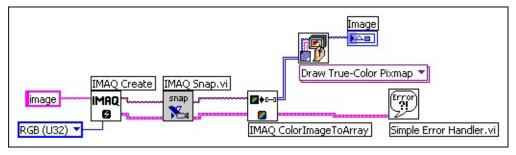
If you have NI Vision 7 for LabVIEW, the image processing and analysis software for LabVIEW, you can display an image in an external window using the External Display VIs on the **External Display** palette. IMAQ WindDraw, located at **Vision Utilities»Display**, displays an image in a separate image window. The following figure illustrates using IMAQ WindDraw to display an image acquired using IMAQ Snap. You can display images in the same way using any acquisition type. Refer to the *NI Vision Concepts Manual* for more information about the display capabilities of NI Vision.



If you do not have either LabVIEW 7.0 or later or NI Vision, you can display an image with a LabVIEW picture control.

To display an image on a picture control, place the picture control on the front panel of the VI. Use either the IMAQ ImageToArray VI or the IMAQ ColorImageToArray VI to copy an image from an image buffer into a LabVIEW array. You can wire this array to the Draw True-Color Pixmap VI. Wire the new image output from Draw True-Color Pixmap to the

picture control indicator. Refer to the *LabVIEW VI, Function, & How To Help* for more information about the picture control. The following figure illustrates using a picture control to display an RGB image acquired with IMAQ Snap.



Camera Attributes

Camera attribute VIs allow you to control camera-specific functions, such as integration time and pixel binning, directly from LabVIEW. You can set them from MAX. Information about specific attributes for some cameras is contained in a camera attribute file <my camera>.txt, which is in the <NI-IMAQ>/Camera Info directory, where <NI-IMAQ> is the location to which NI-IMAQ is installed.

Use the IMAQ Set Camera Attribute VI to set the value of a camera attribute. The camera attribute file lists all attributes for the camera and each attribute description contains four fields: **Attribute Name**, **Description**, **Data Type**, and **Possible Values**.

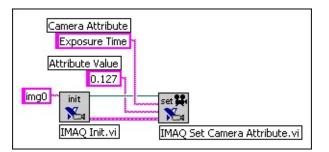
Note Camera attribute files are not available for every camera.

The **Attribute Name** field contains the name of the attribute in quotes. Wire this field to the **Camera Attribute** input on IMAQ Set Camera Attribute VI. The **Data Type** field contains the data type of the attribute which can either be **String**, **Integer**, or **Float**. **String** indicates that there is a list of possible values which are listed in **Possible Values** in quotes. To set the value of a string attribute, wire the appropriate string value to **Attribute Value** on IMAQ Set Camera Attribute.

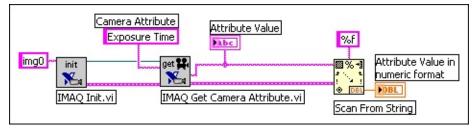


Note The spelling and syntax of the Attribute Name and string values must match the camera attribute file exactly.

A data type of **Integer** indicates that NI-IMAQ converts the string wired to **Attribute Value** to an integer. **Float** indicates that NI-IMAQ converts the string wired to **Attribute Value** to a floating point number. The valid numeric values for integer and float data types are listed in **Possible Values**. Use **Format into String**, located on the **String** subpalette, to convert numerics into strings for use with IMAQ Set Camera Attribute. The following figure shows how to use IMAQ Set Camera Attribute to set the value of a float camera attribute.

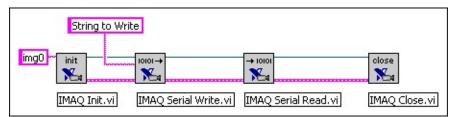


Use the IMAQ Get Camera Attribute VI to get the value of a camera attribute. Use MAX to find information about the attributes for your camera. All camera attributes are returned in string format. If the data type of the attribute is integer or float, use the **Scan from String** function, located on the **String** subpalette, to convert the string into a numeric. The following figure shows how to use IMAQ Get Camera Attribute with **Scan from String** to get the value of a float camera attribute.



Many cameras are configured with serial commands. Camera files provided by National Instruments are already programmed with the appropriate camera serial command set. All serial commands set in the camera file are automatically sent to the camera when IMAQ Init is called. If you need more low-level control over the serial communication between the camera and your IMAQ device, use the IMAQ Serial Read and IMAQ Serial Write VIs.

The following figure illustrates how to use the IMAQ Serial VIs.



Error Handling

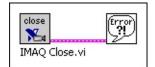
Every NI-IMAQ VI contains an **error in** input cluster and an **error out** output cluster. The following diagram illustrates error clusters.

error in (no error)	error out
status code noerror ≜0	status code
source	source

The clusters contain a Boolean value that indicates if an error occurred, the code for the error, and the source or the name of the VI that returned the error. If **error in** indicates an error, the VI passes the error information to **error out** and does not execute any NI-IMAQ function.

You can use **Functions»Time&Dialog Palette»Simple Error Handler** to check for errors that occur while executing a VI. If you wire an error cluster to the Simple Error Handler VI, the VI deciphers the error information and displays a dialog box that describes the error. If no error occurred, the Simple Error Handler VI does nothing.

The following figure shows how to wire an NI-IMAQ VI to the Simple Error Handler VI.



NI-IMAQ and the LabVIEW Real-Time Module

With NI-IMAQ and the LabVIEW Real-Time Module, you have all the tools necessary to develop a complete image acquisition application on a reliable, embedded platform. NI-IMAQ for the LabVIEW Real-Time Module combines the image acquisition capabilities of Vision hardware and NI-IMAQ software with the real-time programming and execution capabilities of the LabVIEW Real-Time Module.

Develop your image acquisition application with NI-IMAQ for the LabVIEW Real-Time Module. Then download your code to run on a realtime, embedded target. You also can add National Instruments DAQ, Motion Control, CAN, and serial instruments to the system to create a complete, integrated, embedded system.

The following sections describe how you can use NI-IMAQ to create an image acquisition application for a real-time, embedded target with the LabVIEW Real-Time Module.

System Components and Requirements

Displaying Images with NI-IMAQ and the LabVIEW Real-Time Module Troubleshooting NI-IMAQ for the LabVIEW Real-Time Module

System Components and Requirements

Using NI-IMAQ with the LabVIEW Real-Time Module consists of a development system and one or more deployed real-time targets.

Development System

The development system is made up of two components—a Pentiumbased host machine using a Windows operating system and a National Instruments PXI chassis housing a PXI controller. Use the host machine to configure the PXI controller as a real-time target and to develop the application. Execute the application remotely on the PXI controller. The two machines communicate with each other over a network connection and use MAX to share configuration settings and software.



Note You must have a network connection during development to configure your real-time target and download software and code from your host machine. This network connection is optional at runtime.

Pentium-Based Host Computer

The host machine must meet the following minimum system requirements:

- Windows Vista/XP/2000
- Pentium 4 1 GHz processor
- 1024 × 768 resolution video adapter using 16-bit color
- 512 MB RAM
- 2 GB of free hard disk space
- LabVIEW 7.1.1 or later and the LabVIEW Real-Time Module 7.1.1 or later

National Instruments PXI System

Select the PXI controller that best meets the needs of the application.

The following table lists the different controllers you can use with NI-IMAQ and the LabVIEW Real-Time Module.

Device	Functionality
NI 8184, NI 8185, and NI 8186	Supports full functionality
NI 8175 and NI 8176	Supports full functionality
NI 8106, NI 8170, NI 8156B, and NI 8140 RT Series	Does not support real-time Video Out

The PXI system also must meet the following minimum system

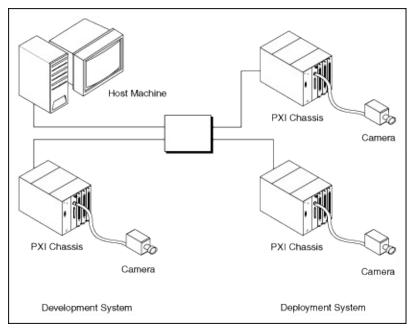
requirements:

- Network adapter
- National Instruments PXI image acquisition device
- Analog or digital camera and associated cables

Deployed System

After you have configured the host development system, you can set up and configure additional LabVIEW Real-Time Module targets for deployment. These deployed systems use the same hardware and software as the development LabVIEW Real-Time Module target.

The deployed LabVIEW Real-Time Module devices can be connected to the same subnet as the host development system or connected to a different subnet as required by your application. These connections are illustrated in the following figure.



Displaying Images with NI-IMAQ and the LabVIEW Real-Time Module

NI-IMAQ and the LabVIEW Real-Time Module give you two options for displaying images—Remote Display and RT Video Out. Choose the display option that best fits the needs of the application.

Remote Display

Remote Display is available in LabVIEW 7.0 or later. Use Remote Display during development and debugging to view the LabVIEW front panel Image Display control from the host machine. Use RT Video Out to display images on a monitor connected to the remote LabVIEW Real-Time Module system.

When you use Remote Display, the real-time target machine must transfer the image back to host machine across the network. Use the IMAQ Remote Compression VI, located on the **Vision Utilities»IMAQ RT** palette, to minimize network bandwidth consumed for remote display.

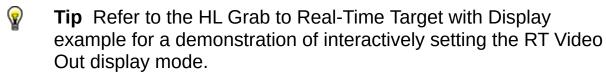
RT Video Out

RT Video Out allows you to display images on a monitor that is connected to the PXI controller. To access this feature, use the IMAQ RT Video Out VI from the **Vision Utilities**»**IMAQ RT** palette.



Note This feature is available only on PXI controllers that feature the i815 chipset or the i845 chipset, such as the National Instruments PXI-8175/76 Series controllers or the PXI-8184/85/86 Series controllers.

To programmatically configure the system to use IMAQ RT Video Out to display system images, use the IMAQ Video Out Display Mode VI. This VI allows you to set parameters for screen area, color depth, and refresh frequency.



Troubleshooting NI-IMAQ and the LabVIEW Real-Time Module

This section describes solutions and suggestions for common errors in NI-IMAQ for the LabVIEW Real-Time Module.

PXI Controller Errors

Why cant MAX find my PXI controller?

Try the following techniques if your PXI controller does not appear in MAX:

- When configuring the PXI controller, ensure that the controller is located on the same subnet of the network as the host PC. If you are unsure of your network configuration, consult your network administrator for assistance.
- If you do not have a keyboard connected to the PXI controller, check the BIOS settings of the controller. The Halt On setting must be set to All, But Keyboard for the PXI controller to boot without a keyboard connected. You can find the Halt On setting in the Standard CMOS Setup options. Refer to the PXI controller user manual for more information about BIOS settings.

Remote Display Errors

Why is my application slow when I use Remote Display?

Remote Display transfers images across the network from the Real-Time target machine to the host machine. Images are large, and this network transfer can be slow. You can use the IMAQ Remote Compression VI, located on the **Vision Utilities»IMAQ RT** palette, to compress the image before transferring it over the network. Compressing the image improves performance but may impact image quality.

Why does my remotely displayed image have low quality?

Try the following to improve your image quality:

- Ensure that the camera aperture is open to allow the appropriate amount of light for an acquisition.
- Check the compression settings.
- Verify that the display settings in Windows are set to use at least 24-bit color.

RT Video Out Errors

Why do I have an invalid Video Out Mode?

To use the RT Video Out functionality in NI-IMAQ for the LabVIEW Real-Time Module, you must have a PXI controller that supports this feature.

Refer to the following table for a list of controllers that supports the RT Video Out functionality.

Device	Functionality
NI 8184, NI 8185, and NI 8186	Supports full functionality
NI 8175 and NI 8176	Supports full functionality
NI 8106, NI 8170, NI 8156B, and NI 8140 RT Series	Does not support RT Video Out
RT Desktop	Support based on graphics card in RT Desktop

If you are using a controller that does not support RT Video Out, consider using Remote Display to display the images.

Why cant I see my images when I use RT Video Out?

Use the IMAQ Video Out VI to configure the video mode before you attempt to display the images. This VI allows you to set the refresh frequency, screen area, and color depth.

 $\overline{\mathbb{N}}$

Note If you are using a monitor that does not support high refresh frequencies, the images do not display correctly. Refer to the monitor documentation for information about supported refresh frequencies.

Programming with ActiveX controls

This topic discusses using National Instruments NI Vision for Visual Basic software with your image acquisition device, including the organization of NI Vision for Visual Basic and using the CWIMAQ control.

Programming with ActiveX controls allows you to easily configure and perform image acquisition tasks using NI Vision for Visual Basic and the CWIMAQ control.

Documentation and Examples

This topic assumes that you are familiar with Visual Basic and can use ActiveX controls in Visual Basic. The following are good sources of information about Visual Basic and ActiveX controls:

- Microsoft Developer Network (MSDN) at www.msdn.microsoft.com
- Documentation that accompanies Microsoft Visual Studio

Several additional documentation resources are available to help you create your vision application:

- *NI Vision Concepts Manual*—If you are new to machine vision and imaging, read this manual to understand the concepts behind NI Vision.
- *NI Vision for Visual Basic Reference Help*—If you need information about individual methods, properties, or objects, refer to this help file.
- **Example programs**—If you want examples of how to create specific applications, go to the <Vision>\Examples\MSVB directory or the <Vision>\Examples\MSVB.NET directory, where <Vision> is the location to which you installed Vision.
- **CWMachineVision source code**—If you want to see the source code for the CWMachineVision control, go to the <Vision>\Source\MSVB\MachineVision directory.
- **Application Notes**—If you want to know more about advanced NI Vision concepts and applications, refer to the Application Notes located on the National Instruments Web site at ni.com/appnotes.nsf.
- NI Developer Zone (NIDZ)—If you want even more information about developing a vision application, visit the <u>NI Developer Zone</u> at ni.com/zone. The NI Developer Zone contains example programs, tutorials, technical presentations, the Instrument Driver Network, a measurement glossary, an online magazine, a product advisor, and a community area where you can share ideas, questions, and source code with vision developers around the world.

NI Vision for Visual Basic

NI Vision for Visual Basic consists of four ActiveX controls contained in two files: cwimaq.ocx and cwmv.ocx.

cwimaq.ocx

cwimaq.ocx contains three ActiveX controls and a collection of ActiveX objects. The ActiveX controls are the CWIMAQ control, the CWIMAQVision control, and the CWIMAQViewer control. Refer to <u>ActiveX Objects</u> for more information about the ActiveX objects.

CWIMAQ Control

Use this control to configure and perform an acquisition from an image acquisition device. The CWIMAQ control has property pages that allow you to modify various properties to configure your acquisition and gather information about your image acquisition device. All of the functionality available from the property pages during design time is also available through the properties of the CWIMAQ control during run-time. The control has methods that allow you to perform and control acquisitions, as well.



Note You must have the NI-IMAQ device driver installed on the target system for the CWIMAQ control to function.

CWIMAQVision Control

Use this control to analyze and process images and their related data. The CWIMAQVision control provides methods for reading and writing images to and from files, analyzing images, and performing a variety of image processing algorithms on images.

Note This control is available only if you have NI Vision for Visual Basic installed. Contact your National Instruments sales representative, or visit the <u>National Instruments Web site</u> at ni.com to purchase NI Vision for Visual Basic.

CWIMAQViewer Control

Use this control to display images and provide the interface through which the user interacts with the displayed image. This control includes the ability to zoom and pan images and to draw regions of interest (ROIs) on an image. The CWIMAQViewer control has property pages that allow you to configure the appearance and behavior of the viewer during design time as well as properties that you can configure during run-time. The control has methods that allow you to attach images to and detach images from the viewer for display purposes.



Note The CWIMAQViewer control is referred to as a viewer in the remainder of this document.

cwmv.ocx

cwmv.ocx contains one ActiveX control and a collection of ActiveX objects. Refer to <u>ActiveX Objects</u> for more information about ActiveX objects.

CWMachineVision Control

Use this control to perform high-level machine vision tasks, such as measuring distances. This control is written entirely in Visual Basic using the methods on the CWIMAQVision and CWIMAQViewer controls. The source code for the CWMachineVision control is included in the product.

Note This control is available only if you have NI Vision for Visual Basic installed. Contact your National Instruments sales representative, or visit the <u>National Instruments Web site</u> at ni.com to purchase NI Vision for Visual Basic.

ActiveX Objects

ActiveX objects are classified as input and output objects. The objects are grouped according to input parameters and output parameters that are used by particular methods, which reduces the number of parameters that you must pass to those methods.



Note ActiveX objects in cwimaq.ocx have a CWIMAQ prefix, and objects in cwmv.ocx have a CWMV prefix.

You must create an ActiveX object before you can use it. You can use the New keyword in Visual Basic to create these objects. For example, use the following syntax to create and store an image in a variable named image:

Dim image as New CWIMAQImage

Tip If you intend to develop your application in Visual C++, National Instruments recommends that you use NI Vision for LabWindows/CVI. However, if you decide to use NI Vision for Visual Basic to develop applications for Visual C++, you can create objects using the respective Create methods on the CWIMAQVision control or CWMachineVision control. For example, to create a CWIMAQImage object, use the CWIMAQVision.CreateCWIMAQImage method.

Buffer Management

The CWIMAQ control uses the CWIMAQImage object in a CWIMAQImage collection as the image buffer when performing an acquisition. You can access this collection using the Images property on the CWIMAQ control. Set the count property of the collection to the number of buffers that your application requires.

You also can set the number of buffers during design time by setting the value of **Image Count** on the **Acquisition** property page for the CWIMAQ control. When you start the acquisition, the CWIMAQ control automatically manages the image type and the size of the images in the collection.

Acquire an Image

Use the CWIMAQ control to acquire images with a National Instruments image acquisition device. You can use NI Vision for Visual Basic to perform one-shot and continuous acquisitions. You can choose the acquisition type during design time by setting the value of the **Acquisition Type** combo box to **One-Shot** or **Continuous**. The **Acquisition Type** combo box is located on the **Acquisition** property page of the CWIMAQ control. You can set the value at run-time by setting the CWIMAQ.AcquisitionType property to cwimaqAcquisitionOneShot or cwimaqAcquisitionContinuous.

One-Shot Acquisition

Use a one-shot acquisition to start an acquisition, perform the acquisition, and stop the acquisition using a single method. The number of frames acquired is equal to the number of images in the images collection. Use the CWIMAQ.AcquireImage method to perform this operation synchronously. Use the CWIMAQ.Start method to perform this operation asynchronously.

If you want to acquire a single field or frame into a buffer, set the image count to 1. This operation is also referred to as a snap. Use a snap for low-speed or single capture applications. The following code illustrates a synchronous snap:

Private Sub Start_Click()

```
CWIMAQ1.AcquisitionType = cwimaqAcquisitionOneShot
```

CWIMAQ1.AcquireImage

End Sub

If you want to acquire multiple frames, set the image count to the number of frames you want to acquire. This operation is called a sequence. Use a sequence for applications that process multiple images. The following code illustrates an asynchronous sequence when numberOfImages is the number of images that you want to process:

Private Sub Start_Click()

```
CWIMAQ1.AcquisitionType = cwimaqAcquisitionOneShot
```

CWIMAQ1.Images.RemoveAll

CWIMAQ1.Images.Add numberOfImages

CWIMAQ1.Start

End Sub

Continuous Acquisition

Use a continuous acquisition to start an acquisition and continuously acquire frames into the image buffers, and then explicitly stop the acquisition. Use the CWIMAQ.Start method to start the acquisition. Use the CWIMAQ.Stop method to stop the acquisition. If you use a single buffer for the acquisition, this operation is called a grab. The following code illustrates a grab:

Private Sub Start_Click()

```
CWIMAQ1.AcquisitionType = cwimaqAcquisitionContinuous
```

CWIMAQ1.Start

End Sub

Private Sub Stop_Click()

CWIMAQ1.Stop

End Sub

A ring operation uses multiple buffers for the acquisition. Use a ring for high-speed applications that require processing on every image. The following code illustrates a ring, where numberOfImages is the number of images that you want to process:

```
Private Sub Start_Click()
```

```
CWIMAQ1.AcquisitionType = cwimaqAcquisitionContinuous
```

```
CWIMAQ1.Images.RemoveAll
```

CWIMAQ1.Images.Add numberOfImages

CWIMAQ1.Start

End Sub

Private Sub Stop_Click()

CWIMAQ1.Stop

End Sub

Triggering

You may need to coordinate a vision event with events that are occurring outside the computer, such as receiving a strobe pulse for lighting or an infrared detector pulse indicating the position of an item on an assembly line. Any TTL-level signal can serve as a trigger for image acquisition devices. All of the lines are fully bidirectional, allowing your image acquisition device to generate or receive the triggers on any line. Use Real-Time System Integration (RTSI) triggers to coordinate the image acquisition device with other National Instruments devices, such as data acquisition (DAQ) devices.

You can configure triggers during both design time and run-time. At design time, use the Signal I/O property page of the CWIMAQ control to configure the triggers. At run-time, use the CWIMAQSignals object, which you can access through CWIMAQ.Signals to configure the triggers.

Each CWIMAQSignal object that is added to the CWIMAQSignals collection corresponds to a trigger line that you want to configure. The CWIMAQSignal.Line property specifies which external or RTSI trigger receives the incoming trigger signal. Each trigger line has a programmable polarity that is specified with CWIMAQSignal.Polarity. Use the CWIMAQ.FrameTimeout property to specify the amount of time to wait for a trigger. When you have configured the triggers, you can perform any acquisition described above. The following code illustrates the use of a trigger to control your acquisition:

Private Sub Start_Click()

'Capture an image on a trigger on RTSI line 0

CWIMAQ1.Signals.Add.Initialize cwimaqRTSI,_ cwimaqCaptureStart,

cwimaqActiveHigh, 0

End Sub

Display an Image

Use the CWIMAQViewer control to display an image. Use the CWIMAQViewer.Attach method to attach the image you want the viewer to display. When you attach an image to a viewer, the image automatically updates the viewer whenever an operation modifies the contents of the image. You can access the image attached to the viewer using the CWIMAQViewer.Image property. Before you attach an image to the viewer, the viewer already has an image attached by default. Therefore, the viewer has an image attached to it at all times. You can use the attached image as either a source image, destination image, or both using the Image property.

Use the following code to attach an image to the viewer:

```
Private Sub Start_Click()
```

CWIMAQ1.AcquisitionType = cwimaqAcquisitionContinuous

```
CWIMAQ1.Images.RemoveAll
```

CWIMAQ1.Images.Add 1

CWIMAQViewer1.Attach CWIMAQ1.Images(1)

CWIMAQ1.Start

End Sub

```
Private Sub Stop_Click()
```

```
CWIMAQ1.Stop
```

End Sub

You can use the CWIMAQViewer.Palette property to access the CWIMAQPalette object associated with the viewer. Use the CWIMAQPalette object to programmatically apply a color palette to the viewer. You can set the CWIMAQPalette.Type property to apply predefined color palettes. For example, if you need to display a binary image—an image containing particle regions with pixel values of 1 and a background region with pixel values of 0—set the Type property to cwimaqPaletteBinary. Refer to Chapter 2, *Display*, of the *NI Vision Concepts Manual* for more information about color palettes.

You also can set a default palette during design time using the **Menu** property page for the CWIMAQViewer control. You also can change the

color palette during run-time by using the context menu on the viewer.

Camera Attributes

Camera attributes allow you to control camera-specific functions, such as integration time and pixel binning. You can access the camera attributes using CWIMAQ.CameraAttribute. You also can set them in MAX. Information about specific attributes for some cameras is contained in a camera attribute file <my camera>.txt, which is in the <NI-IMAQ>/Camera Info directory, where <NI-IMAQ> is the location to which NI-IMAQ is installed.

Note Camera attribute files are not available for every camera.

The camera attribute file lists all attributes for the camera where 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 quotation marks. Pass the value of this field to the Attribute parameter of the CameraAttribute property.

The **Data Type** field contains the data type of the attribute, which can be **String**, **Integer**, or **Float**. **String** indicates that possible values are listed in quotes in the **Possible Values** section. To set the value of a **String** attribute, set the appropriate string value for the CameraAttribute.

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

Error Handling

Errors generated by the CWIMAQ control have an Error Context and a Status Code associated with them. The CWIMAQErrorContext constants specify the Error Contexts. The Status Code for an error is a negative value that corresponds to the actual error that occurred.

There are three ways that errors are reported by the CWIMAQ control:

- Return value of the methods
- Exceptions
- IMAQError event

The CWIMAQ.ExceptionOnError property specifies if the control throws an exception when an error occurs. If ExceptionOnError is set to **False**, no exception is generated. If the error was generated as a result of a method call, the method returns the status code. If ExceptionOnError is set to **True**, the CWIMAQ control throws an exception. If you choose to catch this exception, you can use the Err object in Visual Basic to get information about the exception that the control generated.

The CWIMAQ control generates an IMAQError event if there is an error. To select the contexts for which CWIMAQ generates error events, add the appropriate CWIMAQErrorContexts constants together and assign the sum of the constants to CWIMAQ.ErrorEventMask. The IMAQError event provides information about the status code and the context in which the error occurred

You can handle errors using one of the following methods:

- Set ExceptionOnError to True, and do not provide an exceptionhandling mechanism. This directs your application to generate a run-time error and display a run-time error dialog.
- Set ExceptionOnError to True, and provide an exception-handling mechanism. You can then direct how your application handles the exception.
- Set ExceptionOnError to False to check the value of the return code. This method requires you to check the return values every time you make a method call and handle the return values appropriately.
- Use the IMAQError event handler in conjunction with setting the value of ExceptionOnError to do what is appropriate for your

application. This method allows you to handle errors according to the context. Also, this method allows you to identify errors that might occur during an asynchronous acquisition to handle them appropriately.

Warnings

Warnings are different from errors in the following respects:

- They have a positive Status Code.
- They do not generate exceptions, even if ExceptionOnError is **True**.
- They generate an IMAQWarning event instead of an IMAQError event.

You can handle warnings by checking the return value of the methods or by providing an IMAQWarning event handler.

Variable Height Acquisition (VHA)

In some line scan applications, you may not know the exact size of the object or objects you are imaging. One example is a conveyer belt application that acquires continuous line-scan images of objects with multiple sizes. For such applications, you can use a mode of triggered acquisition that allows you to acquire a variable number of lines in a multiple buffered application.

In this mode, you supply the image acquisition device a trigger which asserts when you want to begin capture. The image acquisition device continues to acquire lines until the trigger is unasserted. When the acquisition is complete, the driver returns the number of lines acquired. By using the variable line mode, you acquire only the amount of data that you need. This technology greatly enhances performance by minimizing the total amount of data to process.

Setup for Acquiring a Variable Number of Lines

To set up your system for acquiring a variable number of lines, first determine the size of the largest possible object under inspection. The NI-IMAQ driver software defines the dimensions of an image as width and height. Width is the number of pixels per line and height is the number of lines in the image. When you determine the maximum possible object size, in lines, for your application, enter this number for the height parameter in MAX, as follows:

- 1. Launch MAX.
- Click Devices and Interfaces»NI-IMAQ Devices»PCI-14xx»XXX, where XXX represents the camera you want to use. Click the Acquisition Parameters tab.
- 3. Enter the maximum possible number of lines in the Acquisition Window **Height** control.

NI-IMAQ uses this line number information to allocate each image into a buffer. This allocation is done prior to the acquisition to ensure that memory is available on the system before the acquisition begins.

Setting Up the Trigger

The VHA trigger can come in any unused external trigger line or RTSI line. The trigger can be either High True or Low True. Acquisition of a buffer begins on the assertion of this trigger line and terminates when the line is deasserted or the number of lines equals the **Height** parameter set in MAX.

Enabling VHA in Your Code

The actual implementation varies depending on the development application you use, but the concept is the same. Complete the following general steps:

- 1. Enable the Variable Height Acquisition attribute or property.
- 2. Configure the buffers.
- 3. Set up the trigger to **Trigger each buffer**.
- 4. Begin the acquisition.
- 5. Check the actual height for each buffer during the acquisition.

Examples

If you install the IMAQ examples for your compiler, you can find a readyto-run example showing how to acquire a variable number of lines with a line scan camera.

- In LabVIEW, refer to LL VHA Ring.vi, which is located in the <LabVIEW>\examples\IMAQ\IMAQ Signal IO.llb file, where LabVIEW is location to which <LabVIEW> is installed.
- In LabWindows/CVI, refer to VHA Ring.prj, which is located in the <CVI>\samples\IMAQ\Signal IO folder.
- In Microsoft C++ version 6, refer to the <NI-IMAQ>\Sample\MSVC\Signal IO\VHA Ring folder.

Integration with DAQ and Motion Control

Platforms that support NI-IMAQ also support NI-DAQ and a variety of National Instruments data acquisition (DAQ) devices. This allows integration between image acquisition devices and DAQ products.

With National Instruments image acquisition hardware and NI Vision pattern matching software, you can quickly and accurately locate objects in instances where objects vary in size, orientation, focus, and even when the part is poorly illuminated. Use National Instruments high-performance stepper and servo motion control products with pattern matching software in inspection and guidance applications, such as locating alignment markers on semiconductor wafers, guiding robotic arms, inspecting the quality of manufactured parts, and locating cells.



Α

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

- active line The region of lines actively being stored. Defined by a line start (relative to the vertical synchronization signal) and a line count.
- activeThe region of pixels actively being stored. Defined by a pixelpixelstart (relative to the horizontal synchronization signal) and aregionpixel count.
- API Application programming interface.
- area A rectangular portion of an acquisition window or frame that is controlled and defined by software.

В

base configuration	A configuration defined by the Camera Link specification that uses one cable between the camera and the image acquisition device. This configuration supports combinations of camera bit depths and number of taps that allow the data to be transmitted and received by a single 28-signal interface chip.
black reference level	The level that represents the darkest value an image can have.
buffer	Temporary storage for acquired data.
bus	A 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.

С

Camera Interface standard for digital video data and camera control Link based on the Channel Link chipset.

Channel The physical interface chip on which Camera Link is based. Link Accepts 28 signals and serializes the data and enable signals chipset at a 7:1 ratio for transmission across the Camera Link cable.

D

- DAQ Data acquisition. (1) Collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing. (2) Collecting and measuring the same kinds of electrical signals with A/D or DIO devices plugged into a computer, and possibly generating control signals with D/A and/or DIO devices in the same computer.
- DIO Digital input/output.
- 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.
- downplugging a larger device into a smaller connector. An example plugging of down-plugging is plugging a 64-bit PCI device into a 32-bit connector. Down-plugging is not allowed by the PCI Express specification; a larger link device will not mechanically fit into a smaller link connector.
- drivers Software that controls a specific hardware device, such as an image acquisition device.

F

full A configuration defined by the Camera Link specification configuration that uses two cables between the camera and the image acquisition device. This configuration supports combinations of camera bit depths and number of taps that allow the data to be transmitted and received by three 28-signal interface chips.

G

gray level The brightness of a pixel in an image.

Н

handshaking A type of protocol that makes it possible for two devices to synchronize operations.

- HSL A color encoding scheme using hue, saturation, and luminance information where each image in the pixel is encoded using 32 bits: 8 bits for hue, 8 bits for saturation, 8 bits for luminance, and 8 unused bits.
- hue Determination of color, such as red, blue, green, and yellow. White, black, and gray are not considered hues. They are intensities.

I

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.

L

- LabVIEW Laboratory Virtual Instrument Engineering Workbench. Program development environment based on the G programming language. LabVIEW is used commonly for test and measurement applications.
- lane PCI Express Lane. A PCI Express lane contains one differential pair for transmitting data and one differential pair for receiving data.
- link PCI Express Link. A collection of one or more PCI Express lanes. A xN Link is composed on *N* lanes. The number of lanes in the link limits device throughput. A x1 link theoretically provides 250 MB/s in each direction—to and from the device. Once overhead is accounted for, a x1 link can provide approximately 200 MB/s of input capability and 200 MB/s of output capability. Increasing the number of lanes in a link increases maximum throughput by approximately the same factor.
- luminance The brightness or intensity of a color. The monochrome level of a video signal.
- LVDS Low Voltage Differential Signaling (EIA-644).

Μ

MAX Measurement & Automation Explorer. The National Instruments Windows-based graphical configuration utility you can use to configure NI software and hardware, execute system diagnostics, add new channels and interfaces, and view the devices and instruments you have connected to your computer. MAX is installed on the desktop during the National Instruments driver software installation.

medium A configuration defined by the Camera Link specification configuration that uses two cables between the camera and the image acquisition device. This configuration supports combinations of camera bit depths and number of taps that allow the data to be transmitted and received by two 28-signal interface chips. Ν

NI- Driver software for National Instruments image acquisition IMAQ hardware.

Ρ

- parity Method of error checking. Ensures that there is always either an even number or an odd number of asserted bits in a byte, character, or word, according to the logic of the system. If a bit should be lost in data transmission, its loss can be detected by checking the parity
- PCI Peripheral component interconnect. A high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA. PCI offers a theoretical maximum transfer rate of 133 Mbytes/s shared among all devices on the bus for input and output.
- PCIe PCI Express. A high-performance expansion bus architecture originally developed by Intel to replace PCI. PCIe offers a theoretical maximum transfer rate that is dependent upon lane width. A x1 link theoretically provides 250 MB/s in each direction—to and from the device. Once overhead is accounted for, a x1 link can provide approximately 200 MB/s of input capability and 200 MB/s of output capability. Increasing the number of lanes in a link increases maximum throughput by approximately the same factor.
- 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 Divides the incoming horizontal video line into pixels. clock

- PLC Programmable logic controller. (1) A highly reliable specialpurpose computer used in industrial monitoring and control applications. PLCs typically have proprietary programming and networking protocols, and special-purpose digital and analog I/O ports. (2) A device with multiple inputs and outputs that contains a program you can alter.
- protocol The exact sequence of bits, characters, and control codes used to transfer data between computers and peripherals through a communications channel.

Q

quadrature An encoding technique for a rotating device where two encoder tracks of information are placed on the device, with the signals on the tracks offset by 90 degrees from each other. The phase difference indicates the position and direction of rotation.

R

- 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.
- 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, on part in 4,096 resolution, and 0.0244 percent of full scale.
- RGB Red, green, blue—A three-component video signal in which all the colors in a scene or image are conveyed as three primary colors (red, green, and blue) on three separate channels. Sometimes, the green signal also carries the horizontal and vertical synchronization information.
- ROI Region of interest. A hardware-programmable rectangular portion of the acquisition window.
- RTSI bus Real-Time System Integration Bus. The National Instruments timing bus that connects Vision and DAQ devices directly, by means of connectors on the devices, for precise synchronization of functions.

3	
saturation	The amount of color pigment present. The lower the
	saturation, the more white is present in the color.
scatter-	A type of DMA that allows the DMA controller to reconfigure

on-the-fly. gather DMA

S

Т

- tap A set of data lines that deliver one pixel per pixel clock from the camera. Some cameras send multiple streams, or taps, of data over a cable simultaneously to increase transfer rate. Also referred to as channels or simultaneous pixels.
- transfer The rate, measured in bytes/s, at which data is moved from rate source to destination after software initialization and set up operations. The maximum rate at which the hardware can operate.

trigger Any event that causes or starts some form of data capture.

trigger Circuitry that routes, monitors, and drives external and RTSI bus trigger lines. You can configure each of these lines to start or stop acquisition on a rising edge or a falling edge.

circuitry

TTL Transistor-transistor logic. A digital circuit composed of bipolar transistors wired in a certain manner. A typical medium-speed digital technology. Nominal TTL logic levels are 0 and 5 V.

U

upplugging a smaller link device into a larger link connector. For example, for the PCI Express bus, up-plugging is plugging a x1 device into a x4 connector or plugging a x4 device into a x16 connector.

V

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.

W

web Applications that involve acquiring images of objects on a inspection fast-moving conveyor or stage in a production system.

Υ

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

Important Information

Warranty Copyright Trademarks Patents Warning Regarding Use of NI Products

Warranty

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

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

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

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

Copyright

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

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about <u>National Instruments trademarks</u>.

FireWire® is the registered trademark of Apple Computer, Inc.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix® and Tek are registered trademarks of Tektronix, Inc.

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

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help**»**Patents** in your software, the patents.txt file on your CD, or <u>ni.com/patents</u>.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS. SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE. INJURY. OR DEATH. THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES. INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR

APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- <u>Support</u>—Online technical support resources at ni.com/support include the following:
 - Self-Help Resources—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable <u>KnowledgeBase</u>, product <u>manuals</u>, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
 - Free Technical Support—All registered users receive free Basic Service, which includes access to hundreds of Applications Engineers worldwide in the <u>NI Discussion</u> <u>Forums</u> at ni.com/forums. National Instruments Applications Engineers make sure every question receives an answer.

For information about other <u>technical support options</u> in your area, visit ni.com/services or <u>contact</u> your local office at ni.com/contact.

- <u>Training and Certification</u>—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- <u>System Integration</u>—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Interface Functions

The interface function set can be used in combination with both high- and low-level functions to set up and close the interface and session. Use interface functions in both high-level and low-level applications.

imgInterfaceOpen	Opens an interface by name as specified by Measurement & Automation Explorer (MAX).
imgSessionOpen	Opens a session and returns a session ID.
imgClose	Closes a session or interface and unlocks and releases all buffers associated with the data type.
imgInterfaceQueryNames	Returns the interface name identified by the index parameter.
imgInterfaceReset	Performs a reset on the interface.

High-Level Functions

High-level functions are divided into the following types:

- <u>Snap functions</u>—Capture all or a portion of a single frame or field to the user buffer.
- <u>Grab functions</u>—Perform a continuous acquisition of data to a single buffer.
- <u>Ring and Sequence functions</u>—Start and stop a continuous acquisition of multiple fields or frames.
- <u>Signal I/O functions</u>—Control the trigger lines on image acquisition devices.
- <u>Miscellaneous functions</u>—Set and get the acquisition window region of interest and return information such as session status and buffer sizes.

Snap Functions

imgSnapPerforms a single frame or field acquisition.imgSnapAreaPerforms an area-specific frame or field acquisition.

Grab Functions

imgGrabSetup Configures and optionally starts a continuous acquisition.

- imgGrab Acquires the most current frame into the specified buffer. Call this function only after calling imgGrabSetup.
- imgGrabArea Performs a transfer from a continuous acquisition using the given parameters. Call this function only after calling imgGrabSetup.

Ring and Sequence Functions

imgRingSetup	Prepares a session for acquiring continuously and looping into a buffer list.
imgSequenceSetup	Prepares a session for acquiring a sequence into a buffer list.
imgSessionStartAcquisition	Starts an acquisition.

imgSessionStopAcquisition Stops an acquisition.

Signal I/O Functions

imgSessionTriggerConfigure2	Configures an acquisition to start based on an external trigger.
imgSessionLineTrigSource2	Configures triggering per line for acquisition from a line scan camera.
imgSessionTriggerClear	Disables all triggers on the session.
imgSessionTriggerDrive2	Configures the specified trigger line to drive a signal out.
imgSessionTriggerRead2	Reads the current value of the specified trigger line.
imgSessionTriggerRoute2	Drives the destination trigger line with the signal on the source trigger line.
imgSessionWaitSignal2	Waits for a signal to be asserted. This function returns when the specified signal is asserted.
imgSessionWaitSignalAsync2	Monitors for a signal to be asserted and invokes a user-defined callback when the signal is asserted.
imgEncoderResetPosition	Resets the absolute encoder position counter to 0.
imgPulseCreate2	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 original level.
imgPulseDispose	Disposes of a pulse ID.
imgPulseRate	Converts delay and width into delay, width, and timebase values needed by imgPulseCreate.
imgPulseStart	Starts the generation of a pulse. You must call imgPulseCreate first to configure the pulse.
imgPulseStop	Stops the generation of a pulse.

Miscellaneous Functions

imgSessionStatus Gets the current session status.

imgSessionSetROI Sets the acquisition region of interest.

imgSessionGetROI Gets the acquisition region of interest.

imgSessionGetBufferSize Gets the minimum buffer size needed for frame buffer allocation.

Low-Level Functions

The low-level function set supports all types of acquisition. Low-level functions are divided into the following types:

- <u>Acquisition functions</u>—Configure, start, and abort an image acquisition, or examine a buffer during an acquisition.
- <u>Attribute functions</u>—Examine and change NI-IMAQ or camera attributes.
- <u>Buffer management functions</u>—Set up objects such as buffer lists and buffers.
- Interface functions—Load and control the selected image acquisition device and cameras. These functions use information stored by Measurement & Automation Explorer (MAX).
- <u>Utility functions</u>—Display an image in a window, save an image to a file, or get detailed error information.
- <u>Serial communication functions</u>—Enables communication between the camera and devices that support serial communication.

Acquisition Functions

imgSessionAbort	Stops an asynchronous acquisition or synchronous continuous acquisition immediately.
imgSessionAcquire	Starts acquisition synchronously or asynchronously to the frame buffers in the associated session buffer list.
imgSessionConfigure	Specifies the buffer list to use with this session.
imgSessionCopyArea	Copies an area of a session buffer to a user- specified buffer.
imgSessionCopyBuffer	Copies a session buffer to a user-specified buffer.
imgSessionExamineBuffer	Extracts a buffer from a live acquisition. This function lets you lock a buffer out of a continuous loop sequence for processing when you are performing a ring (continuous) acquisition.
imgSessionReleaseBuffer	Releases a buffer that was previously held with imgSessionExamineBuffer.

Attribute Functions

imgGetAttribute	Returns an attribute for an interface or session.
imgSetAttribute	Sets an attribute for an interface or session.
imgGetCameraAttributeNumeric	Gets the value of numeric camera attributes.
imgGetCameraAttributeString	Gets the value of textual camera attributes.
imgSetCameraAttributeNumeric	Sets the value of numeric camera attributes.
imgSetCameraAttributeString	Sets the value of textual camera attributes.
imgSessionSetUserLUT8bits	Downloads a custom 8-bit lookup table to your image acquisition device.
imgSessionSetUserLUT16bits	Downloads a custom 16-bit lookup table to your image acquisition device.

Buffer Management Functions

imgCreateBuffer	Creates a user frame buffer based on the geometric definitions of the associated session.	
imgCreateBufList	Creates a buffer list.	
imgDisposeBuffer	Disposes of a user frame buffer created by imgCreateBuffer.	
imgDisposeBufList	Purges all image buffers associated with this buffer list.	
imgGetBufferElement	Gets an element of a specific type from a buffer list.	
imgSetBufferElement	Sets a buffer list element of a given type to a specific value.	
imgSessionClearBuffer	Clears image data from a session to the specified pixel value.	

Interface Functions

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

Utility Functions

imgPlot	Plots a buffer to a window.
imgPlotDC	Plots a buffer to device context.
imgSessionSaveBufferEx	Saves a buffer of a session to disk in bitmap, TIFF, or PNG format.
imgBayerColorDecode	Decodes the color information from Bayer encoded images.
imgCalculateBayerColorLUT	Calculates a look-up table (LUT) based on input gain values that is used in decoding the Bayer encoded images.
imgShowError	Returns a null-terminated string describing the error code.

Serial Communication Functions

imgSessionSerialWrite	Sends data out the serial port on boards that support serial.
imgSessionSerialRead	Reads in data from the serial port on boards that support serial.
imgSessionSerialReadBytes	Reads in an expected number of bytes from the serial port on image acquisition devices that support serial communication.
imgSessionSerialFlush	Clears the internal serial buffer.

Branch Offices

Office	Telephone Number
Australia	1800 300 800
Austria	43 662 457990-0
Belgium	32 (0) 2 757 0020
Brazil	55 11 3262 3599
Canada	800 433 3488
China	86 21 6555 7838
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	385 (0) 9 725 72511
France	33 (0) 1 48 14 24 24
Germany	49 89 7413130
India	91 80 41190000
Israel	972 0 3 6393737
Italy	39 02 413091
Japan	81 3 5472 2970
Korea	82 02 3451 3400
Lebanon	961 (0) 1 33 28 28
Malaysia	1800 887710
Mexico	01 800 010 0793
Netherlands	31 (0) 348 433 466
New Zealand	0800 553 322
Norway	47 (0) 66 90 76 60
Poland	48 22 3390150
Portugal	351 210 311 210
Russia	7 495 783 6851
Singapore	1800 226 5886
Slovenia	386 3 425 42 00

South Africa	27 0 11 805 8197
Spain	34 91 640 0085
Sweden	46 (0) 8 587 895 00
Switzerland	41 56 2005151
Taiwan	886 02 2377 2222
Thailand	662 278 6777
Turkey	90 212 279 3031
United Kingdom	44 (0) 1635 523545
United States (Corporate)	512 683 0100