IMAQ™

NI-IMAQ™ User Manual

June 2004 Edition
Part Number 370160D-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 450 510 3055,
Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838,
Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11,
France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427, India 91 80 51190000,
Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400,
Malaysia 603 9131 0918, Mexico 001 800 010 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 7 095 783 68 51, Singapore 65 6226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197,
Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227,
Thailand 662 992 7519, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

Important Information

Warranty

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

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

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

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

Copyright

Trademarks

Patents

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

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

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

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Contents

About This Manual

Chapter 1
Introduction to NI-IMAQ

Chapter 2
Software Overview

Chapter 3
Programming with NI-IMAQ

Chapter 4
Programming with NI-IMAQ VIs

Chapter 5
Programming with ActiveX Controls

Chapter 6
NI-IMAQ for LabVIEW Real-Time

About This Manual

The *NI-IMAQ User Manual* contains information you need to get started with NI-IMAQ.

This manual presents the basics of image acquisition, provides an overview of the software, and briefly describes the application development environments (ADEs) for NI-IMAQ.

Conventions

The following conventions appear in this manual:

» The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.

 This icon denotes a 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.

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

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

`monospace bold` Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.

Related Documentation

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

- National Instruments IMAQ hardware user manual
- *NI-IMAQ Function Reference Help*
- *NI-IMAQ VI Reference Help*
- *Measurement & Automation Explorer Help for IMAQ*

1

Introduction to NI-IMAQ

This chapter describes the NI-IMAQ software, lists the application development environments compatible with NI-IMAQ, describes the fundamentals of creating NI-IMAQ applications for Windows 2000/NT/XP, describes the files used to build these applications, and tells you where to find sample programs. For information about programming with NI-IMAQ for LabVIEW Real-Time, refer to Chapter 6, *NI-IMAQ for LabVIEW Real-Time*.

About the NI-IMAQ Software

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

NI-IMAQ contains methods for performing tasks ranging from device initialization to advanced high-speed image acquisition. The services you need for your application depends on the types of IMAQ devices you have and the complexity of your application.

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

NI-IMAQ includes a buffer manager that lets you simultaneously acquire and process data. NI-IMAQ uses direct memory access (DMA) to transfer all data.

Application Development Environments

NI-IMAQ supports the following Application Development Environments (ADEs) for Windows 2000/NT/XP:

- LabVIEW version 6.1 and later
- LabWindows™/CVI™ version 6.0 and later
- Microsoft Visual C/C++ version 6.0 and later
- Microsoft Visual Basic version 6.0 and later

Note Although NI-IMAQ has been tested with these ADEs, it also may work with other ADEs.

NI-IMAQ includes the CWIMAQ control, allowing you to use Visual Basic to control an IMAQ device. For more information about the CWIMAQ control, refer to Chapter 5, *Programming with ActiveX Controls*.

Fundamentals of Building Applications with NI-IMAQ

Architecture

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

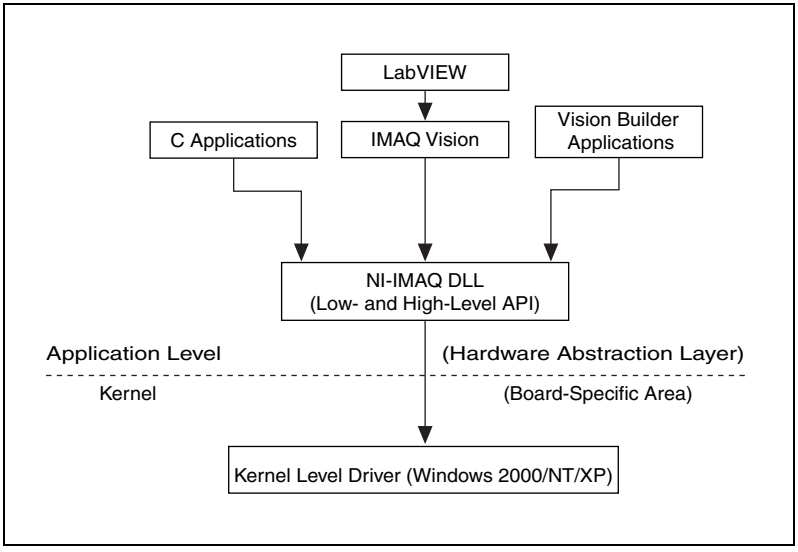

Figure 1-1. NI-IMAQ Architecture

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

The NI-IMAQ Libraries

The NI-IMAQ for Windows 2000/NT/XP 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 through function declarations. Your NI-IMAQ software kit contains function prototypes for all routines.

Creating an Application

This section outlines basic guidelines for developing NI-IMAQ applications in C. Detailed instructions about creating project and source files are not included. For information about creating and managing project files, refer to the documentation included with the development environment.

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.

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.

2

Software Overview

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

Introduction

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

- Generic functions
- High-level functions
  - Camera Control functions
  - Snap functions
  - Grab functions
  - *Ring* and *Sequence* functions
  - Signal I/O functions
  - Miscellaneous functions
- Low-level functions
  - Acquisition functions
  - Attribute functions
  - Buffer Management functions
  - Interface functions
  - Utility functions
  - Serial Communication functions

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

Generic Functions

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

`imgInterfaceOpen`	Opens an interface by name.
`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.

High-Level Functions

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

Camera Control Functions

Camera Control functions get and set camera-specific attributes and control the camera modes.

`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.

Snap Functions

Snap functions capture all or a portion of a single image to the user buffer.

`imgSnap`	Performs a single image acquisition.
`imgSnapArea`	Performs an area-specific image acquisition.

Grab Functions

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

`imgGrabSetup`	Configures and optionally starts a continuous acquisition.
`imgGrab`	Copies the current image 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

Ring and Sequence functions start and stop a continuous acquisition of multiple images.

`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

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

`imgSessionTriggerConfigure`	Configures an acquisition to start based on an external trigger.
`imgSessionLineTrigSource`	Configures triggering per line for acquisition from a line scan camera.
`imgSessionTriggerClear`	Disables all triggers on the session.
`imgSessionTriggerDrive`	Configures the specified trigger line to drive a signal out.
`imgSessionTriggerRoute`	Drives the destination trigger line with the signal on the source trigger line.
`imgSessionTriggerRead`	Reads the current value of the specified trigger line.
`imgSessionWaitSignal`	Waits for a signal to be asserted. This function returns when the specified signal is asserted.
`imgSessionWaitSignalAsync`	Monitors for a signal to be asserted and invokes a user-defined callback when the signal is asserted.
`imgPulseCreate`	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

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

`imgSessionStatus`	Gets the current session status.
`imgSessionConfigureROI`	Sets the acquisition region of interest.
`imgSessionFitROI`	Evaluates ROI you specify, and returns valid left, top, height, and width values. Returns an ROI that NI-IMAQ is guaranteed to be able to acquire, and that best accommodates the ROI you specify.
`imgSessionGetROI`	Gets the acquisition region of interest.
`imgSessionGetBufferSize`	Gets the minimum buffer size needed for frame buffer allocation.

Low-Level Functions

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

Acquisition Functions

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

`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

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

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

`imgSessionSetUserLUT8bits`	Downloads a custom 8-bit lookup table to your IMAQ device.
`imgSessionSetUserLUT16bits`	Downloads a custom 16-bit lookup table to your IMAQ device.

Buffer Management Functions

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

`imgCreateBuffer`	Creates a user frame buffer based on the attributes of the associated session, such as the ROI.
`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.

Interface Functions

Interface functions load and control the selected IMAQ device and cameras. These functions use interface information, which you can configure and store using Measurement & Automation Explorer (MAX).

`imgInterfaceQueryNames`	Returns the interface name identified by the index parameter.
`imgInterfaceReset`	Performs a hardware reset on the interface type, and reloads the interface file.

Utility Functions

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

`imgPlot`	Plots a buffer to a window.
`imgPlotDC`	Plots a buffer to device context.
`imgSessionSaveBufferEx`	Saves a buffer of a session to disk in BMP, 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 devices that support serial.
`imgSessionSerialRead`	Reads in data from the serial port on devices that support serial.
`imgSessionSerialFlush`	Clears the internal serial buffer.

3

Programming with NI-IMAQ

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

Introduction

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.

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.

High-Level Functions

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

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

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

Low-Level Functions

The low-level function set supports all types of acquisition. 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.

Generic Functions

The generic function set can be used in combination with both high-level 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.

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 *Interface Functions* section and the *Session Functions* section for information about how to manage interfaces and sessions, and 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 IMAQ 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. You 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, the camera file associated with each channel on the device, and the default channel.

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 Table 3-1.

Table 3-1. Interface Naming Convention

Interface Name	Device Installed
`img0`	Device 0
`img1`	Device 1
...	...
`img`*n*	Device *n*

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 then 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_ID interfaceID;
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.

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

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.

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

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

The camera attributes allow you to control camera-specific functions, such as integration time and pixel binning, directly from NI-IMAQ. You also can set these attributes in MAX on the **Camera Attributes** tab. Information about specific attributes for your camera is contained in `<my camera>.txt`, which can be found in the `NI-IMAQ/Camera Info` directory. For more information about camera attributes and their uses, refer to the camera documentation.

Note Currently, camera attributes are supported only by the IMAQ PCI/PXI-1409, IMAQ PCI-1410, IMAQ PCI/PXI-1422, IMAQ PCI-1424, and IMAQ PCI/PXI-1428.

All parameters configured for a camera type are stored in a camera (`.icd`) file. The camera file is then associated with a specific channel on an IMAQ 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 four 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. Camera files provided by National Instruments are already programmed with the camera serial command set. When you use `imgSetCameraAttributeString` or `imgSetCameraAttributeNumeric`, 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 IMAQ 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 is a sequence, acquisition in progress stays TRUE throughout 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 IMAQ 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.

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

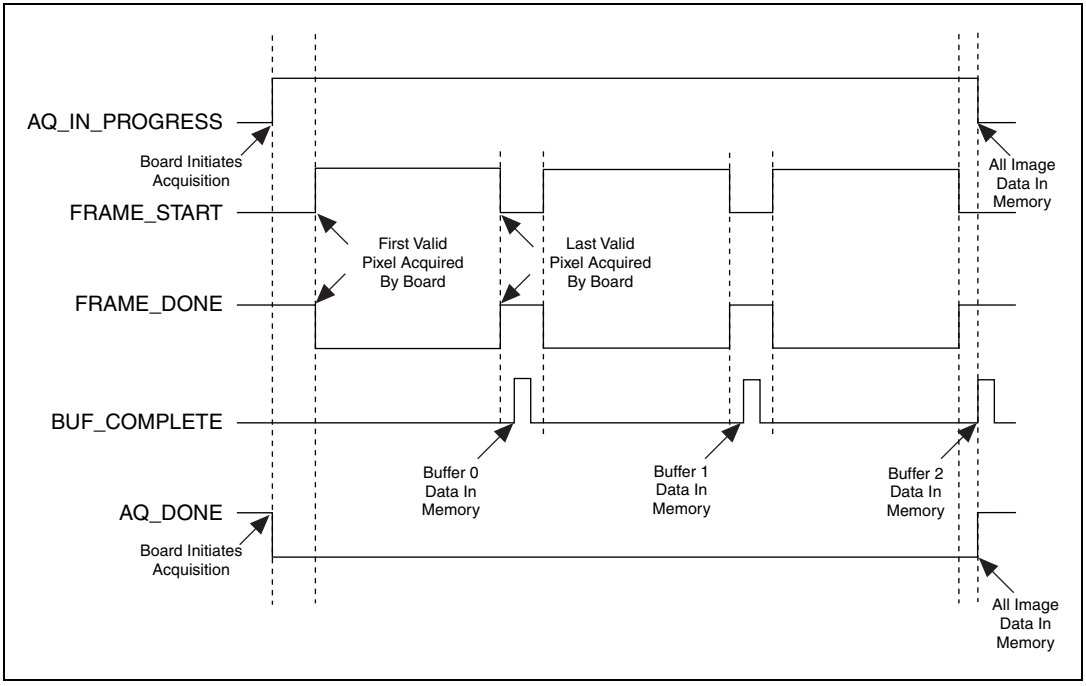

Figure 3-1. NI-IMAQ Status Signals

You can use the NI-IMAQ status signals for many purposes. 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 based on 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 to be invoked when **Buffer Complete** is asserted.

Line Scan Image Acquisition

Unlike area scan cameras, line scan cameras output only one line at a time. However, the programming interface for area scan and line scan cameras is identical. The driver builds up the lines acquired into a 2D image. The height of this image is set in 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 Appendix B, *Variable Height Acquisition*, for more information.

Triggering line scan images is similar to triggering area scan images. You can use the `imgSessionTriggerConfigure` function to trigger the start of each buffer. You can use the `imgSessionLineTrigSource` function to trigger each line 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. A continuous line scan acquisition into system memory is supported.

Geometric Definitions

The following list defines several terms you must 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.

Figure 3-2 illustrates the geometric relationship of these terms.

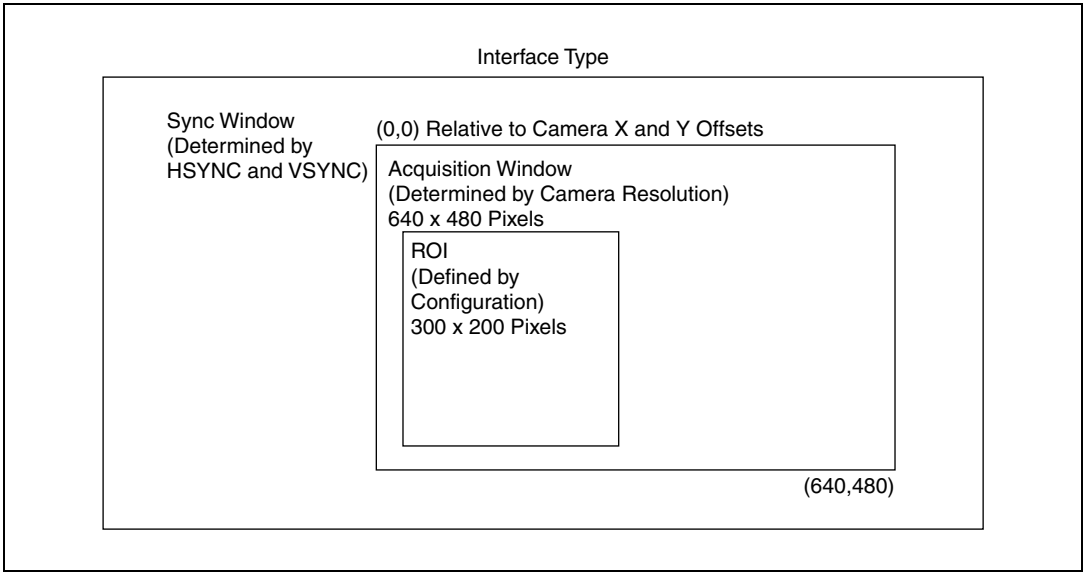

Figure 3-2. Geometric Relationship

Introductory Programming Examples

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

Note If you installed support for LabWindows/CVI, you can find the example C code in the `ni-imaq\samples` directory.

High-Level Snap Functions

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

When you invoke a snap, it initializes the 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. Figure 3-3 illustrates a typical snap programming order.

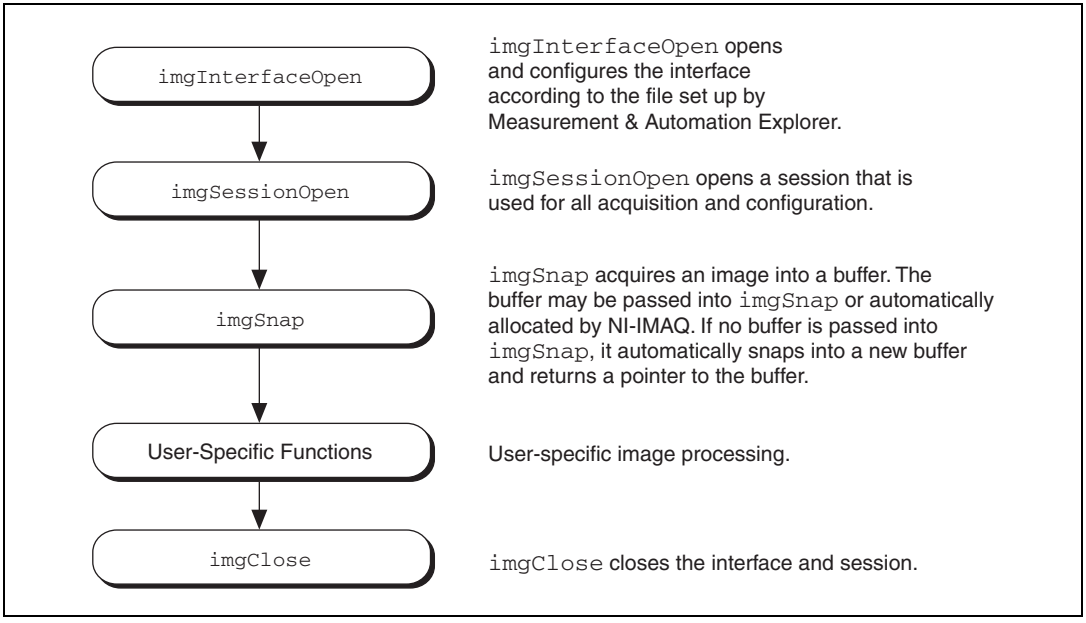

Figure 3-3. Snap Programming Flowchart

The `hlsnap.c` example demonstrates how to perform a single snap using `imgSnap`. The example opens an interface and a session and then performs a single snap. The buffer pointer that is passed to `imgSnap` is initialized to NULL, which instructs `imgSnap` to automatically allocate a buffer for the image. The size of the buffer is calculated based on the 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. Figure 3-4 illustrates a typical grab programming order.

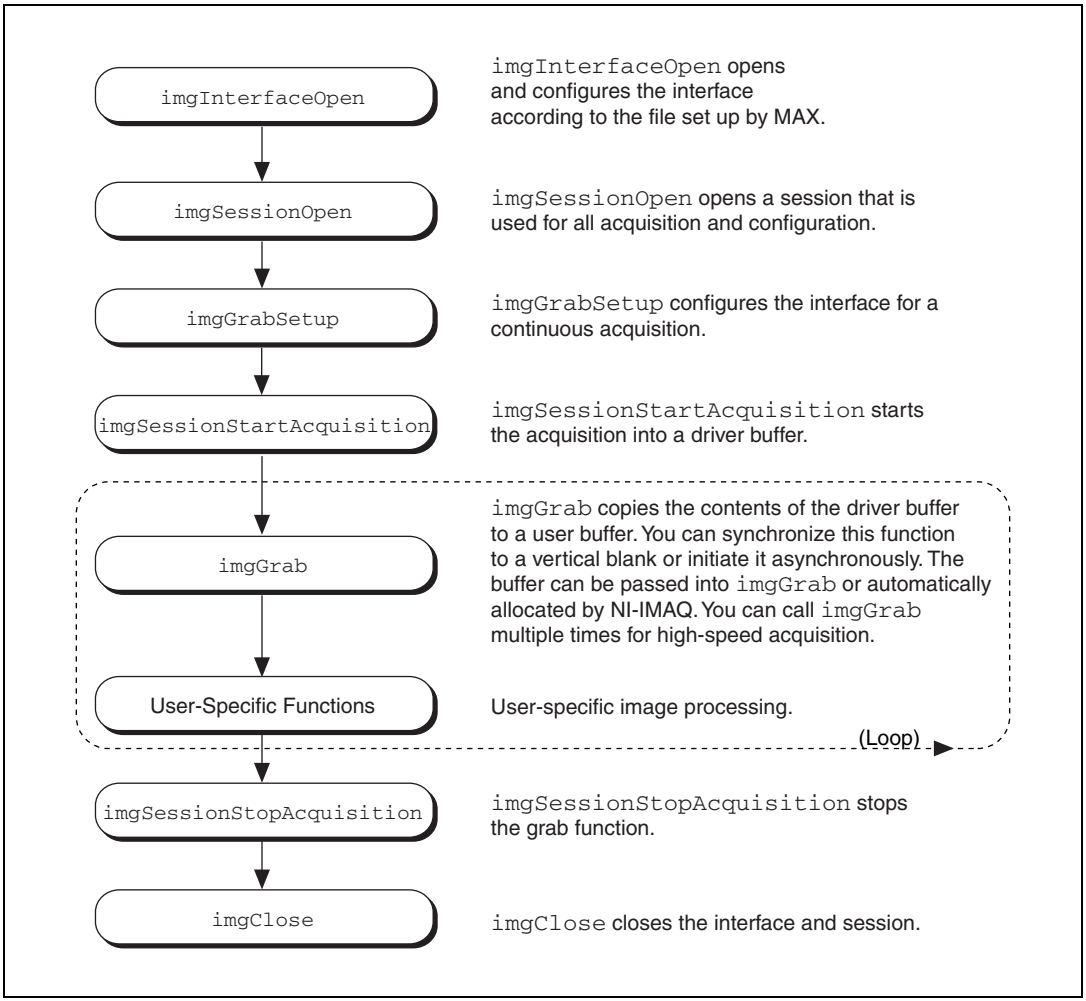

Figure 3-4. Grab Programming Flowchart

The hlgrab.c example demonstrates how to perform a grab using imgGrabArea. The example performs multiple grabs until an appropriate condition is met. The program configures the session to perform a grab operation by calling the imgGrabSetup function. The program then calculates the area to grab using the current ROI, rowPixels, and bytesPerPixel, and 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. Figure 3-5 illustrates a typical sequence programming order.

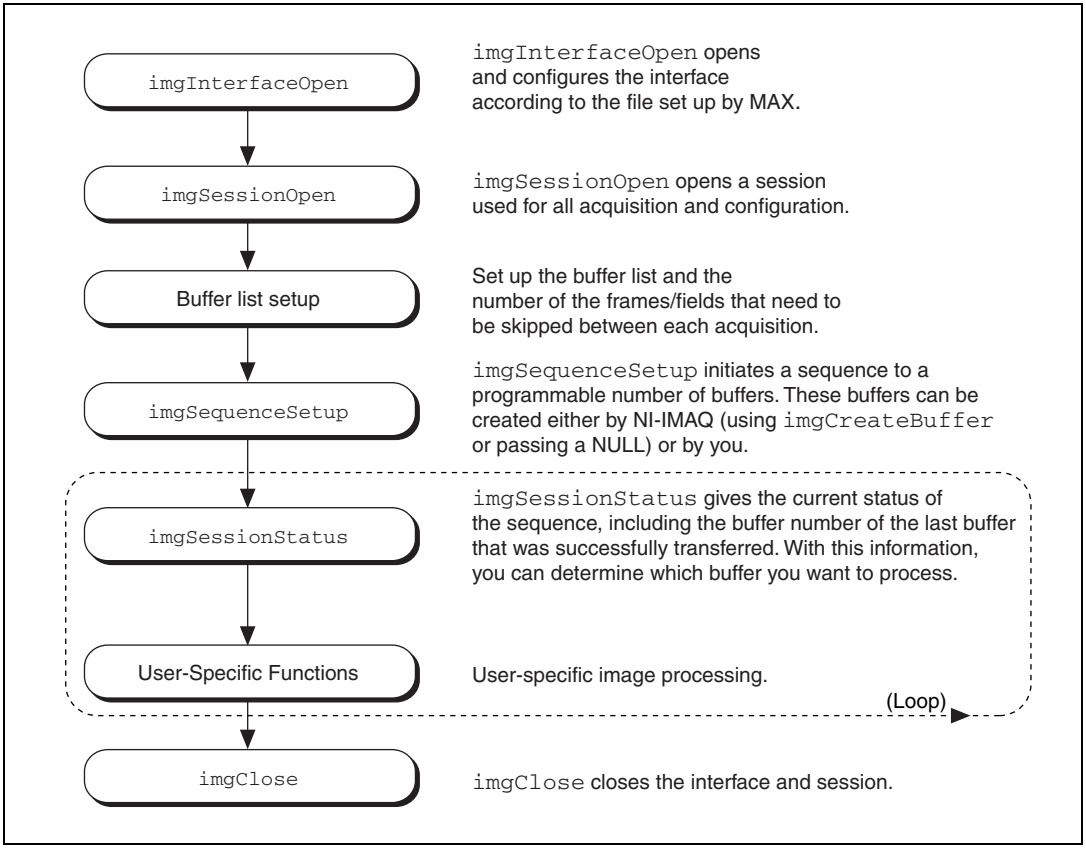

Figure 3-5. Sequence Programming Flowchart

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

High-Level Ring Functions

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

A ring initiates a continuous high-speed 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. Figure 3-6 illustrates a typical ring programming order.

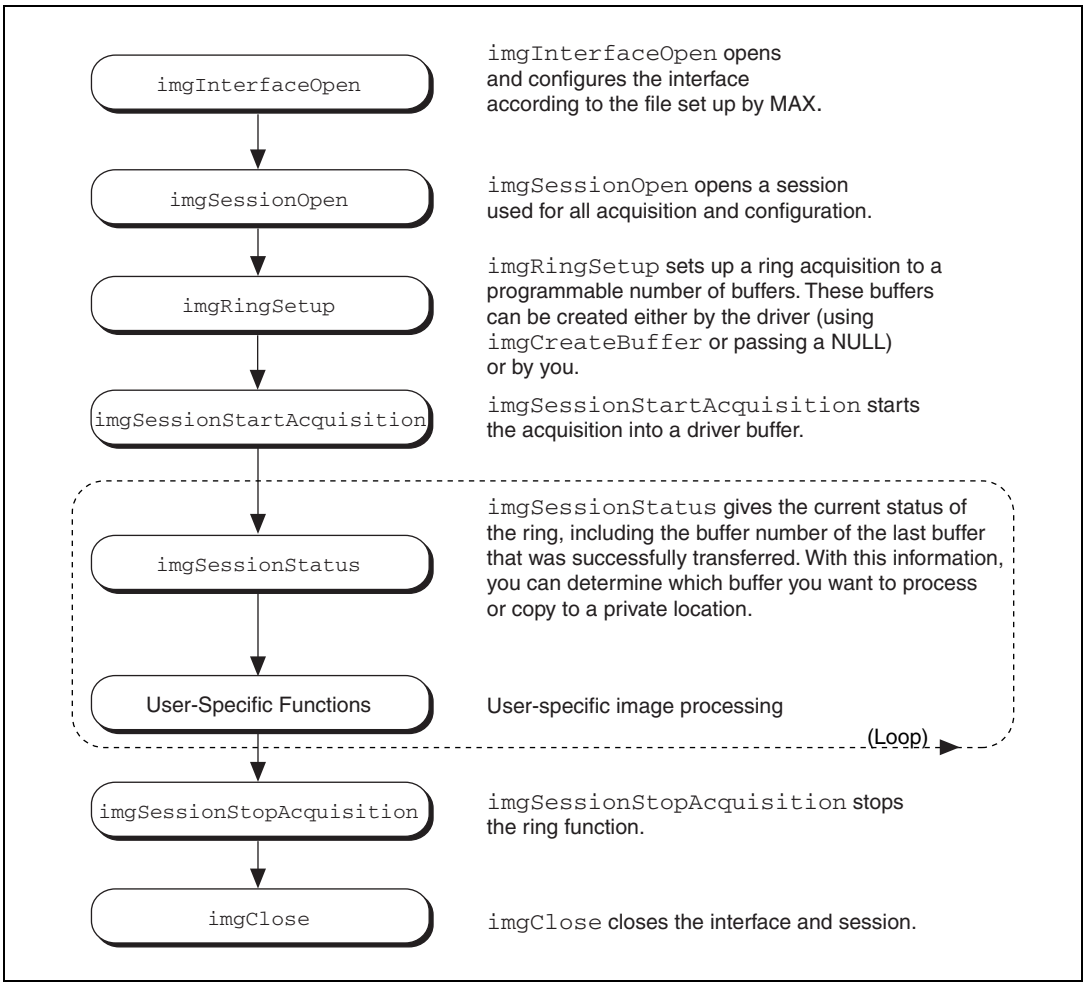

Figure 3-6. Ring Programming Flowchart

The `HLRing.c` example demonstrates how to perform a ring acquisition using `imgRingSetup`. The example sets up a ring containing five buffers and sets the skip count to zero, which causes the program to acquire every 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 when a sequence acquisition begins.

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.

Some applications need to send signals out from the IMAQ 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`.

Figure 3-7 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.

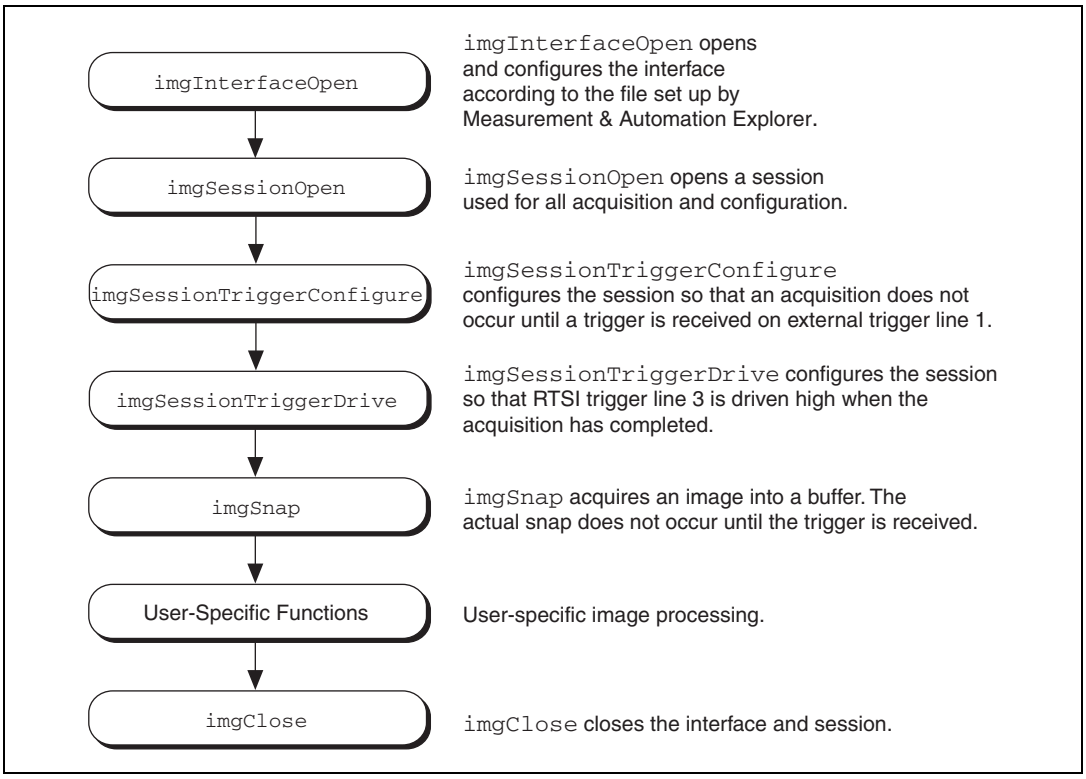

Figure 3-7. Signal I/O Function Programming Flowchart

Advanced Programming Examples

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

Performing a Snap Using Low-Level Functions

The `LLSnap.c` example demonstrates how to perform a snap acquisition using low-level calls. The example sets up a single-frame acquisition to a buffer allocated by NI-IMAQ. The program retrieves the acquisition window width of the selected camera and aligns it on a 4-byte boundary. You must align both the acquisition window width and `rowPixels` on a 4-byte boundary to ensure that the image is acquired properly. 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 4-byte 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 4-byte 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 4-byte 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 4-byte 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 `0xFFFFFFFF` as the IMG_ATTR_LAST_VALID_FRAME 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_FRAME 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.

4

Programming with NI-IMAQ VIs

This chapter describes how to use National Instruments programming and application software, such as LabVIEW and IMAQ Vision, with your IMAQ hardware and NI-IMAQ VIs. For information about programming with NI-IMAQ for LabVIEW Real-Time, refer to Chapter 6, *NI-IMAQ for LabVIEW Real-Time*.

Introduction

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 IMAQ device, is included with the NI-IMAQ software.

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

Before you start building your IMAQ application, you must be familiar with the following basic NI-IMAQ for LabVIEW concepts:

- Location of the NI-IMAQ examples
- Location of the NI-IMAQ VIs in LabVIEW
- Common NI-IMAQ VI parameters
- Buffer management
- NI-IMAQ acquisition types
- Acquisition VIs
- Triggering
- Image Display

- Camera attributes
- Error handling
- Error code format

Location of NI-IMAQ Examples

The NI-IMAQ VI examples illustrate some common applications. You can find these examples in the `LabVIEW\examples\IMAQ` directory for 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.

Location of the NI-IMAQ VIs

You can find the NI-IMAQ VIs in the **Functions** palette from your LabVIEW block diagram. In LabVIEW 6.1, select the **Motion and Vision** palette and then select the **Image Acquisition** palette, as shown in Figure 4-1. In LabVIEW 7.0 and later, select the **NI Measurements** palette, select the **Vision** palette, and then select the **Image Acquisition** palette, as shown in Figure 4-2.

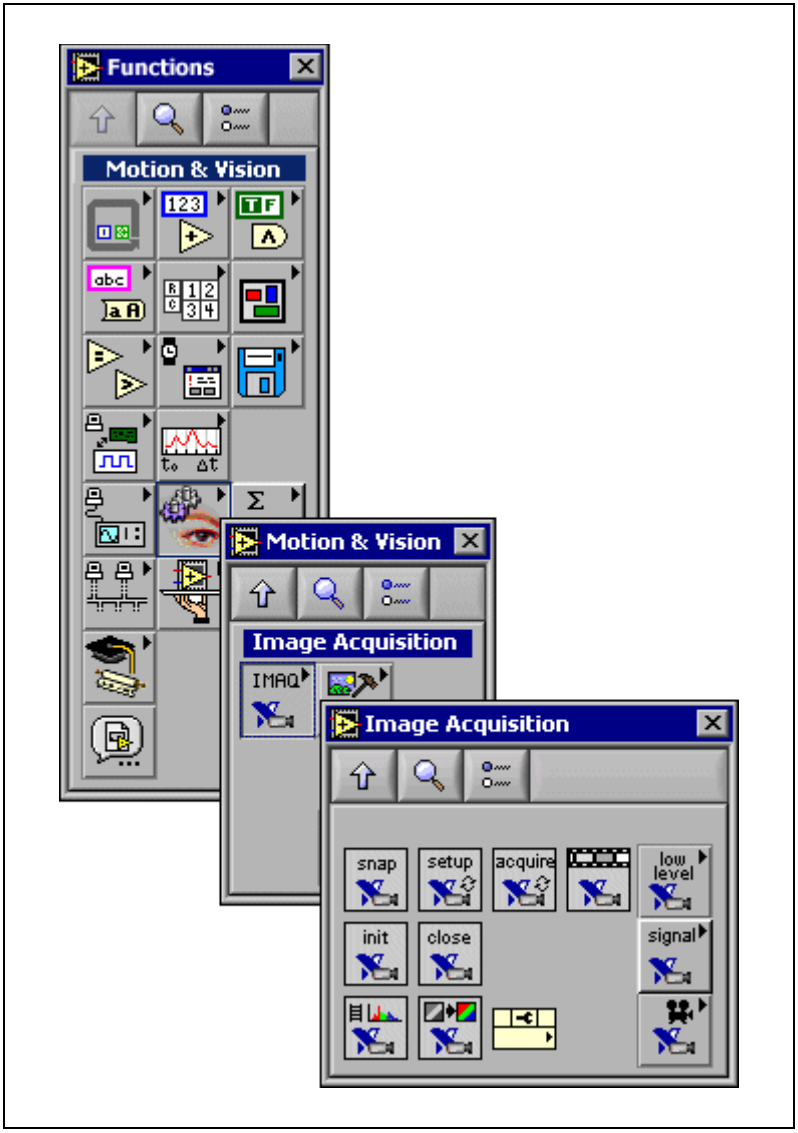

Figure 4-1. LabVIEW 6.1 Functions Palette with IMAQ Palette

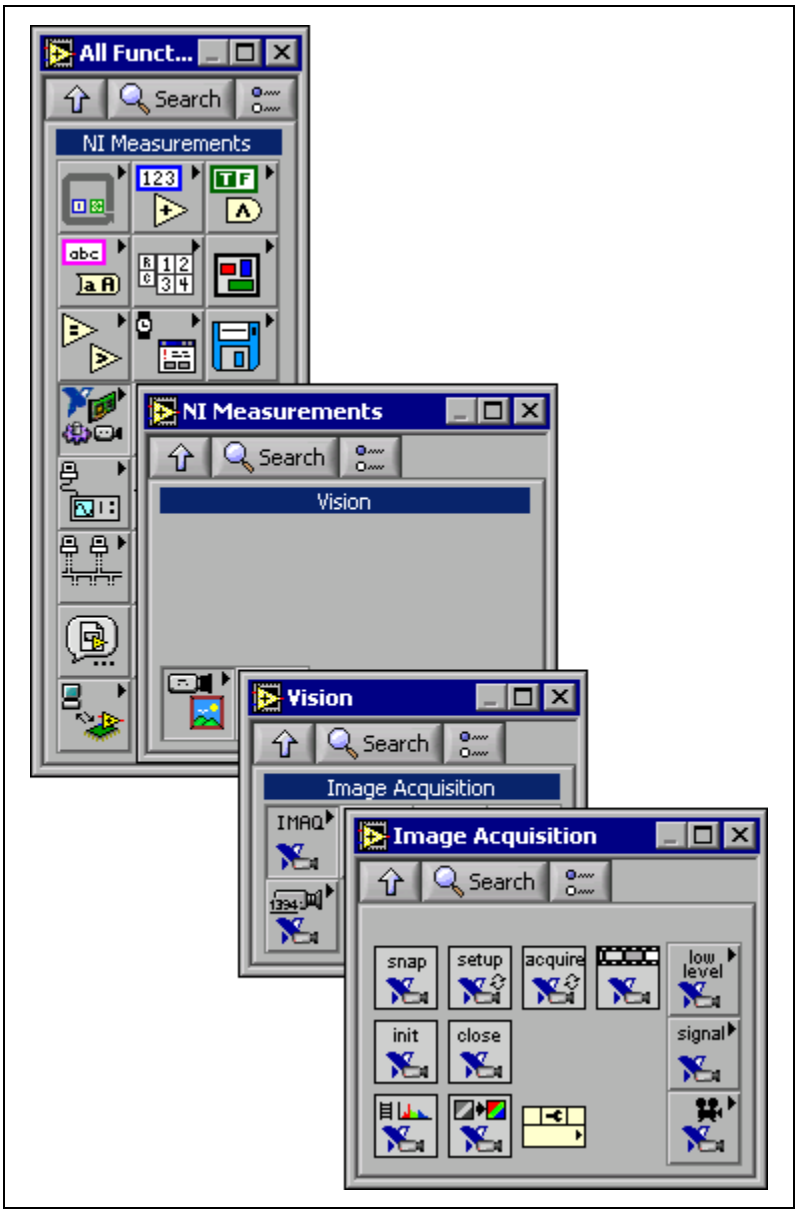

Figure 4-2. LabVIEW 7.0 and 7.1 Functions Palette with IMAQ Palette

The most commonly used IMAQ VIs are on the **Image Acquisition** palette. You can find VIs for basic acquisition and changing attributes. The **Image Acquisition»IMAQ Low Level** palette contains VIs for more advanced applications.

The **Image Acquisition»IMAQ Signal I/O** palette contains VIs for using triggers and pulse generation with IMAQ devices.

The **Image Acquisition»IMAQ Camera Control** palette contains VIs for controlling camera-specific attributes and serial commands.

Refer to the *NI-IMAQ VI Reference Help* for more information about these VIs.

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 the *Buffer Management* section 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.

Buffer Management

IMAQ Create and IMAQ Dispose manage image buffers in LabVIEW. IMAQ Create, shown in Figure 4-3, allocates an image buffer. **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 programatically 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.

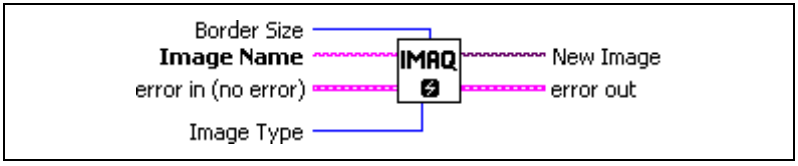

Figure 4-3. IMAQ Create

IMAQ Dispose, shown in Figure 4-4, frees the memory allocated for the image buffer. Call this VI only after the image is no longer required for processing.

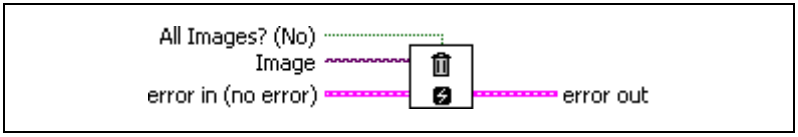

Figure 4-4. IMAQ Dispose

Note In LabVIEW 7.0 and later, images appear on LabVIEW block diagrams as purple wires. In LabVIEW 6.1, images appear as pink wires. There is no difference in functionality.

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. Figure 4-5 shows a simplified block diagram for using IMAQ Snap.

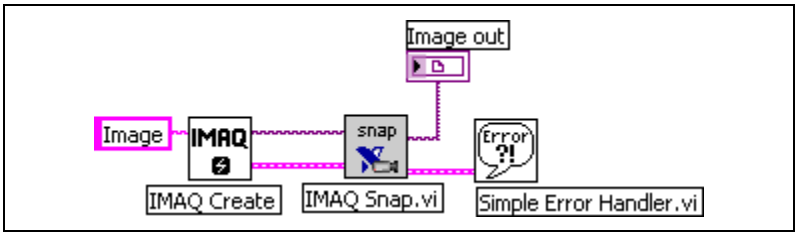

Figure 4-5. Acquiring an Image Using IMAQ Snap

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 G 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.

Figure 4-6 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.

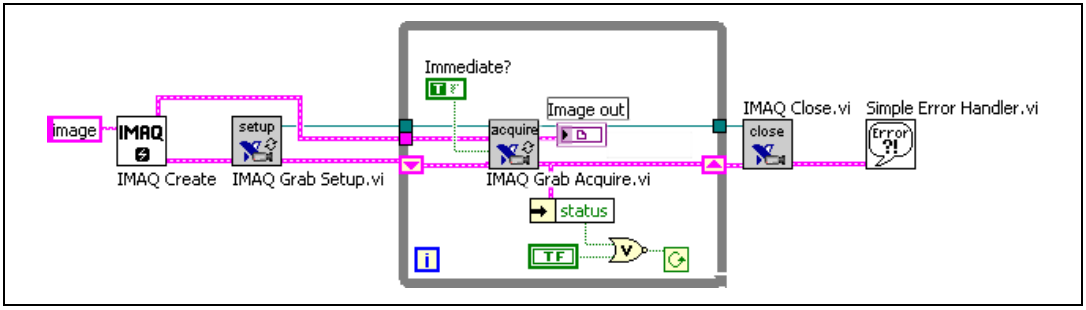

Figure 4-6. Acquiring Images Using IMAQ Grab

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.

Figure 4-7 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.

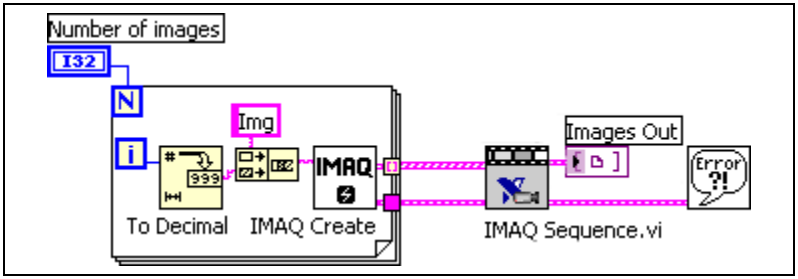

Figure 4-7. Acquiring Images Using IMAQ Sequence

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 `examples\imaq\IMAQ Low Level.llb` file.

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.

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 the *NI-IMAQ Acquisition Types* section. You can find examples of using the high-level acquisition VIs in the `examples\imaq\IMAQ High Level.llb` file.

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 IMAQ 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 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 `examples\imaq\IMAQ Low Level.llb`.

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 **Programs»National Instruments»Vision»Bayer 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.

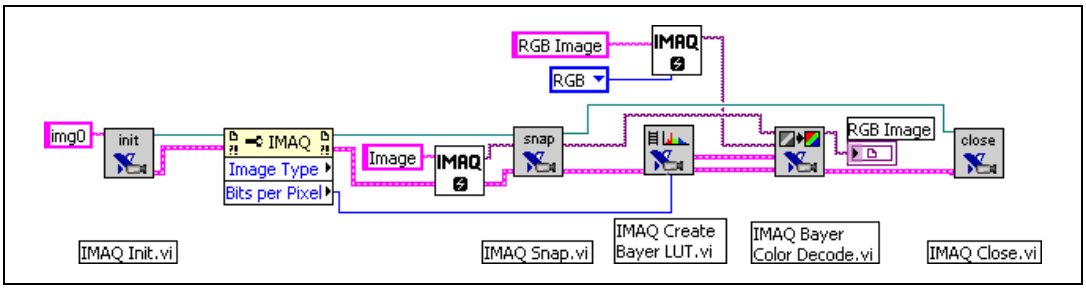

Figure 4-8. Acquiring an IMAQ Bayer Decoded Image

Triggering

You can use trigger lines on the IMAQ 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. On an IMAQ device, a trigger can be any TTL-level signal. All of the trigger lines are fully bidirectional so that the device can generate or receive the triggers on any line. The IMAQ PCI-1405, PCI/PXI-1407, and PCI/PXI-1411 have one external trigger line. The PCI/PXI-1409, PCI-1410, PCI/PXI-1422, PCI-1424, and PCI/PXI-1428 have four external trigger lines and seven Real-Time System Integration (RTSI) bus lines for general purpose use. Use the RTSI triggers to coordinate your IMAQ device with other National Instruments devices, such as data acquisition (DAQ) devices.

Use IMAQ Configure Trigger to configure the trigger conditions for an acquisition. You must call IMAQ Configure Trigger before the acquisition VI. The **Trigger line** input specifies which external or RTSI trigger receives the incoming 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. Figure 4-9 shows how to use IMAQ Configure Trigger to perform a snap acquisition based on a trigger.

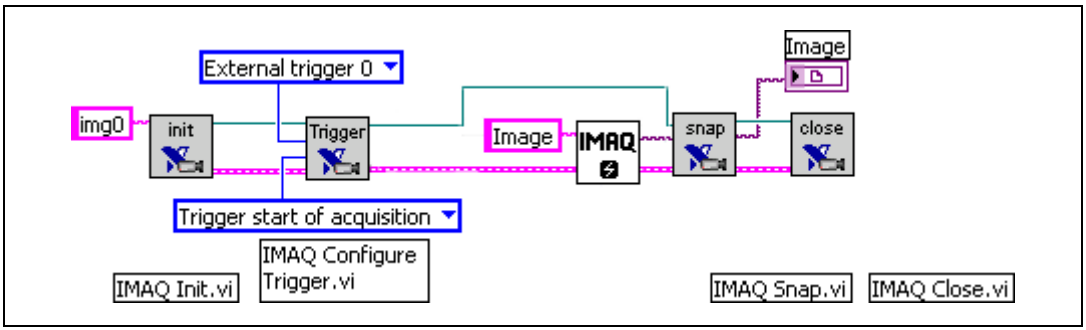

Figure 4-9. IMAQ Triggering

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. Figure 4-10 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**.

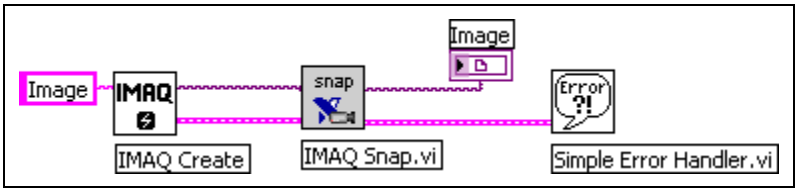

Figure 4-10. Displaying an Image Using an Image Control

If you have IMAQ 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. Figure 4-11 illustrates using IMAQ WindDraw to display an image acquired using IMAQ Snap. You can display images in the same way using any acquisition type. For more

information about the display capabilities of IMAQ Vision, refer to the *IMAQ Vision Concepts Manual*.

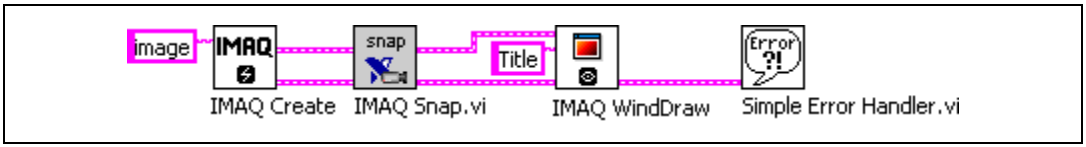

Figure 4-11. Displaying an Image Using IMAQ WindDraw

If you do not have either LabVIEW 7.0 or later or IMAQ Vision, you can display an image on 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. For more information about the picture control, refer to the *LabVIEW VI, Function, & How To Help*. Figure 4-12 illustrates using a picture control to display an RGB image acquired with IMAQ Snap.

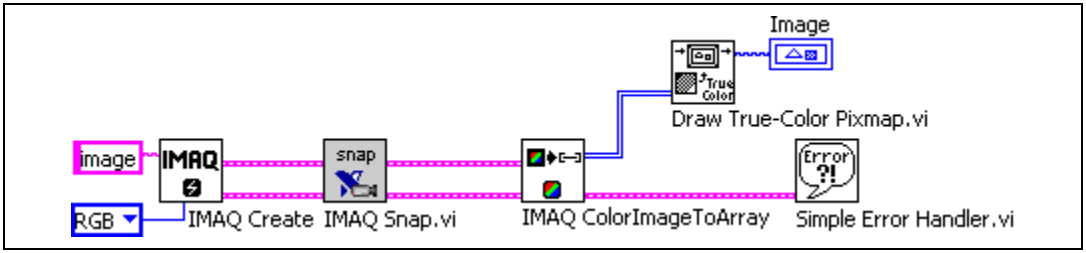

Figure 4-12. Using a Picture Control to Display an RGB Image

Camera Attributes

The camera attribute VIs allow you to control camera functions, such as integration time and pixel binning, directly from LabVIEW. These camera attributes are camera-specific, and you also can set them from MAX on the **Camera Attributes** tab. You can find information about specific attributes for your camera in the `<my camera>.txt` file, which is in the `camera info` directory in your `ni-imaq` directory. For more information about the camera attributes and their uses, refer to the camera documentation.

Note Currently only the IMAQ PCI/PXI-1409, IMAQ PCI-1410, IMAQ PCI/PXI-1422, IMAQ PCI-1424, and IMAQ PCI/PXI-1428 devices support camera attributes.

Use the Set Camera Attribute VI to set the value of a camera attribute. The camera attribute file mentioned above lists all attributes for the camera and each attribute description contains four fields: **Attribute Name**, **Description**, **Data Type**, and **Possible Values**. The **Attribute Name** field contains the name of the attribute in quotes. Wire this field to the **Camera Attribute** input on 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 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 the IMAQ Set Camera Attribute VI. Figure 4-13 shows how to use IMAQ Set Camera Attribute to set the value of a float camera attribute.

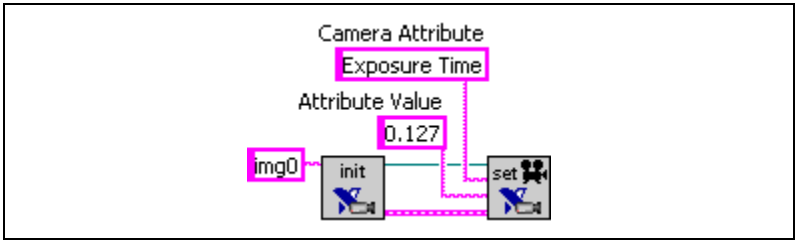

Figure 4-13. IMAQ Set Camera Attribute

Use the IMAQ Get Camera Attribute VI to get the value of a camera attribute. Use the camera attribute file described above 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. Figure 4-14 shows how to use IMAQ Get Camera Attribute with **Scan from String** to get the value of a float camera attribute.

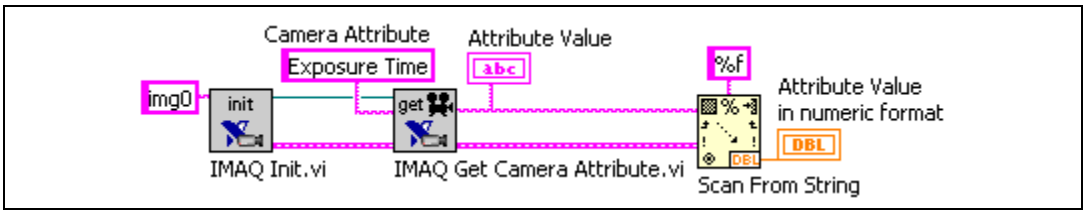

Figure 4-14. Using the IMAQ Get Camera Attribute

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.

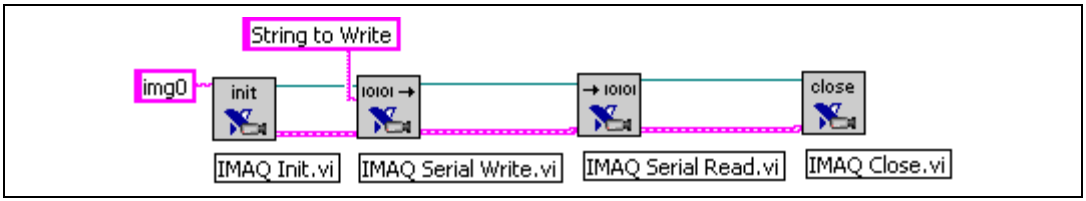

Figure 4-15. Using the IMAQ Serial VIs

Error Handling

Every NI-IMAQ VI contains an **error in** input cluster and an **error out** output cluster, as shown in Figure 4-16. 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.

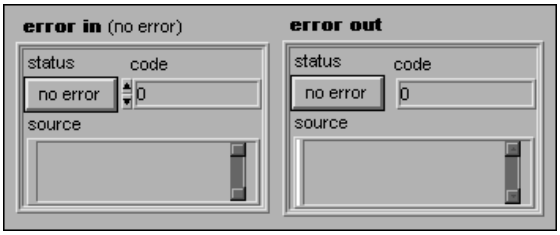

Figure 4-16. Error Clusters

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. Figure 4-17 shows how to wire an NI-IMAQ VI to the Simple Error Handler VI.

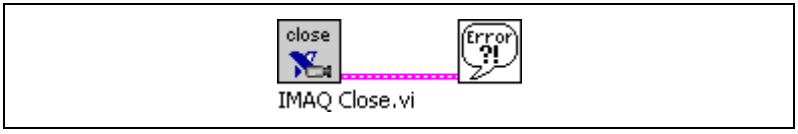

Figure 4-17. Error Checking Using the Simple Error Handler VI

Error Code Format

The error format, which is the same for all NI-IMAQ VIs, is as follows:

error in (no error) is a cluster that describes the error status before this VI executes. If **error in** indicates that an error occurred before this VI was called, this VI may choose not to execute its function, but just pass the error through to its error out cluster. If no error has occurred, then this VI executes normally and sets its own error status in error out. Use the error handler VIs to look up the error code and to display the corresponding error message. Using **error in** and **error out** clusters is a convenient way to check errors and to specify execution order by wiring the error output from one subVI to the error input of the next.

status is TRUE if an error occurred before this VI was called, or FALSE if not. If status is TRUE, code is a non-zero error code. If status is FALSE, code can be zero or a warning code.

code is the number identifying an error or warning. If status is TRUE, code is a non-zero error code. If status is FALSE, code can be zero or a warning code. Use the error handler VIs to look up the

meaning of this code and to display the corresponding error message.

source is a string that indicates the origin of the error, if any. Usually source is the name of the VI in which the error occurred.

error out is a cluster that describes the error status after this VI executes. If an error occurred before this VI was called, **error out** is the same as **error in**. Otherwise, **error out** shows the error, if any, that occurred in this VI. Use the error handler VIs to look up the error code and to display the corresponding error message. Using **error in** and **error out** clusters is a convenient way to check errors and to specify execution order by wiring the error output from one subVI to the error input of the next.

status is TRUE if an error occurred, or FALSE if not. If **status** is TRUE, code is a non-zero error code. If **status** is FALSE, code can be zero or a warning code.

code is the number identifying an error or warning. If **status** is TRUE, code is a non-zero error code. If status is FALSE, code can be zero or a warning code. Use the error handler VIs to look up the meaning of this code and to display the corresponding error message.

source is a string that indicates the origin of the error, if any. Usually **source** is the name of the VI in which the error occurred.

5

Programming with ActiveX Controls

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

Introduction

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

Documentation and Examples

This chapter 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:

- `www.msdn.microsoft.com`
- Documentation that accompanies Microsoft Visual Studio

In addition to this manual, several documentation resources are available to help you create your vision application:

- *IMAQ Vision Concepts Manual*—If you are new to machine vision and imaging, read this manual to understand the concepts behind IMAQ Vision.
- *IMAQ Vision for Visual Basic 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 `MeasurementStudio\VB\Samples\MachineVision` directory.

- CWMachineVision source code—If you want to see the source code for the CWMachineVision control, go to the `MeasurementStudio\VB\SourceCode\IMAQ` directory.
- Application Notes—If you want to know more about advanced IMAQ 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 NI Developer Zone 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.

IMAQ Vision for Visual Basic Organization

IMAQ 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 the *ActiveX Objects* section for more information about the ActiveX objects.

CWIMAQ Control

Use this control to configure and perform an acquisition from an IMAQ acquisition device. The CWIMAQ control has property pages that allow you to modify various properties to configure your acquisition and gather information about your IMAQ 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 You must purchase IMAQ Vision for Visual Basic to use this control.

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 the *ActiveX Objects* section 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 IMAQ Vision for Visual Basic installed. Contact your National Instruments sales representative, or visit `ni.com` to purchase IMAQ 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 IMAQ Vision for LabWindows/CVI. However, if you decide to use IMAQ 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 CWIMAQImages 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 IMAQ device. You can use IMAQ 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 IMAQ image acquisition devices. All of the lines are fully bidirectional, allowing your IMAQ device to generate or receive the triggers on any line. The IMAQ PCI-1405, IMAQ PCI/PXI-1407, and IMAQ PCI/PXI-1411 feature one external trigger line. The IMAQ PCI/PXI-1409, IMAQ PCI-1410, IMAQ PCI/PXI-1422, IMAQ PCI-1424, and IMAQ PCI-1428 each feature four external trigger lines and seven Real-Time System Integration (RTSI)

bus lines for general purpose use. Use these RTSI triggers to coordinate the IMAQ 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

Display an image using the CWIMAQViewer control. 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.

```
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`. For more information about color palettes, refer to Chapter 2, *Display*, of the *IMAQ Vision Concepts Manual*.

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 pop-up menu on the viewer.

Camera Attributes

Camera attributes allow you to control camera functions, such as integration time and pixel binning. These camera attributes are camera-specific. You can access the camera attributes using `CWIMAQ.CameraAttribute`. You also can set them in MAX on the **Camera Attributes** tab. You can find information about specific attributes for your camera in the `<my camera>.txt` file, which is in your `NI-IMAQ\Camera Info` directory. For more information about specific camera attributes and their uses, refer to your camera documentation.

Note Currently, only the IMAQ PCI/PXI-1409, IMAQ PCI-1410, IMAQ PCI/PXI-1422, IMAQ PCI-1424, and IMAQ PCI/PXI-1428 devices support camera attributes.

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 exception-handling 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.

6

NI-IMAQ for LabVIEW Real-Time

This appendix introduces NI-IMAQ for LabVIEW Real-Time (RT) module and describes how you can use the driver software to create an image acquisition application for a real-time, embedded target.

About NI-IMAQ for LabVIEW RT

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

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

System Components and Requirements

The NI-IMAQ for LabVIEW RT system consists of a development system and one or more deployed RT targets. Refer to Figure 6-1 for an illustration of a complete NI-IMAQ for LabVIEW RT system.

Development System

The development system is made up of two components—a Pentium-based 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 an RT 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 Measurement & Automation Explorer (MAX) to share configuration settings and software.

Note You must have a network connection during development to configure your RT 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 2000/NT/XP
- 233 MHz Pentium-class MMX processor
- 1024 × 768 (or higher) resolution video adapter, using 16-bit color
- Minimum of 64 MB RAM (128 MB recommended)
- Minimum of 50 MB of free hard disk space
- LabVIEW 7.0 or later and LabVIEW RT 7.0 or later

National Instruments PXI System

Select the PXI controller that best meets the needs of the application. Table 6-1 lists the different controllers you can use with NI-IMAQ for LabVIEW RT.

Table 6-1. National Instruments RT Series PXI Controllers

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

The PXI system also must meet the following minimum system requirements:

- Network adapter
- National Instruments IMAQ 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 RT targets for deployment. These deployed systems use the same hardware and software as the development LabVIEW RT target.

The deployed LabVIEW RT devices can be connected to the same subnet as the host development system, or you can connect them to a different subnet as required by your application. These connections are illustrated in Figure 6-1.

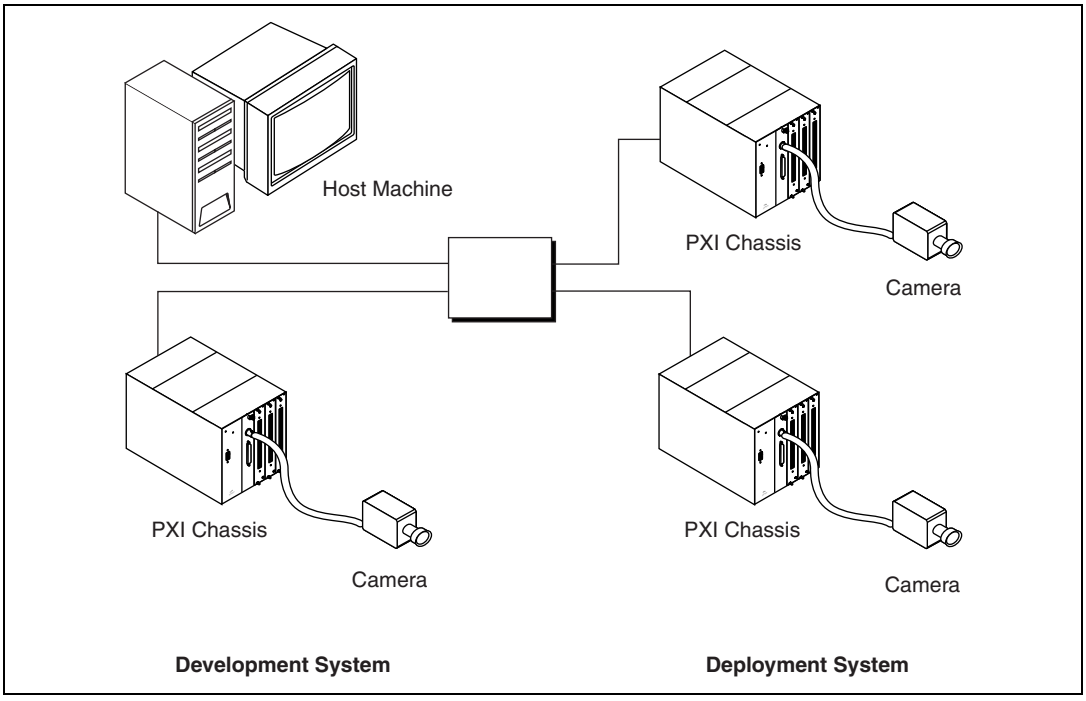

Figure 6-1. NI-IMAQ for LabVIEW RT System Components

Documentation and Examples

In addition to this manual, several documentation resources are available to help you create an image acquisition application:

- Example programs—To view examples of how to create specific applications in LabVIEW 6.1 or earlier, go to `LabVIEW\Examples\IMAQ`, or refer to the help file at **Help»Search IMAQ Examples** from within LabVIEW. In LabVIEW 7.0 or later, select **Help»Find Examples** to browse or search installed example VIs and example VIs on the Web.
- NI Developer Zone—For more information about developing your image acquisition application, visit the NI Developer Zone at `ni.com/zone`. NI Developer Zone contains example programs, tutorials, technical presentations, the Instrument Driver Network, a measurement glossary, an online magazine, and a product advisor, as well as a community area where you can share ideas, questions, and source code with developers around the world.

Displaying Images with NI-IMAQ and LabVIEW RT

NI-IMAQ and LabVIEW RT 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 RT system.

When you use Remote Display, the RT 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.

Tip Try the HL Grab to Real-Time Target with Display example for a demonstration of interactively setting the RT Video Out display mode.

Troubleshooting NI-IMAQ for LabVIEW RT

This section describes solutions and suggestions for common errors in NI-IMAQ for LabVIEW RT.

PXI Controller Errors

Why can't 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 RT 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.
- Make sure 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 LabVIEW RT, you must have a PXI controller that supports this feature. Refer to Table 6-1, *National Instruments RT Series PXI Controllers*, for a list of controllers that support this feature.

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

Why can't 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.

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.

A

Color Basics

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

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 spectrum—the portion that goes from the red edge of infrared light (the longest wavelength) to the blue edge of ultraviolet light (the shortest wavelength). This continuous spectrum is called the visible spectrum.

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

- *Hue*, which is the perceived dominant color. Hue depends directly on the wavelength of a color.
- *Saturation*, which is dependent on the amount of white light present in a color. Pastels typically have a low saturation while very rich colors have a high saturation. For example, pink typically has a red hue but has a low saturation.
- *Luminance*, which is the brightness information in the video picture. The luminance signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.

Image Representations

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

RGB

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

0	RED	GREEN	BLUE

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

Color Planes

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

Hue, Saturation, and Luminance Planes

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.

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

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

$$\text{Saturation} = \sqrt{x^2 + y^2}$$

$$\text{Hue} = \text{ATN2}(\text{Y}, \text{X}) \times 128/\pi$$

where

$$\text{X} = Red - \frac{1}{2}\,(Green + Blue)$$

$$\text{Y} = \sqrt{3}\,\times (Green - Blue)/2$$

Refer to Figure A-1 for an illustration of the HSL color space.

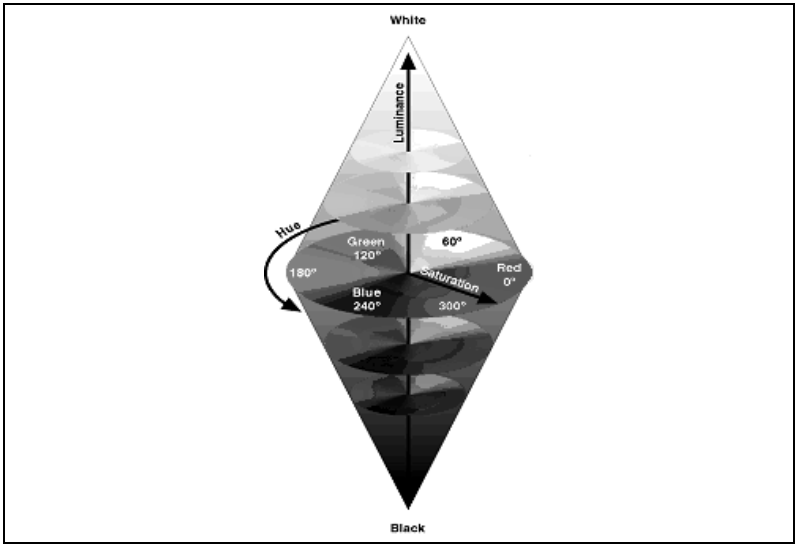

Figure A-1. HSL Color Space

32-Bit HSL

You also can pack the three 8-bit HSL planes in one array of 32-bit integers, which is equivalent to the 32-bit RGB representation.

Camera Types

You can use three basic video camera types for color acquisition—RGB cameras, composite color video cameras, and Bayer encoded cameras.

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

A composite color camera transmits the video signal on a single wire. The signal is composed of two 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 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, light-sensitive 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 intensity.

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.

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.

B

Variable Height Acquisition

Introduction

When you perform a line scan application, you may not know the exact size of the object you are imaging. A good example is a conveyer belt application in which you need to perform a continuous acquisition of objects of multiple sizes. For such applications, National Instruments has introduced a mode of triggered acquisition on its PCI/PXI-1409, PCI-1410, PCI/PXI-1422, PCI-1424, and PCI/PXI-1428 IMAQ devices that allows you to acquire a variable number of lines in a multiple buffered application.

In this mode, you supply the IMAQ device a trigger which is asserted when you want to begin capture. The IMAQ 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 to image. 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.
2. Click **Devices and Interfaces»NI-IMAQ Devices»PCI/PXI-14xx»XXX**, where **XXX** represents the camera you want to use.
3. Select the **Acquisition Parameters** tab.
4. Enter the maximum possible number of lines in **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 on 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 if you use LabVIEW or C, but the concept is the same in either case. 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 ready-to-run example showing how to acquire a variable number of lines with a line scan camera.

- In LabVIEW, refer to the `LL VHA Ring.vi` in the `IMAQ Signal IO.llb`.
- In LabWindows/CVI, refer to the `VHA Ring.prj` in the `CVI Samples` folder.
- In Microsoft C++, refer to the `VHA Ring.mdp` in the `NI-IMAQ Samples` folder.

C

Technical Support and Professional Services

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

- **Support**—Online technical support resources at ni.com/support include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at ni.com/exchange. National Instruments Application Engineers make sure every question receives an answer.
- **Training and Certification**—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program 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. Phone numbers for our worldwide offices are listed at the front of this manual. 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.

Glossary

Symbol	Prefix	Value
m	milli	10^{-3}
k	kilo	10^{3}

A

A/D	Analog-to-digital.
acquisition window	The image size specific to a video standard or camera resolution.
active line region	The region of lines actively being stored, which is defined by a line start relative to the vertical synchronization signal (VSYNC) and a line count.
active pixel region	The region of pixels actively being stored, which is defined by a pixel start relative to the horizontal synchronization signal (HSYNC) and a pixel count.
address	Value that identifies a specific location (or series of locations) in memory.
analog camera	A camera that transforms light into pixels and then translates the level of light for each pixel into an analog video signal.
antichrominance filter	Removes the color information from the video signal.
API	Application programming interface.
area	A rectangular portion of an acquisition window or frame that is controlled and defined by software.
arithmetic operators	The multiply, divide, add, subtract, and remainder image operations.
array	Ordered, indexed set of data elements of the same type.
aspect ratio	The ratio of picture or image width to height.
asynchronous	Property of a function or operation that begins an operation and returns control to the program before the completion or termination of the operation.

B

b	Bit. One binary digit, either 0 or 1.
B	Byte. Eight related bits of data, an eight-bit binary number. Also denotes the amount of memory required to store one byte of data.
back porch	The area of the video signal between the rising edge of the horizontal synchronization signal (HSYNC) and the active video information.
Bayer encoding	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.
binary image	An image in which the objects usually have a pixel intensity of 1 (or 255) and the background has an intensity value of 0.
bit depth	The number of bits (n) used to encode the value of a pixel. For a given n, a pixel can take $2n$ different values. For example, if n equals eight bits, a pixel can take 256 different values ranging from 0 to 255. If n equals 16 bits, a pixel can take 65,536 different values, ranging from 0 to 65,535 or –32,768 to 32,767.
black reference level	The level that represents the darkest an image can get. *See also* white reference level.
blob	Binary large object. A connected region or grouping of pixels in an image in which all pixels have the same intensity level. Blobs are also referred to as objects or particles.
blob analysis	A series of processing operations and analysis functions that produce some information about the objects in an image.
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.

C

cache	High-speed processor memory that buffers commonly used instructions or data to increase processing throughput.
CCIR	Comite Consultatif International des Radiocommunications. A committee that developed standards for video signals. Also used to describe signals, devices, and cameras that adhere to the CCIR standards.
chroma	The color information in a video signal.
chrominance	*See* chroma.
CSYNC	Composite synchronization signal. Signals in a color video system that multiplex all picture information into a single signal, such as NTSC, PAL, or SECAM.

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.
dB	Decibel. The unit for expressing a logarithmic measure of the ratio of two signal levels: $dB = 20\log_{10} V1/V2$, for signals in volts.
default setting	A default parameter value recorded in the driver. In many cases, the default input of a control is a certain value (often 0).
determinism	Characteristic of a system that describes how reliably it can respond to external events, or perform operations, within a given time limit.
device	Plug-in data or image acquisition device that can contain multiple channels and conversion devices.
digital camera	A camera that transforms light information into pixels and then translates the level of light for each pixel into a digital number inside the camera.

DLL	Dynamic link library. A software module in 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 between computer memory and 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.
DNS	Domain Name Server.
DRAM	Dynamic RAM.
driver	Software that controls a specific hardware device, such as an IMAQ or DAQ device.
dynamic range	The ratio of the largest signal level a circuit can handle to the smallest signal level it can handle (usually taken to be the noise level), normally expressed in decibels.

E

Ethernet	A high-speed local area network (LAN) which conforms to the IEEE 802.3 standard.
external trigger	A voltage pulse from an external source that triggers an event, such as A/D conversion.

F

field	For an interlaced video signal, a field is half the number of horizontal lines needed to represent a frame of video. The first field of a frame contains all the odd-numbered lines, the second field contains all of the even-numbered lines.
FIFO	First-in first-out memory buffer. The first data stored is the first data sent to the acceptor. FIFOs are used on IMAQ devices to temporarily store incoming data until that data can be retrieved.
frame	A complete image. In interlaced formats, a frame is composed of two fields.

front porch — The area of a video signal between the start of the horizontal blank and the start of the horizontal synchronization signal (HSYNC).

ftp — File transfer protocol. Protocol based on TCP/IP to exchange files between computers.

function — A set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed.

G

gamma — The nonlinear change in the difference between the video signal brightness level and the voltage level needed to produce that brightness.

genlock — The process of synchronizing a video source to the signal from a separate video source. The circuitry aligns the video timing signals by locking together the horizontal, vertical, and color subcarrier frequencies and phases and generates a pixel clock that clocks pixel data into memory for display or into another circuit for processing.

grab — Performs an acquisition that loops continually on one buffer. You obtain a copy of the acquisition buffer by grabbing a copy to a separate buffer that can be used for analysis.

H

hardware — The physical components of a computer system, such as the circuit boards, plug-in devices, chassis, enclosures, peripherals, and cables.

host PC — The computer on which the LabVIEW RT Development System is used to develop and target VIs to RT Series hardware.

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

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

I

I/O	Input/output. The transfer of data to/from a computer system, including communications channels, operator interface devices, and/or data acquisition and control interfaces.
image processing	Encompasses various processes and analysis functions that you can apply to an image.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that control specific plug-in computer devices. Instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW.
interlaced	A video frame composed of two interleaved fields. The number of lines in a field are half the number of lines in an interlaced frame.
interrupt	A computer signal indicating that the CPU should suspend its current task to service a designated activity.
IRE	A relative unit of measure (named for the Institute of Radio Engineers). 0 IRE corresponds to the blanking level of a video signal, 100 IRE to the white level. Notice that for CCIR/PAL video, the black level is equal to the blanking level or 0 IRE, while for RS-170/NTSC video, the black level is at 7.5 IRE.

L

line count	The total number of horizontal lines in the picture.
lossy compression	Compression in which the decompressed image is visually similar but not identical to the original image.
LSB	Least significant bit.
luma	The brightness information in the video picture. The luma signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.

luminance	*See* luma.
LUT	Lookup table. Table containing values used to transform the gray-level values of an image. For each gray-level value in the image, the corresponding new value is obtained from the lookup table.

M

MB	Megabyte.
memory buffer	*See* buffer.
memory window	Continuous blocks of memory that can be accessed quickly by changing addresses on the local processor.
method	In Visual Basic, a set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed.
MSB	Most significant bit.
mux	Multiplexer. A switching device with multiple inputs that selectively connects one of its inputs to its output.

N

NI-IMAQ	Driver software for National Instruments IMAQ hardware.
noninterlaced	A video frame where all the lines are scanned sequentially, instead of divided into two frames as in an interlaced video frame.
NTSC	National Television Standards Committee. The committee that developed the color video standard used primarily in North America, which uses 525 lines per frame. *See also* PAL.
NVRAM	Nonvolatile RAM. RAM that is not erased when a device loses power or is turned off.

O

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

P

PAL	Phase Alternation Line. One of the European video color standards. PAL uses 625 lines per frame. *See also* NTSC.
PCI	Peripheral Component Interconnect. A high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA. PCI offers a theoretical maximum transfer rate of 132 Mbytes/s.
PCLK	Pixel clock signal. Times the sampling of pixels on a video line.
picture aspect ratio	The ratio of the active pixel region to the active line region. For standard video signals like RS-170 or CCIR, the full-size picture aspect ratio normally is 4/3 (1.33).
pixel	Picture element. The smallest division that makes up the video scan line. For display on a computer monitor, the optimum dimension for each pixel is square (aspect ratio of 1:1, or the width equal to the height).
pixel aspect ratio	The ratio between the physical horizontal size and the vertical size of the region covered by the pixel. An acquired pixel should optimally be square, thus the optimal value is 1.0, but typically it falls between 0.95 and 1.05, depending on camera quality.
pixel clock	Divides the incoming horizontal video line into pixels.
pixel count	The total number of pixels between two horizontal synchronization signals (HSYNCs). The pixel count determines the frequency of the pixel clock.
pixel depth	The number of bits used to represent the gray level of a pixel.
PLL	Phase-locked loop. Circuitry that provides a very stable pixel clock that is referenced to another signal, such as an incoming horizontal synchronization signal (HSYNC).
protocol	The exact sequence of bits, characters, and control codes used to transfer data between computers and peripherals through a communications channel.
PXI	PCI eXtensions for Instrumentation. An open specification that builds on the CompactPCI specification by adding instrumentation-specific features.

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.
relative accuracy	A measure in LSB of the accuracy of an ADC; it includes all nonlinearity and quantization errors but does not include offset and gain errors of the circuitry feeding the ADC.
resolution	(1) The number of rows and columns of pixels. An image composed of m rows and n columns has a resolution of $m \times n$. This image has n pixels along its horizontal axis and m pixels along its vertical axis. (2) The smallest signal increment that can be detected by a measurement system. Resolution can be expressed in bits, proportions, or a percentage of full scale. For example, a system has 12-bit resolution, one part in 4,096 resolution, and 0.0244 percent of full scale.
RGB	Color encoding scheme using red, green, and blue (RGB) color information where each pixel in the color image is encoded using 32 bits: eight bits for red, eight bits for green, eight bits for blue, and eight bits for the alpha value (unused).
ring	Performs an acquisition that loops continually on a specified number of buffers.
ROI	Region of interest. (1) An area of the image that is graphically selected from a window displaying the image. This area can be used focus further processing. (2) A hardware-programmable rectangular portion of the acquisition window.
RS-170	The U.S. standard used for black-and-white television.
RT Series PXI controller	National Instruments PXI controller running the RT Engine.
RT Series PXI system	National Instruments PXI chassis with an RT Series PXI controller installed.
RTSI bus	Real-Time System Integration Bus. The National Instruments timing bus that connects IMAQ and DAQ devices directly by means of connectors on top of the devices for precise synchronization of functions.

S

saturation	The amount of white added to a pure color. Saturation relates to the richness of a color. A saturation of zero corresponds to a pure color with no white added. Pink is a red with low saturation.
scaling down circuitry	Circuitry that scales down the resolution of a video signal.
sequence	Performs an acquisition that acquires a specified number of buffers, then stops.
signal conditioning	The manipulation of signals to prepare them for digitizing.
snap	Acquires a single frame or field to a buffer.
StillColor	A post-processing algorithm that allows the acquisition of high-quality color images generated either by an RGB or composite (NTSC or PAL) camera using a monochrome video acquisition device.
sync	Tells the display where to put a video picture. The horizontal sync indicates the picture's left-to-right placement and the vertical sync indicates top-to-bottom placement.
synchronous	Property or operation that begins an operation and returns control to the program only when the operation is complete.

T

transfer rate	The rate, measured in bytes/s, at which data is moved from source to destination after software initialization and set up operations. The maximum rate at which the hardware can operate.
time-bounded	Characteristic associated with an RT event assuring that the execution time of the event or task is predictable.
trigger	Any event that causes or starts some form of data capture.
trigger control and mapping circuitry	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 or falling edge.
TTL	Transistor-transistor logic.

U

UV plane	*See* YUV.

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.
video line	A video line consists of a horizontal synchronization signal, back porch, active pixel region, and a front porch.
VSYNC	Vertical synchronization signal. The synchronization pulse generated at the beginning of each video field that tells the video monitor when to start a new field.

W

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

Y

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

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W