NI-IMAQdx™

NI-IMAQdx User Manual

Image Acquisition Software

February 2007
371970B-01

Important Information

Conventions

The following conventions are used in this manual:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** 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. Italic text also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
`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.
`monospace italic`	Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Contents

Chapter 3
Advanced Programming with NI-IMAQdx

Chapter 4
Using NI-IMAQdx in LabVIEW

1 Introduction to NI-IMAQdx

This chapter describes the NI-IMAQdx driver software, lists the supported application development environments (ADEs), describes the fundamentals of creating applications using NI-IMAQdx, describes the files used to build these applications, and explains where to find sample programs.

About the NI-IMAQdx Software

NI-IMAQdx gives you the ability to use GigE Vision cameras and IEEE 1394 industrial digital video cameras to acquire images. You can use cameras with the following output formats:

- Monochrome (8–16 bits/pixel)
- RGB (24–48 bits/pixel)
- YUV 4:1:1 (12 bits/pixel)
- YUV 4:2:2 (16 bits/pixel)
- YUV 4:4:4 (24 bits/pixel)
- Bayer (8–16 bits/pixel)

The cameras may operate at various resolutions and frame rates, depending on camera capabilities.

NI-IMAQdx complies with the Automated Imaging Association GigE Vision specification and the 1394 Trade Association Industrial and Instrumentation specification for Digital Cameras (IIDC), and controls all available modes and features of the digital camera.

Note Refer to the *NI Vision Acquisition Software Release Notes* for the specific version of the IIDC specification or the GigE Vision specification to which this driver complies.

Application Development Environments

This release of NI-IMAQdx supports the following ADEs for Windows Vista/XP/2000:

- LabVIEW version 7.1.1 and later
- LabVIEW Real-Time Module version 7.1.1 and later
- LabWindows™/CVI™ version 7.0 and later
- Microsoft Visual C/C++ version 6.0 and later
- Microsoft Visual Basic version 6.0 and later
- Microsoft Visual Studio .NET 2003 and later

Note Although the NI-IMAQdx software has been tested and found to work with these ADEs, other ADEs may also work.

Configuring Your Camera

Use National Instruments Measurement & Automation Explorer (MAX) to configure your camera. Refer to the *Measurement & Automation Explorer Help for NI-IMAQdx* for information about configuring your camera. You can access the *Measurement & Automation Explorer Help for NI-IMAQdx* from within MAX by going to **Help»Help Topics»NI-IMAQdx**.

The camera configuration is saved in a camera file, which the NI-IMAQdx VIs and functions use to configure a camera and supported attributes.

Fundamentals of Building Applications with NI-IMAQdx

Architecture

Figure 1-1 illustrates the NI-IMAQdx driver architecture.

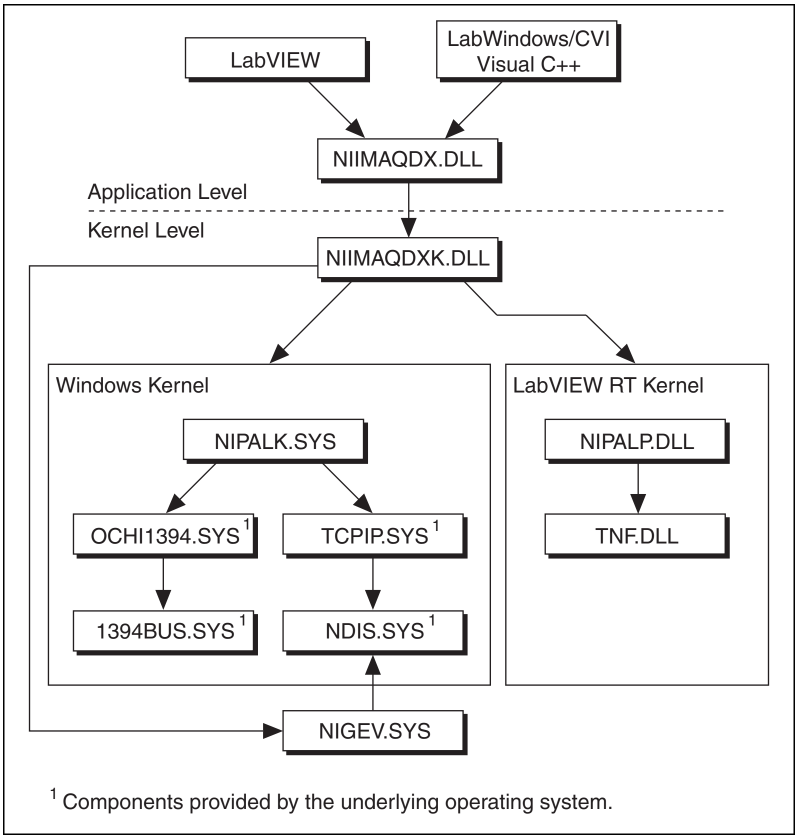

Figure 1-1. NI-IMAQdx Architecture

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 run your application on different operating systems and use updated versions of the driver without having to recompile your application.

NI-IMAQdx Libraries

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

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

Example Programs

You can find NI-IMAQdx code examples in the following directories.

Note If you installed NI-IMAQdx in the default location, you can find the following example directories within `C:\Program Files\National Instruments`.

- LabVIEW—`LabVIEW\examples\imaq`. For a brief description of any example VI, open the VI, and select **Windows»Show VI Info** for a text description of the example.

Tip You can access the NI-IMAQdx examples from the NI Example Finder. From LabVIEW, go to **Help»Find Examples** to launch the NI Example Finder.

- CVI—`CVI\samples\imaqdx`.
- C—`NI-IMAQdx\examples\MSVC`.
- Visual Basic—`NI-IMAQdx\examples\VB`.
- Microsoft Visual Studio .NET 2003—`NI-IMAQdx\examples\MSVB.NET`. The .NET examples are converted from the NI-IMAQdx for Visual Basic examples. The .NET examples are written in Visual Basic .NET and demonstrate use of the NI-IMAQdx 3.0 Assemblies and the IMAQ Vision 8.2 Viewer control.

Refer to the `readme.rtf` file located in your target installation directory for the latest details about the example programs.

2 Basic Acquisition with NI-IMAQdx

This chapter contains an overview of the NI-IMAQdx library, a description of the acquisition flow of NI-IMAQdx, and generic programming examples. The chapter also contains flowcharts of high-level and low-level *snap*, *grab*, and *sequence* operations.

Introduction

The NI-IMAQdx application programming interface (API) is divided into two main function groups: high-level and low-level.

- High-level functions—Use to capture images quickly and easily. If you need more advanced functionality, you can mix high-level functions with low-level functions.
  - Snap functions—Capture all or a portion of a single image to the *user buffer*.
  - Grab functions—Perform an acquisition that loops continually on one or more *internal buffers*. You can copy the last acquired buffer to a separate user buffer for processing or analysis.
  - Sequence functions—Acquire a specified number of internal buffers and then stops.
- Low-level functions—Use when you require more direct control of the image acquisition.
  - Acquisition functions—Configure, start, stop, and unconfigure an image acquisition, or examine a user buffer during an acquisition.
  - Attribute functions—Examine and change the acquisition or camera attributes.
  - Event functions—Register events.
  - Register functions—Access registers.
  - Session functions—Open, close, or enumerate sessions.
  - Utility functions—Get detailed error information.

Both high-level and low-level functions support snap, grab, sequence, and triggered acquisitions. Using 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 to use these functions.

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

Acquisition Flow

This section describes the basic steps of performing an acquisition with the NI-IMAQdx software. The basic steps are initialization, configuration, and acquisition.

Opening a Camera

To acquire images using the high-level or low-level functions, you first must open a *camera session*. A camera session is a *process-safe handle* to a camera. The driver uses a camera session to identify the camera to which further NI-IMAQdx functions apply. You can simultaneously open as many camera sessions as there are cameras connected to your system.

When opening the camera session, you need to specify two parameters: camera name and camera control mode. Refer to the following sections for detailed information about these parameters. When an application is finished with the camera, call the Close Camera function to close the camera session.

Camera Name

NI-IMAQdx references all camera sessions by a name. The driver creates default names for each camera in your system in the order that the cameras are connected. The names observe the convention shown in Table 2-1.

Table 2-1. Camera Naming Convention

Camera Name	Camera Installed
`cam0`	Device 0
`cam1`	Device 1
...	...
`cam`*n*	Device *n*

Every camera has an `.iid` interface file and an `.icd` camera file.

- Interface files—Store information about which physical camera is associated with a camera name. Each interface file can be used by only a single camera.
- Camera files—Store all the configurable attributes. Camera files can be shared between identical cameras. Use MAX to configure the default state of a particular camera.

Figure 2-1 shows the relationship between cameras, interface files, and camera files.

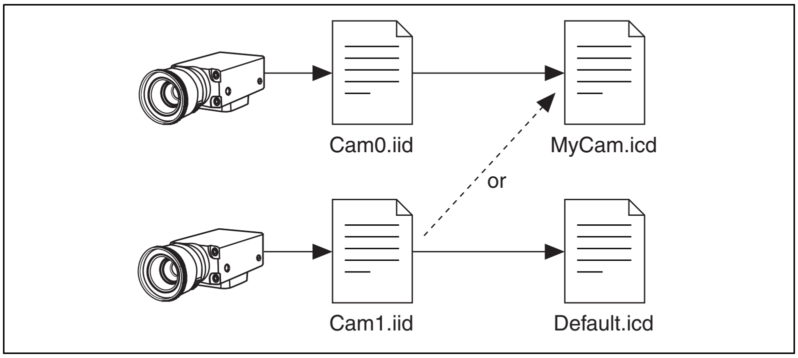

Figure 2-1. Relationship Between Cameras, Interface Files, and Camera Files

Note Use the Enumerate function to query the number and names of available cameras. Use the Discover Ethernet Cameras function to find ethernet cameras on the network with a remote subnet.

When you open a camera session with the Camera Open function, the camera with the unique serial number described by the interface file `cam`*n*`.iid` opens, where *n* is the reference to the camera. If the camera is not present and a camera of the same make and model is present, as described in the interface file, the driver opens the available camera. The interface file updates to use the new camera. The camera file described by the interface file opens, and all the user attributes are set in the driver. If no camera of the same make and model is present, the Camera Open function returns an error.

Camera Control Mode

The camera control mode parameter has two options: controller and listener. The default option—controller—controls the camera and receives video data. The listener only receives video data. Use the listener option in broadcasting applications. Refer to the *Broadcasting* section of Chapter 3, *Advanced Programming with NI-IMAQdx*, for more information about broadcasting.

Configure the Acquisition

After opening the camera, configure the camera for acquisition by specifying the following parameters: whether the acquisition is one-shot or continuous, and the number of internal buffers to use.

While configuring the camera, the driver validates all the user-configurable attributes. If any attributes are invalid or out of range, the driver returns an error and does not configure the acquisition.

If you want to reconfigure the acquisition, call the Unconfigure Acquisition function before calling the Configure function again.

Note National Instruments recommends that you do *not* configure an acquisition in a loop because doing so is time-intensive.

One-Shot/Continuous Acquisition

Use a one-shot acquisition to start an acquisition, perform the acquisition, and stop the acquisition using a single function. The number of images acquired is equal to the number of images in the images collection.

With a one-shot acquisition, you specify a certain number of internal buffers. The camera transfers each image up to and including the specified number of buffers. The driver acquires every image during a one-shot acquisition. National Instruments recommends one-shot acquisition for applications that do not require real-time acquisition or processing.

Use a continuous acquisition to start an acquisition, continuously acquire images into the internal buffers, and explicitly stop the acquisition. With continuous acquisition, the driver acquires video data continuously from the camera and enables you to examine the most current buffer. National Instruments recommends continuous acquisition for real-time acquisition and processing.

Note If CPU activity increases during a continuous acquisition, the driver might miss subsequent images. Check the buffer number output to determine if you have missed any images.

Number of Buffers

Another aspect of configuration is specifying the number of internal buffers into which you want to acquire image data. During configuration, buffers are allocated from system memory and page-locked. Once the acquisition starts, the camera transfers video data over the IEEE 1394 bus to the IEEE 1394 interface card *FIFO*. Then, video data is directly transferred to the internal buffer. This transfer requires negligible CPU resources. For the GigE Vision bus, CPU resources are used to pass network packets. For ethernet, ethernet packets are evaluated by software and copied into an internal buffer.

Each internal buffer you allocate is the exact size of the raw data being transmitted by the camera. For continuous acquisitions, allocate three or more buffers. Allocating a single buffer for a continuous acquisition may result in a high number of lost images. For one-shot acquisitions, specify the number of buffers that the application requires. For example, if the application runs for two seconds, and the camera acquires at 30 frames per second, allocate 60 buffers to capture each image.

Having more buffers available for the GigE Vision bus can increase the reliability of data transfer, especially when there are adverse network conditions.

Region of Interest

The region of interest (*ROI*) specifies a rectangular portion of the image to be captured. If the camera supports scalable video modes, the ROI defines the portion of the image to transfer from the camera to system memory. If the camera does not support scalable video modes, the entire image is transferred from the camera to system memory. In all video modes, the ROI specifies the amount of data decoded by the driver while acquiring into a user buffer.

By default, the driver transfers the entire image. Specify a smaller ROI for the following reasons:

- To acquire only the necessary subset of data
- To increase the acquisition speed by reducing the amount of data transferred and/or decoded
- To allow for multiple simultaneous acquisitions by reducing bandwidth usage

Note Although you can specify an ROI of any size, the NI-IMAQdx software coerces the ROI into one that is more compatible for the given camera. Refer to Chapter 3, *Advanced Programming with NI-IMAQdx*, for more information about defining an ROI for scalable images.

Pixel Format

The pixel format specifies the source type of the image format. Different pixel formats decode into different image types. Refer to the *Decoding* section for more information.

Acquisition

After configuring and starting your acquisition, the camera sends data to the internal buffers. To process the acquired image data, you must copy the data from the internal buffer into your user buffer.

User Buffer

Before starting the acquisition, you must allocate a user buffer in addition to configuring internal buffers. The driver copies or decodes image data from the internal buffer into the user buffer during acquisition. Then, process and analyze the image in the user buffer.

When acquiring data into an image, the driver resizes and casts the image as needed. However, if you acquire data into a user buffer, you must allocate enough space for one decoded image.

Note Unlike internal buffers, you are responsible for destroying user buffers.

Buffer Number Mode

Specify one of the following options for the buffer number mode.

- Buffer Number—Gets the exact buffer number specified in the Buffer Number parameter.
- Last—Gets the most recently acquired buffer.
- Next—Gets the next incoming buffer.

Buffer Number

A buffer number is a zero-based index that represents the cumulated transferred image count. For example, during a continuous acquisition with three internal buffers, the buffer number is updated as follows: 0, 1, 2, 3, 4, 5, and so on. Buffer numbers 0 and 3 refer to the same internal buffer in the buffer ring.

For a one-shot acquisition, you can request only one of the available buffer numbers. For a continuous acquisition, you can request any present or future buffer number. You can also request the next logical buffer or the buffer containing the most recently acquired data. With high-level grab acquisitions, the buffer number defaults to the next transferred buffer.

When you complete the buffer acquisition step, the driver returns the actual buffer number with the image.

Overwrite Mode

Ideally, a continuous acquisition acquires and processes every image that is transferred from the camera. However, because of processing time fluctuations, some images from the camera may not be processed before the camera transfers the next image. Using multiple internal buffers in a continuous acquisition allows for a small amount of jitter. However, if a delay becomes too long, the camera overwrites the requested buffer with new image data.

NI-IMAQdx is able to detect overwritten internal buffers. You can configure the driver to manage an overwritten buffer in one of the following ways:

- Get newest valid buffer
- Get oldest valid buffer
- Fail and return an error

In all cases, the camera continues to transfer data when a buffer is overwritten.

The default overwrite mode for all types of acquisition is to get the newest valid buffer. This option, which National Instruments recommends for most applications, enables you to process the most recent image. If you need to get the image closest in time to a requested buffer, configure the driver to get the oldest valid buffer. If your application requires that every image be processed, configure the driver to fail when a buffer is overwritten so that you are alerted.

Timeouts

A *timeout* is the length of time, in milliseconds, that the driver waits for an image from the camera before returning an error. A timeout error usually occurs if the camera has been removed from the system or when the camera did not receive an external trigger signal.

Decoding

Except for 8-bit monochrome images, all video modes require decoding before you can interpret the image data. For example, many color cameras output images of type YUV 4:2:2. However, NI Vision does not natively support the YUV mode. To process and display the image, the driver automatically decodes the YUV image into a 32-bit RGB image.

Table 2-2 lists common video modes and their corresponding image types after being decoded by NI-IMAQdx.

Table 2-2. Decoder Inputs and Corresponding Outputs

Raw Camera Output	Decoded Destination Image Type
8-bit monochrome	Image_U8
10–16-bit monochrome	Image_I16
YUV 4:1:1	Image_RGB
YUV 4:2:2	Image_RGB
YUV 4:4:4	Image_RGB
24-bit RGB	Image_RGB
30–48-bit RGB	Image_RGB_U64
8-bit Bayer	Image_RGB
10–16-bit Bayer	Image_RGB

Decoding images requires CPU resources. However, many of the decoding algorithms have been optimized in the driver. If you do not want decoded image data, you can use NI-IMAQdx to get a copy of the raw camera output.

Programming Examples

This section contains examples of high-level and low-level image acquisitions. Refer to the *Example Programs* section of Chapter 1, *Introduction to NI-IMAQdx*, for directory paths to the code examples discussed in this section.

High-Level Function Examples

Use high-level functions to write programs quickly without having to learn the details of the low-level API and driver.

Snap

A snap acquires a single image into a user buffer. Figure 2-2 illustrates the typical programming order of a high-level snap acquisition.

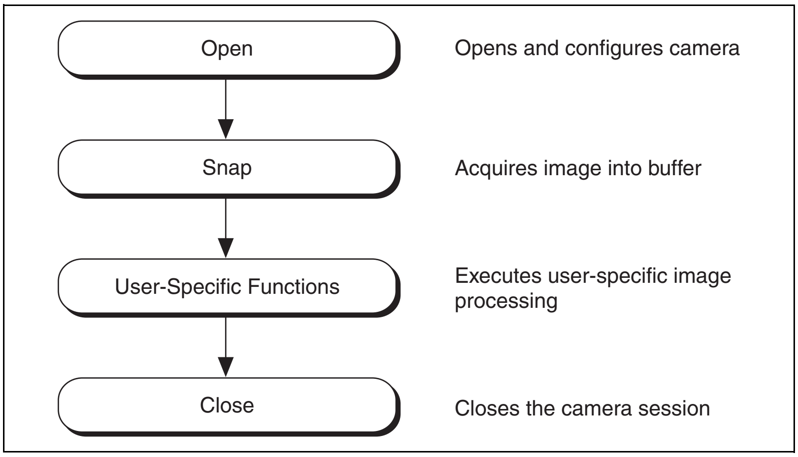
Open
Opens and configures camera
Snap
Acquires image into buffer
User-Specific Functions
Executes user-specific image processing
Close
Closes the camera session

Figure 2-2. High-Level Snap Flowchart

Use a snap for low-speed or one-shot applications where ease of programming is essential. When you invoke a snap, the driver opens a session on a camera and initializes the camera. Opening a session sets the ROI to the size of the settings you configured in MAX.

Note If you do not have a valid session, a temporary session is created using `cam0`.

Then, the snap acquires the next incoming image into a user buffer. After the image is acquired, the program calls image processing and analysis functions. When the processing and analysis functions are finished, the program calls the Close Camera function using the camera handle. This function instructs NI-IMAQdx to free all of the resources associated with this camera, which releases the session.

Grab

A grab initiates a continuous high-speed acquisition of images to one or more internal buffers. Figure 2-3 illustrates the typical programming order of a high-level grab acquisition.

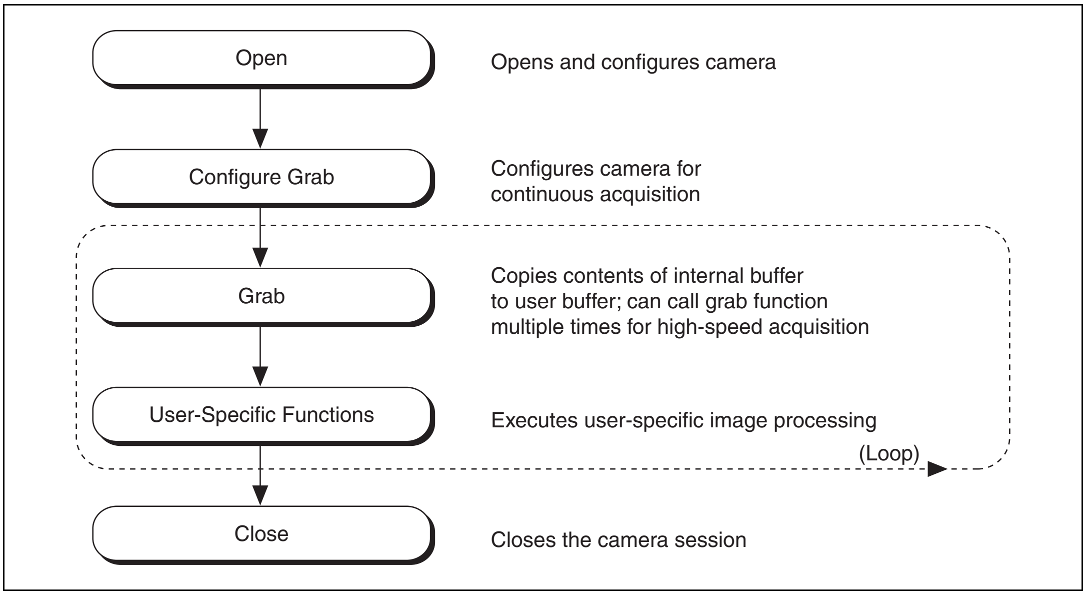
Open
Opens and configures camera
Configure Grab
Configures camera for continuous acquisition
Grab
Copies contents of internal buffer to user buffer; can call grab function multiple times for high-speed acquisition
User-Specific Functions
Executes user-specific image processing
(Loop)
Close
Closes the camera session

Figure 2-3. High-Level Grab Flowchart

Use a grab for high-speed applications during which you need to process only one image at a time. You can copy the last acquired buffer to a separate user buffer for processing or analysis. To use these functions, you must have a valid session. If you do not have a valid session, the NI-IMAQdx Configure Grab function creates a session using `cam0`.

Calling the Configure Grab function opens a session for a grab acquisition. During acquisition, each successive grab copies the last acquired internal buffer into a user buffer where you can process the image.

Sequence

A sequence acquires a specified number of internal buffers and then stops. Figure 2-4 illustrates the typical programming order of a high-level sequence acquisition.

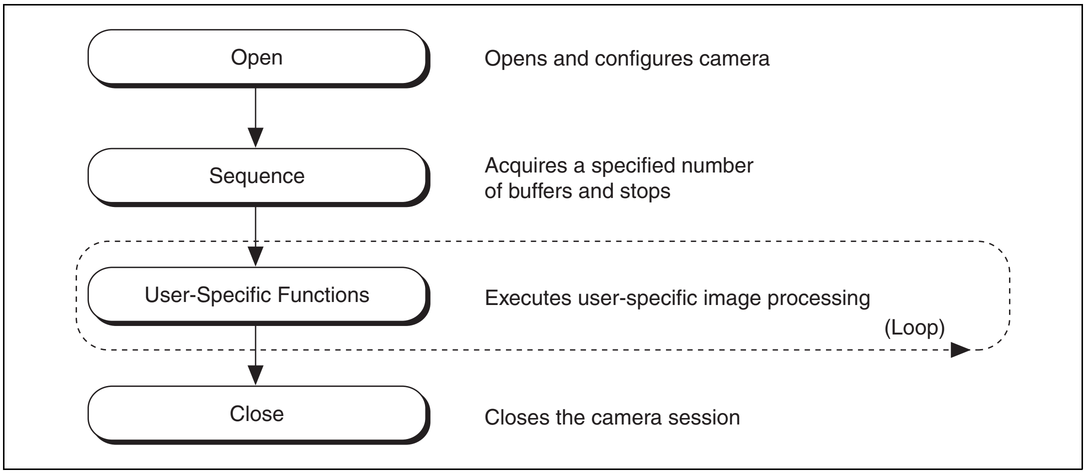
Open
Opens and configures camera
Sequence
Acquires a specified number of buffers and stops
User-Specific Functions
Executes user-specific image processing
(Loop)
Close
Closes the camera session

Figure 2-4. High-Level Sequence Flowchart

Use a sequence in applications where you need to process a series of consecutive images. Sequence acquisitions are synchronous. If you do not specify a session, a temporary session is created using `cam0`.

Low-Level Function Examples

Use low-level functions for more advanced programming techniques. In general, low-level functions have more parameters than high-level functions.

Snap

The low-level snap examples set up a one-shot, single-image acquisition and start the acquisition. The program acquires an image and processes it. Finally, the program stops the acquisition, unconfigures the acquisition, and closes the session.

Figure 2-5 illustrates the programming order of a low-level snap acquisition.

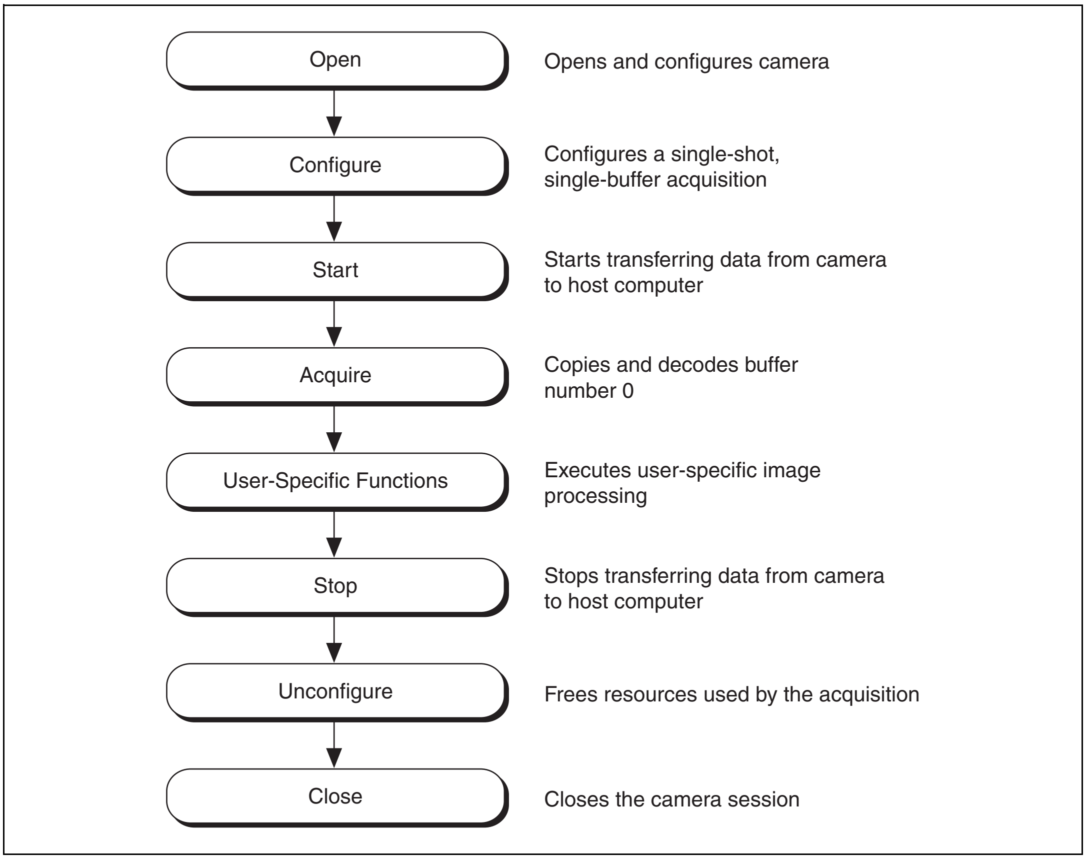
Open
Opens and configures camera
Configure
Configures a single-shot, single-buffer acquisition
Start
Starts transferring data from camera to host computer
Acquire
Copies and decodes buffer number 0
User-Specific Functions
Executes user-specific image processing
Stop
Stops transferring data from camera to host computer
Unconfigure
Frees resources used by the acquisition
Close
Closes the camera session

Figure 2-5. Low-Level Snap Flowchart

Grab

The low-level grab examples demonstrate how to perform a grab acquisition using low-level function calls. The program sets up a continuous acquisition into three internal buffers and starts the acquisition. The main loop iterates continuously. In the main processing loop, the program acquires an image and processes it. After the loop, the program stops the acquisition, unconfigures the acquisition, and closes the session.

Figure 2-6 illustrates the programming order of a low-level grab acquisition.

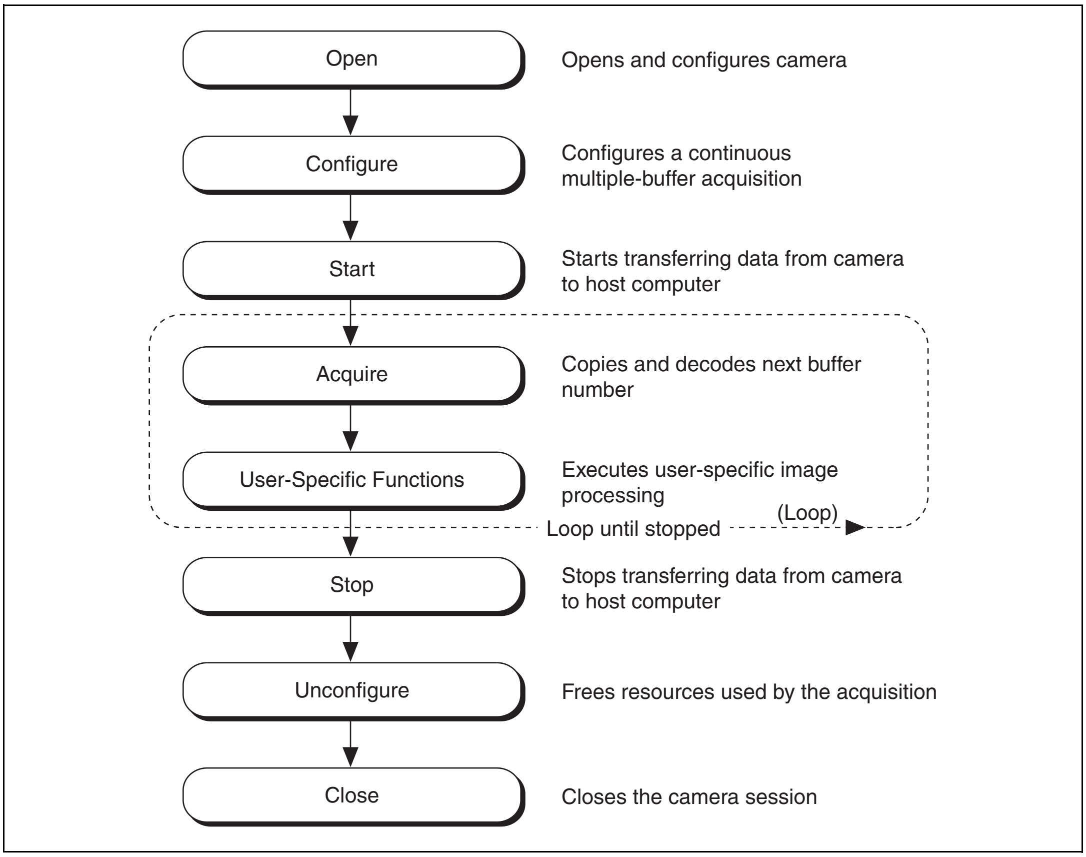
Open
Opens and configures camera
Configure
Configures a continuous multiple-buffer acquisition
Start
Starts transferring data from camera to host computer
Acquire
Copies and decodes next buffer number
User-Specific Functions
Executes user-specific image processing
(Loop)
Loop until stopped
Stop
Stops transferring data from camera to host computer
Unconfigure
Frees resources used by the acquisition
Close
Closes the camera session

Figure 2-6. Low-Level Grab Flowchart

Sequence

The low-level sequence examples demonstrate how to perform a sequence acquisition using low-level calls. The program sets up a one-shot, multi-image acquisition and starts the acquisition. The main loop iterates once for each internal buffer. In the main processing loop, the program acquires an image and processes it. After the loop, the program stops the acquisition, unconfigures the acquisition, and closes the session.

Figure 2-7 illustrates the programming order of a low-level sequence acquisition.

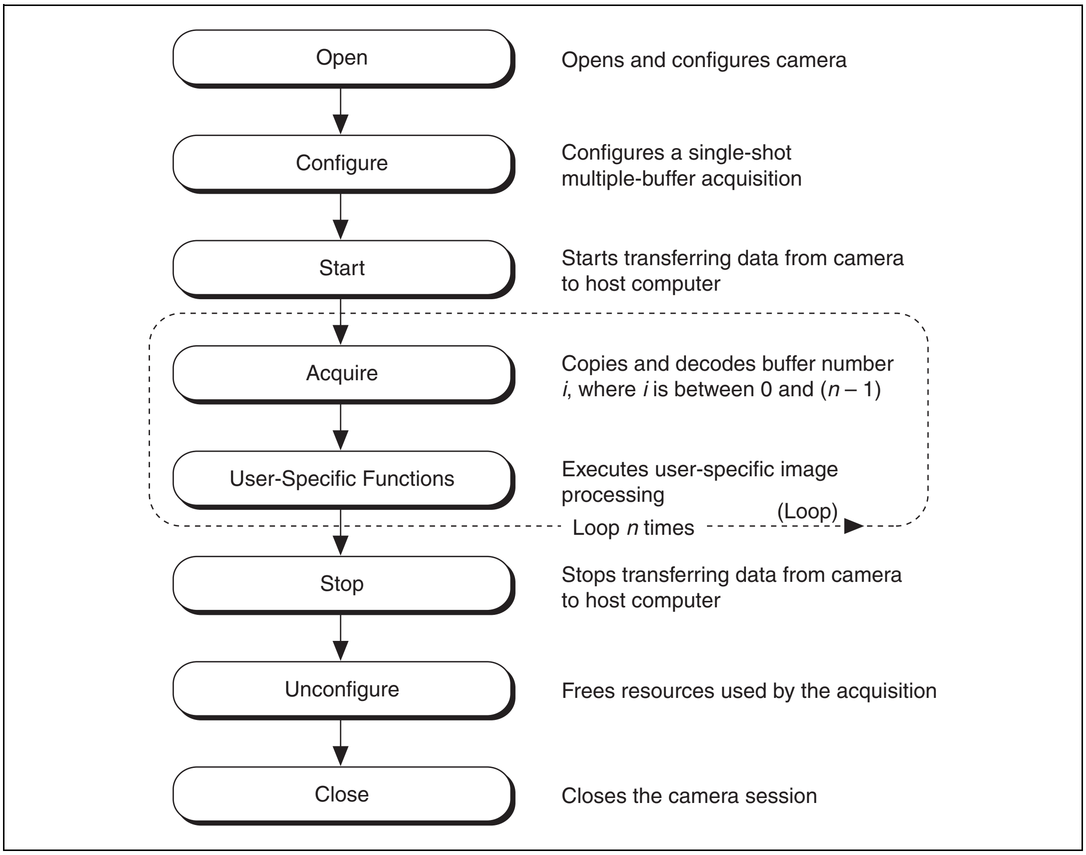
Open
Opens and configures camera
Configure
Configures a single-shot multiple-buffer acquisition
Start
Starts transferring data from camera to host computer
Acquire
Copies and decodes buffer number *i*, where *i* is between 0 and (*n* – 1)
User-Specific Functions
Executes user-specific image processing
(Loop)
Loop *n* times
Stop
Stops transferring data from camera to host computer
Unconfigure
Frees resources used by the acquisition
Close
Closes the camera session

Figure 2-7. Low-Level Sequence Flowchart

3 Advanced Programming with NI-IMAQdx

This chapter contains information about setting camera attributes, broadcasting acquired images to multiple machines, using scale to define the size of transferred images, and triggering.

Camera Attributes

After opening a camera, configure the camera attributes by specifying the following parameters: the attribute name, the attribute type, and the attribute value. Specify the same attributes when querying the camera attributes.

Use the **Set Attribute** function to set an attribute value. Use the **Get Attribute** function to get an attribute value.

Note To configure your camera in LabVIEW, use the IMAQdx Property Node.

The driver returns an error if an attribute is not accessible. Query the attribute access before accessing an attribute. When setting an attribute, the driver returns an error if the value is out of range. Query the attribute range before setting an attribute.

Attribute Name

The attribute name is a string constant. Each camera has a different set of attributes. Use the **Enumerate Attributes** function to list all available attributes for a given camera. The attribute name contains several keywords separated by a double colon namespace marker. AcquisitionAttributes::Timeout and AcquisitionAttributes::Bayer::Pattern are two examples of attribute names.

The namespace marker separates different levels in the attribute tree as described by the driver and the camera. Use the fully qualified attribute name such as AcquisitionAttributes::Timeout, or use the short version of the attribute name such as Timeout, when specifying the attribute name.

Attribute Type

The attribute type determines how a camera attribute is stored in the driver and the camera. Use the **Get Attribute Type** function to query the type of a given attribute. Each attribute is represented as one of the following types.

Table 3-1. Attribute Types and Descriptions

Attribute Type	Description
U32	32-bit unsigned integer
I64	64-bit signed integer
F64	Double precision floating point
String	String
Enum	Name and value pair
Bool	Boolean (true or false)
Command	Action

Attribute Value

The attribute value represents the active value of an attribute in the driver and the camera. The value type must be compatible with the attribute type. The following value types are supported.

Table 3-2. Value Types and Descriptions

Value Type	Description
U32	32-bit unsigned integer
I64	64-bit signed integer
F64	Double precision floating point
String	String
EnumItem	Name and value pair
Bool	Boolean (true or false)

The following value types are compatible for any given attribute type.

Table 3-3. Attribute Types and Compatible Value Types

Attribute Type	Compatible Value Types
U32	**U32**, I64, F64, String
I64	U32, **I64**, F64, String
F64	U32, I64, **F64**, String
String	**String**
Enum	U32, I64, F64, String, **EnumItem**
Bool	U32, I64, F64, String, **Bool**
Command	U32, I64, F64, String, **Bool**

The **bolded** compatible value type indicates the native value type. For example, use a 32-bit unsigned integer value type when dealing with a 32-bit unsigned integer attribute type. Optionally, use a signed 64-bit integer, double precision floating point, or string when dealing with a 32-bit unsigned integer attribute type.

Attribute Access

Each attribute has a read access and a write access. Use the **Is Attribute Readable** function to query read access for a given attribute. Use the **Is Attribute Writable** function to query write access for a given attribute. Some attributes are only settable before configuring the acquisition, while other attributes can be changed at any time.

Attribute Range

Most attributes have a specific range of valid values. Use the **Get Attribute Minimum**, **Get Attribute Maximum**, and **Get Attribute Increment** functions to query the range of numeric attribute types. Use the **Enumerate Attribute Values** function to query the range of enumerated types. String, bool, and command attributes do not have a range.

The following range is applicable for any given attribute.

Table 3-4. Attribute Types and Compatible Range

Attribute Type	Compatible Range
U32	Minimum, Maximum, Increment
I64	Minimum, Maximum, Increment
F64	Minimum, Maximum, Increment
String	N/A
Enum	Attribute Values
Bool	N/A
Command	N/A

Broadcasting

Many machine vision applications involve a single host computer acquiring data from a single industrial camera. Other machine vision applications acquire data from multiple industrial cameras concurrently. With the broadcasting feature, a machine vision application can run on multiple host computers while acquiring data from a single camera, as shown in Figure 3-1.

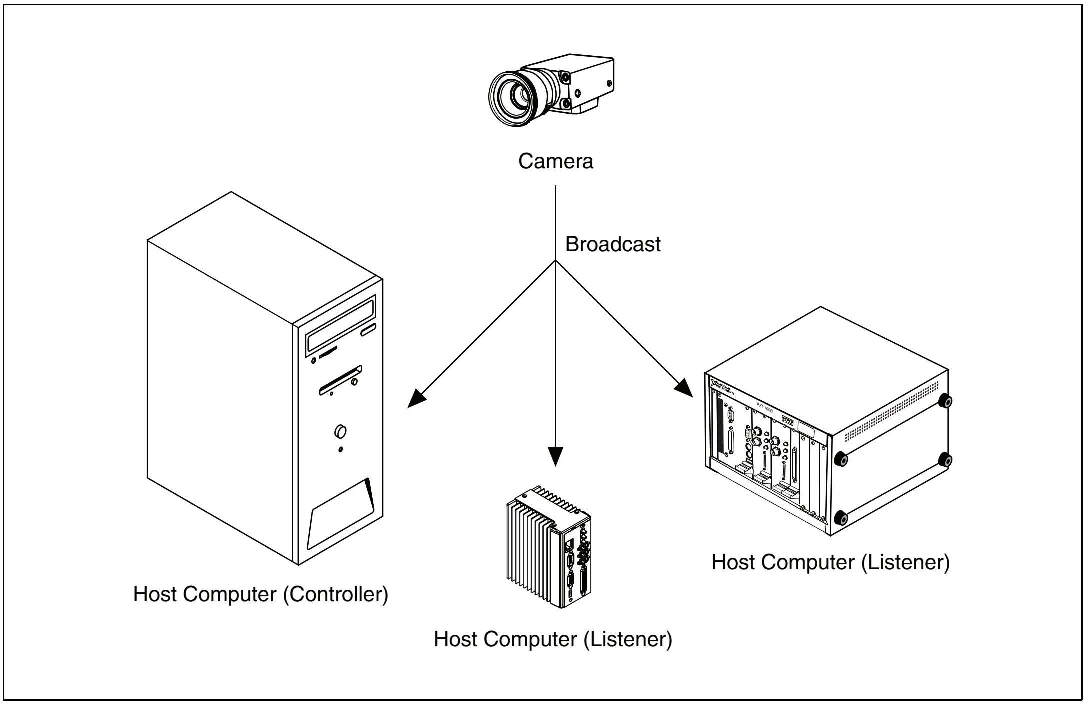

Figure 3-1. One Camera Broadcasting to Multiple Host Computers

The camera broadcasts video data on the camera bus and all the connected host computers receive the same image data. In this scenario, one host computer is designated as the controller. The controller is responsible for starting/stopping the camera feed. There can be only one controller per camera. The listeners obtain image data from the camera bus. The listeners do not control the camera in any way. There may be one or more listeners per camera.

Broadcasting has many uses. Computationaly intensive tasks can be spread across different machines, thus effectively distributing computations. Multiple host computers can also perform redundancy checks. Additionally, listeners can monitor the current status of a headless system.

Implementation

Usage for the controller is unchanged from a stand-alone application. Open your camera interface with the default interface name (for example, `cam0`) configured in MAX. Configure and start your acquisition.

For GigE Vision cameras, you can configure the camera to broadcast or multicast image data to all nodes on the network. Broadcast is not routable, and everyone on the same network sees the data, even if they are not listening. Multicast is routable if the network is configured properly. A multicast configuration is preferred if supported by the camera because only interested parties can see the data traffic.

Next, start the listener(s). On the listening computer, open your camera interface with the 64-bit unique identifier of the target camera, which you can find in the **General** tab in MAX. The controller can get a unique ID and send it to the listener sessions. Additionally, you must set the listener camera control mode parameter.

At this point, both the controller and listener systems are acquiring the same live data from the same camera. When running as a listener, most acquisition attributes are read-only. No camera feature or control is accessible when running as a listener system. Attempts to set these attributes result in the following error: **`Attribute not writable`**.

There is no synchronization between the controller and the listener host computers provided by the low-level driver. The user must start the controller before starting the listener. If the camera is not transmitting data when the listener initializes, the session returns the following error: **`No acquisition in progress`**. If the controller stops the video feed of the camera, the listener times out.

Scalable Image Size

Digital cameras support a predefined set of image sizes, which you can select through the **Acquisition** attributes in MAX. Refer to your camera documentation for a list of supported formats.

If you are using LabVIEW, the NI-IMAQdx software recognizes the predefined formats and automatically allocates enough memory to accommodate the image. The image is resized as necessary.

Some cameras support user-configurable ROI, which allows you to define the size of the acquired image. If you use this format, you must input the image size using the **Region of Interest** attributes—offset x, offset y, width, and height. The size and position of the sub-image you are acquiring must be a multiple of the **Increment** attribute, as shown in Figure 3-2, or the driver acquires the smallest sub-image that contains the ROI you defined.

The **Increment** values are camera-specific. Refer to the camera documentation or query the **Increment** attributes for width and height to obtain the actual values.

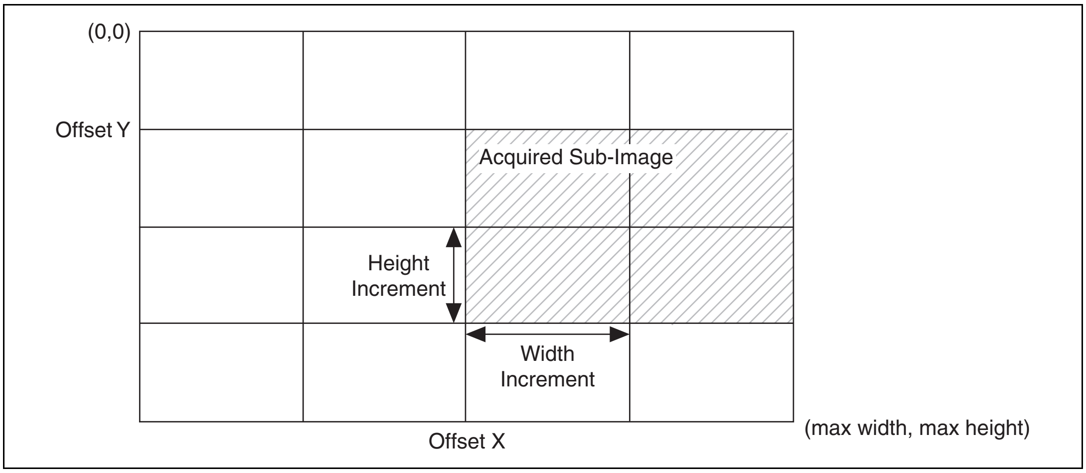

Figure 3-2. Partial Image Size Format (Scalable)

Trigger Modes

The IIDC 1.31 and GigE Vision 1.0 specifications provide several external triggering modes for cameras. A camera may support one or more of the triggering modes. Additionally, a camera may support triggering modes that are vendor specific. Refer to your camera documentation to find out which standard modes are supported.

Configure triggers before configuring and starting the acquisition. Use the **Camera Attribute** functions to configure the triggers.

Trigger Modes for IIDC Cameras

All IIDC cameras that support triggering have the following attributes:

- **TriggerMode**—Specifies one of the trigger modes—Mode0, Mode1...Mode5—described in the following sections. A value of **Off** indicates that the camera is not externally triggered.
- **TriggerActivation**—Specifies when the trigger input is active. A value of **LevelHigh** indicates that the trigger is considered active when the signal is high. A value of **LevelLow** indicated that the trigger is considered active when the signal is low.

- **TriggerParameter**—Certain trigger modes require an additional parameter. Refer to the following sections to see if the optional parameter is required.

Trigger Mode 0

With trigger mode 0, the camera starts frame integration when the external trigger input changes to an active value. The frame is exposed for a duration specified by the shutter attribute before the camera transfers the image to the host computer. No optional parameter is required.

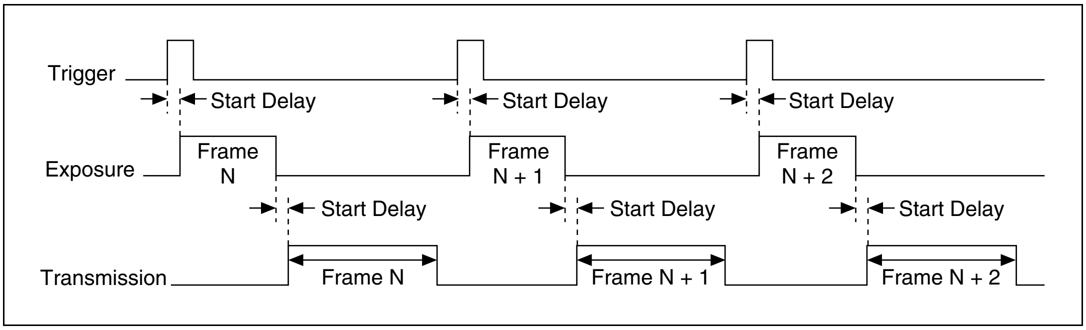

Figure 3-3. Timing Diagram for Trigger Mode 0

Trigger Mode 1

With trigger mode 1, the camera starts frame integration when the external trigger input changes to an active value. The frame is exposed while the external trigger is active. When the trigger becomes inactive, the camera stops frame integration and transfers the image to the host computer. No optional parameter is required.

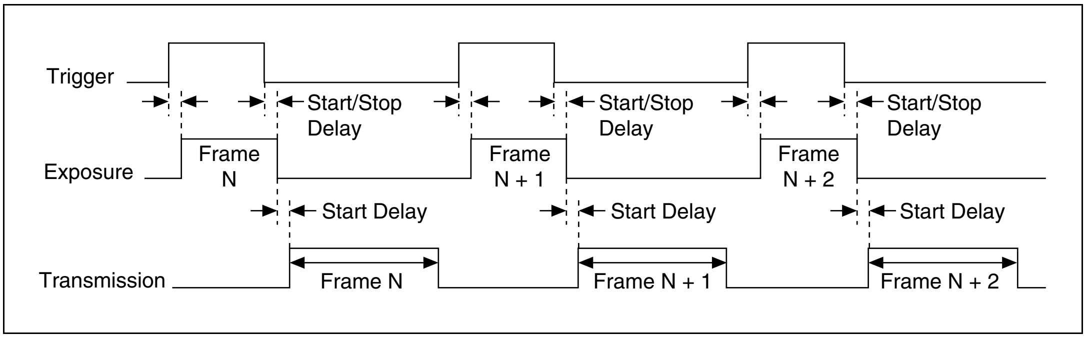

Figure 3-4. Timing Diagram for Trigger Mode 1

Trigger Mode 2

With trigger mode 2, the camera starts frame integration when the external trigger input changes to an active value. The same frame is exposed for multiple triggers. The number of triggers is specified by the optional parameter, which must have a value of 2 or more.

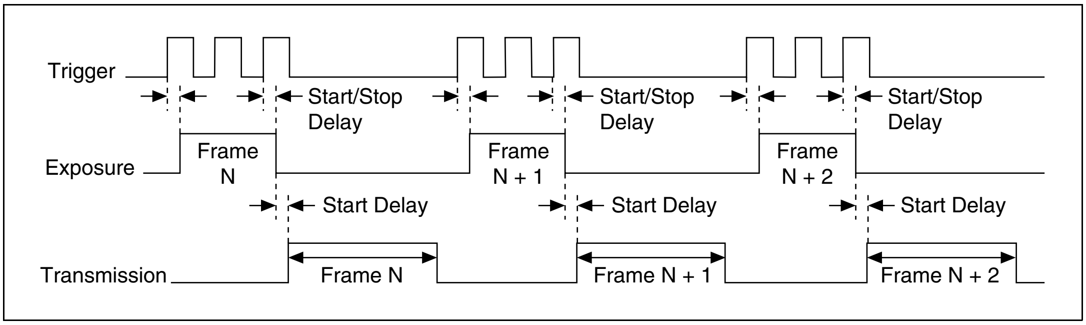

Figure 3-5. Timing Diagram for Trigger Mode 2

Trigger Mode 3

With trigger mode 3, the camera triggers continuously internally. The frame is exposed for a duration specified by the shutter attribute before the camera transfers the image to the host computer. The next internal trigger becomes active after a set cycle time. The cycle time is *N* times the cycle time of the fastest frame rate. *N* is specified by the optional parameter, which must have a value of 1 or more.

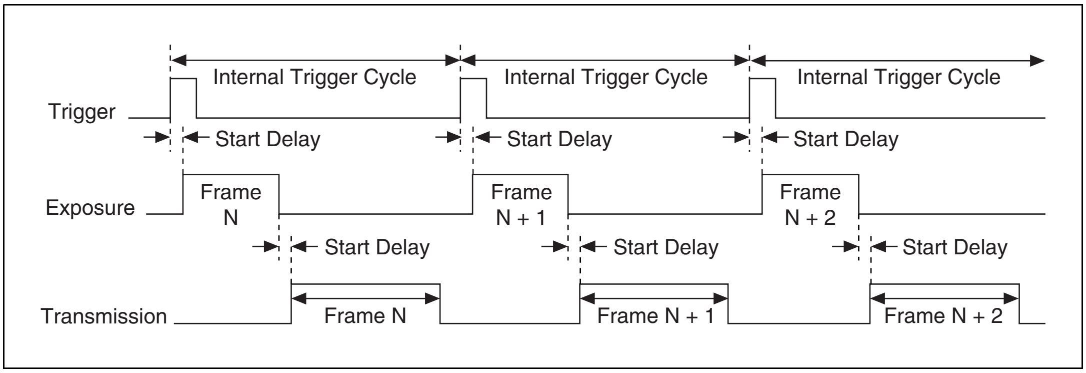

Figure 3-6. Timing Diagram for Trigger Mode 3

Trigger Mode 4

With trigger mode 4, the camera starts frame integration when the external trigger input changes to an active value. Multiple frames are exposed before the camera transfers the image to the host computer. Each frame is exposed for a duration specified by the shutter attribute. The number of frames is specified by the optional parameter, which must have a value of 1 or more.

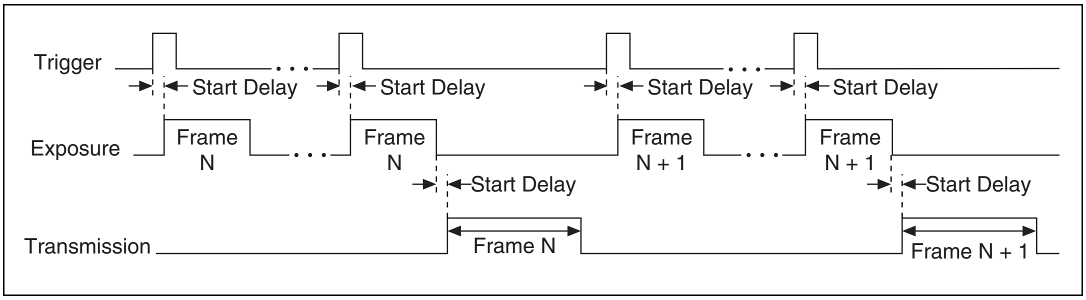

Figure 3-7. Timing Diagram for Trigger Mode 4

Trigger Mode 5

With trigger mode 5, the camera starts frame integration when the external trigger input changes to an active value. Multiple frames are exposed before the camera transfers the image to the host computer. Each frame is exposed while the external trigger is active. The number of frames is specified by the optional parameter, which must have a value of 1 or more.

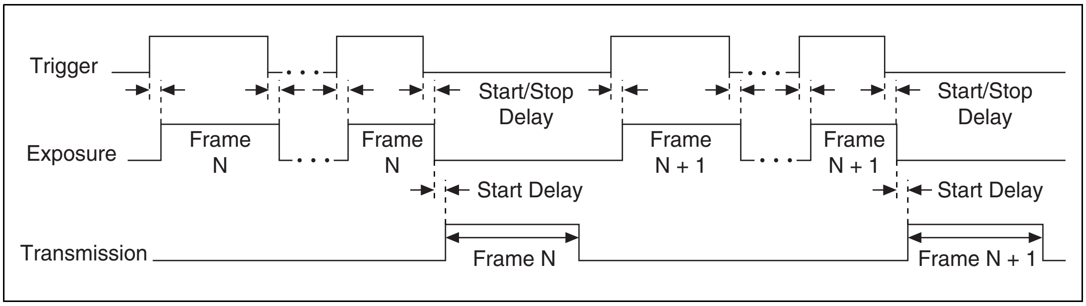

Figure 3-8. Timing Diagram for Trigger Mode 5

Trigger Modes for GigE Vision Cameras

Note All triggering modes and parameters for GigE Vision cameras are subject to camera vendor implementation. Refer to your camera documentation for triggering modes and parameters available for your camera.

Most GigE Vision cameras that support triggering have the following attributes:

- **TriggerSelector**—Specifies the type of trigger to control. Examples of triggers to control are FrameStart, FrameEnd, FrameActive, AcquisitionActive, LineStart, ExposureStart, ExposureEnd, ExposureActive.
- **TriggerMode**—Specifies the operation mode of the trigger for acquisition. Examples of operation modes are Off, Hardware, and Software.
- **TriggerActivation**—Specifies the type of signal change. Examples of signals are RisingEdge, FallingEdge, LevelHigh, and LevelLow.
- **TriggerSource**—Specifies the input line signal. Examples of input line signals are Off, Line1, and Line2.
- **TriggerSoftware**—Generates a software trigger to start the acquisition in software trigger mode.

4 Using NI-IMAQdx in LabVIEW

This chapter describes how to use NI-IMAQdx VIs in LabVIEW.

Introduction

The NI-IMAQdx VI library—part of the NI-IMAQdx software—is a group of virtual instruments (VIs) that enable you to use LabVIEW with your camera.

NI Vision for LabVIEW is the National Instruments image processing and analysis library, which consists of more than 400 VIs. Some of the basic NI Vision VIs are shared with NI-IMAQdx. If you do not have NI Vision, you can use the NI Vision VIs included with NI-IMAQdx to create an image acquisition application. When you use these basic VIs, you can upgrade your application later to use additional NI Vision VIs without making changes to your initial image acquisition application.

NI-IMAQdx adds a subpalette of VIs to the **Vision and Motion** Functions palette and an Image Display control to the Controls palette.

Create NI-IMAQdx applications as you would any other LabVIEW or LabVIEW Real-Time (RT) application. Drop icons onto the block diagram to create the program, and use the front panel to design the user interface. Click **Run** to compile and run the application.

Before you start building an image acquisition application, familiarize yourself with the basic knowledge and concepts contained in the following sections.

Location of the NI-IMAQdx VIs

You can find the NI-IMAQdx VIs in the LabVIEW **Functions** palette. From the LabVIEW block diagram, select **Vision and Motion» NI-IMAQdx**.

The most commonly used, high-level VIs are on the **NI-IMAQdx** palette. You can find VIs for basic acquisition and changing attributes.

The **Vision and Motion»NI-IMAQdx»NI-IMAQdx Low Level** palette contains VIs for more advanced applications.

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

Common VI Parameters

The following sections describe commonly used VIs and important parameters common to many VIs.

IMAQdx Session

IMAQdx Session is a unique identifier that specifies which interface file to use for the acquisition. The IMAQdx Session is produced by the IMAQdx Open Camera VI and used as an input to all other NI-IMAQdx VIs. The NI-IMAQdx VIs use IMAQdx Session Out, which is identical to IMAQdx Session, to simplify dataflow programming. IMAQdx Session Out is similar to the duplicate file sessions provided by the file I/O VIs. The high-level acquisition VIs—IMAQdx Snap, IMAQdx Configure Grab, and IMAQdx Sequence—require you to wire IMAQdx Session In only in the following instances:

- If you are using an interface other than the default `cam0`
- If you are using multiple cameras
- If you need to set IMAQdx properties before the acquisition

To get and set properties of the acquisition and camera, wire the IMAQdx Session to the LabVIEW property node.

Image Buffer

Many acquisition VIs require an image buffer to receive the captured image. You can create this image buffer with IMAQ Create. Refer to the *Buffer Management* section of this chapter for more information about using buffers. **Image In** receives the image buffer. **Image Out** returns the captured image.

Acquisition VIs

Two types of acquisition VIs are available in LabVIEW: high-level and low-level.

High-Level

Use the high-level acquisition VIs for basic image acquisition applications. VIs are included for snap, grab, and sequence, as described in the *Acquisition Types* section of this chapter.

Low-Level

Use the low-level acquisition VIs for more advanced image acquisition applications. The low-level VIs configure an acquisition, start an acquisition, retrieve the acquired images, and stop an acquisition. You can use these VIs to construct advanced Vision applications.

Complete the following general steps to perform a low-level acquisition.

1. Call IMAQdx Open Camera to initialize the board and create an IMAQdx Session.
2. Call IMAQdx Configure Acquisition to allocate resources for the acquisition.
3. Call IMAQdx Start Acquisition to start transferring data from the camera.
4. Call IMAQdx Get Image to obtain a copy of the requested image data.
5. After an acquisition, call IMAQdx Stop Acquisition to stop transferring data from the camera.
6. Call IMAQdx Unconfigure Acquisition to release the resources associated with the acquisition.
7. Call IMAQdx Close Camera to close the camera session.

Note If an acquisition is in progress and you call IMAQdx Close Camera, the driver automatically stops the acquisition and releases resources associated with the acquisition.

Buffer Management

The IMAQ Create VI and IMAQ Dispose VI manage image buffers in LabVIEW.

IMAQ Create, shown in Figure 4-1, 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 **Grayscale (U8)** for 8-bit monochrome images, **Grayscale (I16)** for 16-bit monochrome images, and **RGB (U32)** for RGB color images.

Note If **Image Type** is set to a value incompatible with the current video mode, NI-IMAQdx automatically changes the value to a compatible one when acquiring images.

New Image contains pointer information to 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 provide a value for **Border Size**. **Border Size** is the width, in pixels, of a border created around an image. Some image processing functions, such as labeling or morphology, require a border.

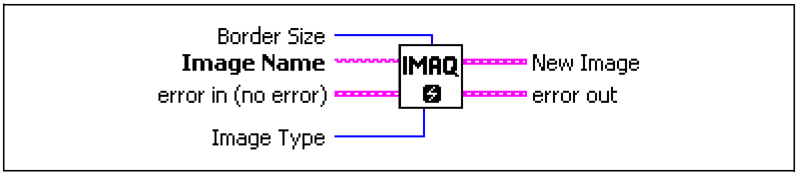

Figure 4-1. IMAQ Create

IMAQ Dispose, shown in Figure 4-2, frees the memory allocated for the image buffer. Call this VI only after the image is no longer required for processing.

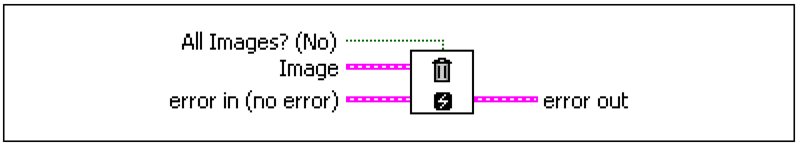

Figure 4-2. IMAQ Dispose

Acquisition Types

The following sections describe snap, grab, and sequence acquisitions in LabVIEW and give examples.

Snap

Use the IMAQdx Snap VI for snap applications. Figure 4-3 shows a simplified block diagram for using IMAQdx Snap.

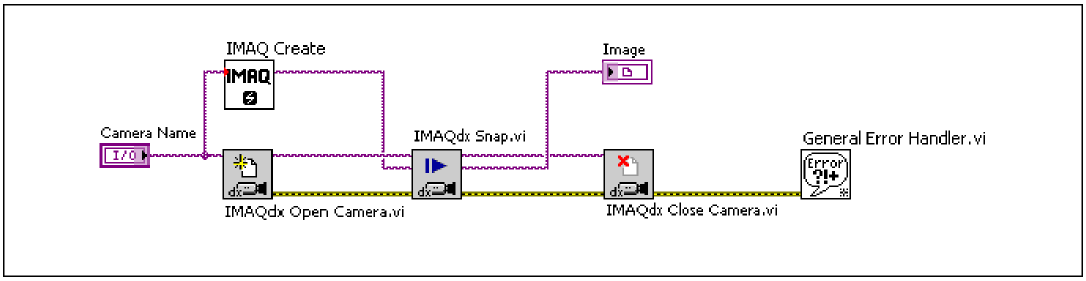

Figure 4-3. Acquiring an Image Using Snap

Grab

Use two VIs—IMAQdx Configure Grab and IMAQdx Grab—for a grab acquisition in LabVIEW. Call IMAQdx Configure Grab once to open the acquisition and start capturing the image to an internal software buffer. You can call IMAQdx Grab multiple times to copy the image currently stored in the internal buffer to a LabVIEW image buffer. After the program finishes copying images, call IMAQdx Close Camera once to shut down the acquisition.

Figure 4-4 shows a simplified block diagram for using IMAQdx Configure Grab and IMAQdx Grab.

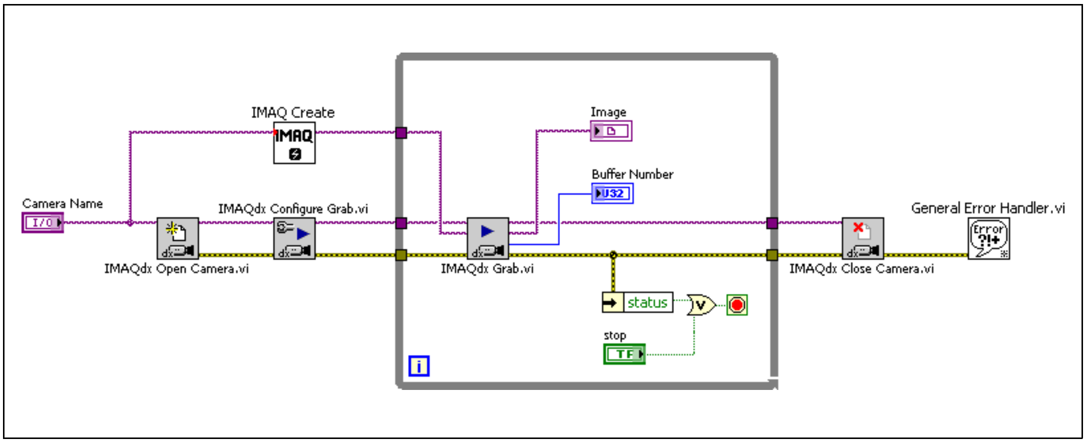

Figure 4-4. Acquiring Images Using Grab

Sequence

Use the IMAQdx Sequence VI for sequence applications. IMAQdx Sequence starts, acquires, and releases a sequence acquisition. IMAQdx Sequence does not return until the entire sequence is acquired.

Figure 4-5 shows a simplified block diagram for using IMAQdx Sequence. Place the IMAQ Create VI inside a For Loop to create an array of images for the **Image Array In** input to IMAQdx Sequence. The Number to Decimal String VI and Concatenate String VI create a unique name for each image in the array.

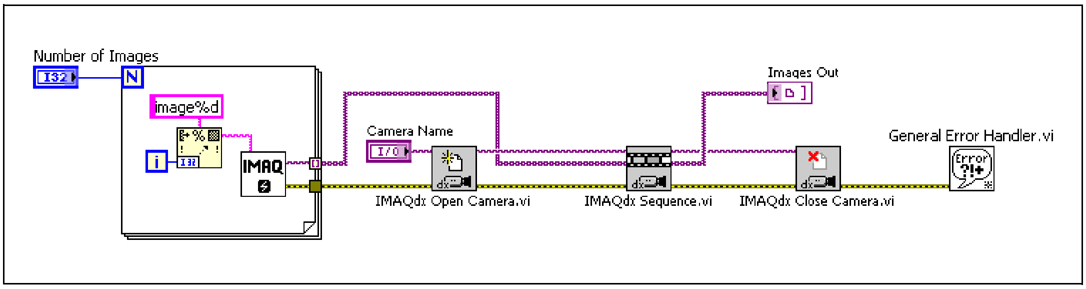

Figure 4-5. Acquiring Images Using Sequence

Image Display

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

You can display an image directly on the front panel using an Image Display control, which is available on the **Vision** Controls palette. To display an image on an Image Display control, place the control on the front panel of your VI. On the block diagram, wire **Image Out** from an acquisition VI to the Image Display control terminal.

Figure 4-6 illustrates using an image control to display an image using an Image Display control. For more information about Image Display controls, refer to the *NI Vision for LabVIEW VI Reference Help*.

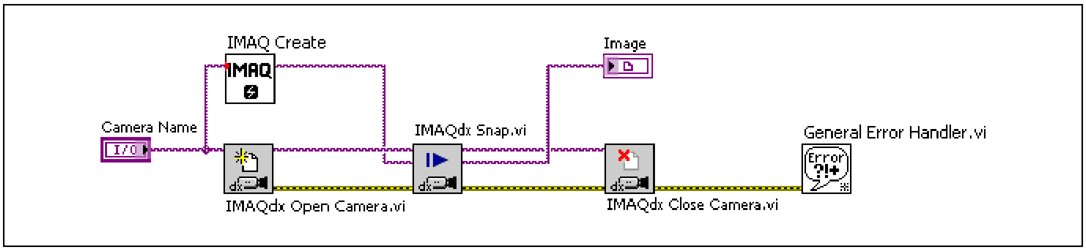

Figure 4-6. Displaying an Image Using an Image Control

If you have NI Vision for LabVIEW, you can display an image in an external window using IMAQ WindDraw, located at **Vision and Motion» Vision Utilities»External Display**. Use IMAQ WindDraw when you need more image size and location control.

Figure 4-7 illustrates using IMAQ WindDraw to display an image acquired using IMAQdx Snap. You can display images in the same way using any acquisition type. For more information about the display capabilities of NI Vision, refer to the *NI Vision for LabVIEW User Manual*.

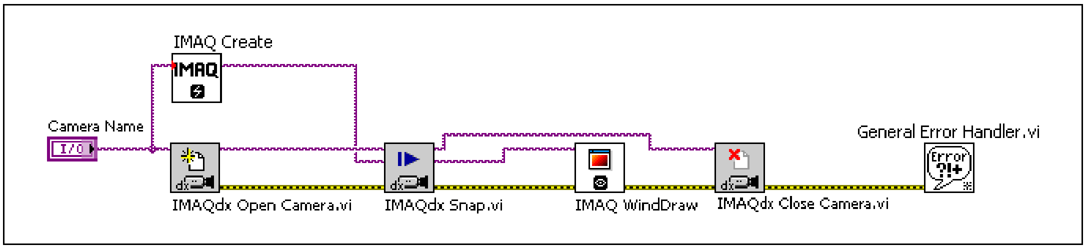

Figure 4-7. Displaying an Image Using IMAQ WindDraw

If you have LabVIEW RT, you can use IMAQ RT Video Out, located at **Vision and Motion»Vision Utilities»IMAQ RT**, to display an image on the monitor connected to your RT device. Use IMAQ Video Out Display Mode, located at **Vision and Motion»Vision Utilities»IMAQ RT**, to configure the monitor for display. Figure 4-8 illustrates configuring the monitor and displaying an image acquired with IMAQdx Snap.

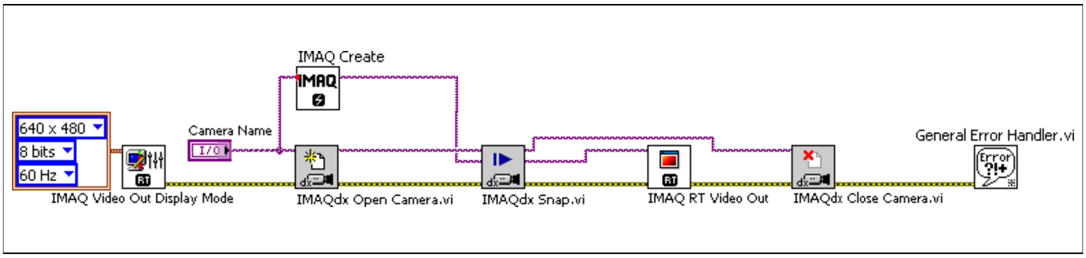

Figure 4-8. Displaying an Image Using RT Video Out

Note The IMAQ RT Video Out VI is available only on RT devices with Intel i815 or i845 video chipsets. These devices include NI CVS-1450 Series devices, PXI-817*x* controllers, and PXI-818*x* controllers.

Camera Attributes

To modify camera attributes in LabVIEW, use the IMAQdx Property Node. Every camera attribute has two parameters: Attribute Name and Attribute Value.

- **Attribute Name**—Specify the attribute name with the attribute property node. The attribute name is a string constant. The attribute name contains several keywords separated by a double colon namespace marker. The namespace marker separates different levels in the attribute tree as described by the driver and the camera. Use the fully qualified attribute name, for example AcquisitionAttributes::Timeout, or the short version of the attribute name, for example Timeout, when specifying the attribute name.
- **Attribute Value**—Enter a value type for the attribute. The value type must be compatible with the attribute type. Refer to Table 3-2 for a list of attribute value types. Use the **Range** property nodes to find the valid range for the current camera.

Figure 4-9 shows how to set camera attributes with the property nodes in NI-IMAQdx.

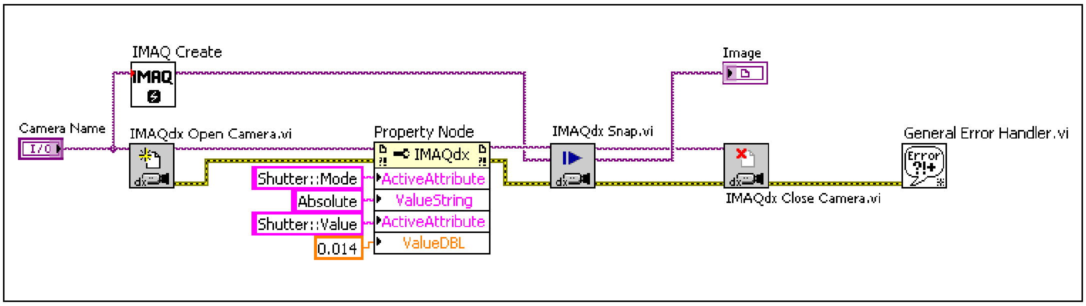

Figure 4-9. Setting Camera Attributes with Property Nodes

Error Handling

Every NI-IMAQdx VI contains an **error in** input cluster and an **error out** output cluster. The clusters, shown in Figure 4-10, contain a Boolean value that indicates whether 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-IMAQdx function.

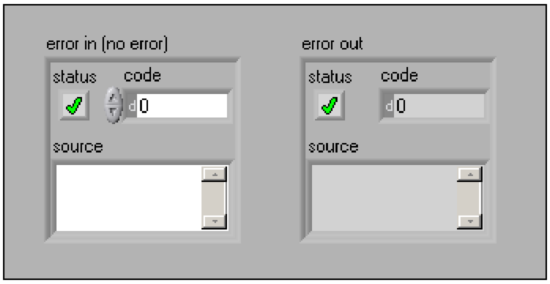

Figure 4-10. Error Clusters

You can use the Simple Error Handler VI, located on the **Functions» Dialog & User Interface** palette, to check for errors that occur while executing a VI. If you wire an error cluster to the Simple Error Handle 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 4-11 illustrates wiring an NI-IMAQdx VI to the Simple Error Handler VI.

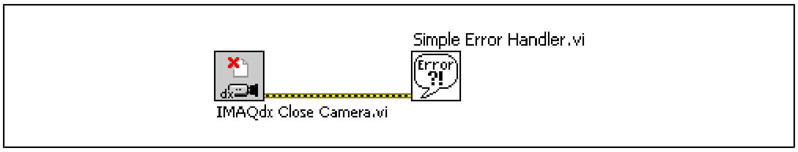

Figure 4-11. Error Checking Using the Simple Error Handler VI

5

Using NI-IMAQdx in C and .NET

This chapter briefly describes how to use NI-IMAQdx functions in Microsoft Visual C and Microsoft Visual Studio .NET.

Using NI-IMAQdx for C

This section outlines the process for developing NI-IMAQdx applications using 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 your particular development environment.

Note 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.

When programming, use the following guidelines:

- Include the `niimaqdx.h` header file in all C source files that use NI-IMAQdx functions. Add this file to the top of your source files.
- Add the `niimaqdx.lib` import library to your project. In some environments, you can add import libraries simply by inserting them into your list of project files. In other environments, you can 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-IMAQdx`. You can find the include files under the `include` subdirectory. The import libraries for Microsoft Visual C++ are located under the `lib\msvc` subdirectory.

You can use the additional Image functions installed with NI-IMAQdx. These functions use the NI Vision memory management feature, which automatically allocates the memory for your image. To use these Image functions, first create an image using `imaqCreate`, and then pass that image to an acquisition function.

Using NI-IMAQdx for Microsoft Visual Studio .NET

NI-IMAQdx installs the following assemblies that enable .NET languages to interact with the driver software:

- `NationalInstruments.CWNIIMAQDX.Interop.dll`
- `NationalInstruments.AxCWNIIMAQDXControlsLib.Interop.dll`

  Uses NI Vision to display images with the included Viewer control

The `CWIMAQdx` assembly is installed in the `<NI-IMAQdx>\dotNET\Assemblies\Current` directory. The `AxCWIMAQControlsLib` assembly is installed in the `<Vision>\dotNET\Assemblies\Current` directory. Refer to the *NI-IMAQdx Function Reference Help* for information about the properties, methods, and events available with these assemblies.

Creating a New .NET Application

You first must add a reference to the `NI-IMAQdx` assembly in your project when creating a new application. Complete the following steps to add a reference to the `NI-IMAQdx` assembly in Microsoft Visual Studio .NET 2003 or later:

1. Create a new application, or open an existing one.
2. Select **Project»Add Reference**.
3. Under the **.NET Framework Components** tab, select **NI-IMAQdx**.

If you need to display acquired images, you also must add an NI Vision Viewer control to your toolbox and to your form. Complete the following steps to add the NI Vision Viewer control to the Microsoft Visual Studio .NET toolbox.

1. With your project open, open a form in Design View.
2. Open the Toolbox (**View»Toolbox**).
3. Select the category in which you want the NI Vision controls to appear (General, Components, and so on).

4. Select **Tools»Add/Remove Toolbox Items**.
5. Under the **.NET Framework Components** tab, select the CWIMAQViewer control.

When the Viewer control is in the toolbox, you can add it to your forms by clicking on the tool and drawing an area on the form. References to the NI Vision Interop Assemblies are automatically added to your project.

A Register-Level Programming

This appendix explains how to access and program register locations using the NI-IMAQdx software, and discusses the caveats involved in programming registers.

Introduction

All cameras communicate to the host computer through register maps. The register map reflects the system memory located on the camera. The register map allows the host computer to read and write information with minimal overhead.

The host computer sends *asynchronous* messages over the host bus to the connected camera. When the data is written into memory on the camera, the camera processes the incoming request. If possible, the camera responds immediately. Otherwise, a pending transaction message is returned. When the pending request is completed, the camera returns the results of the request.

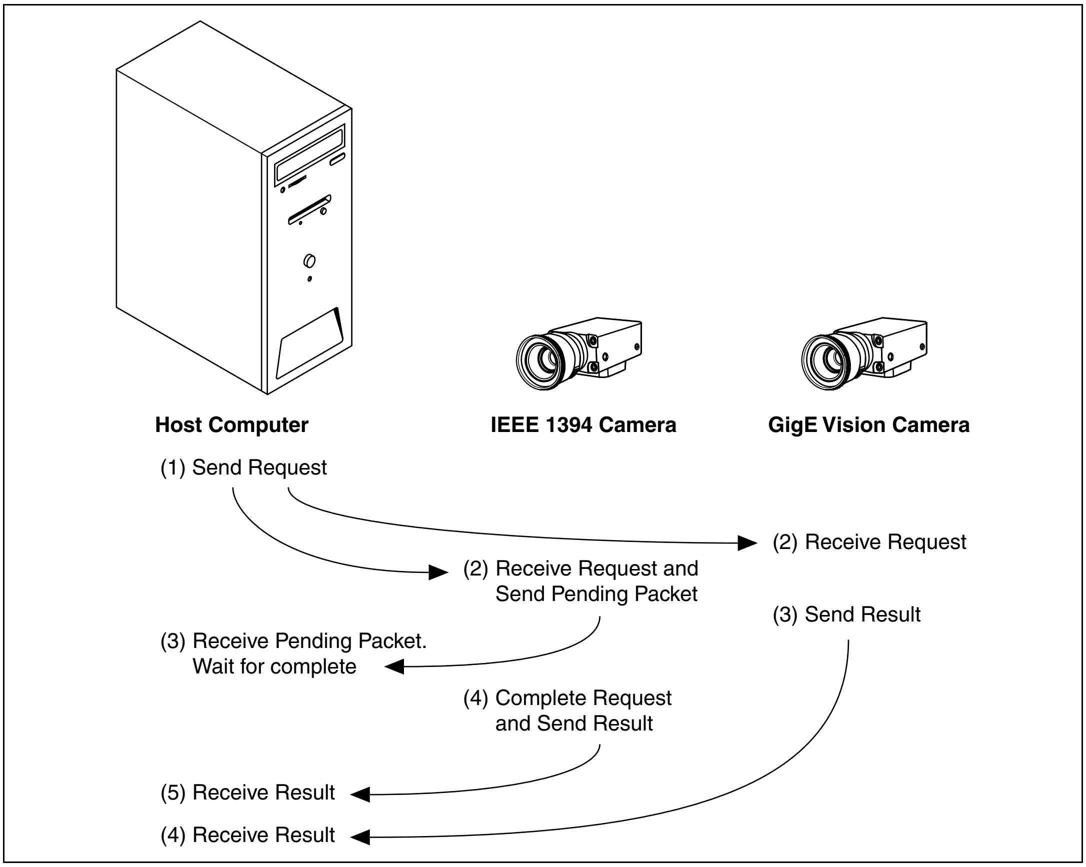

Figure A-1. Explanation of Split Transactions

NI-IMAQdx supports the 1394 Trade Association IIDC 1.31 register specification and the GigE Vision 1.0 specification for industrial cameras. Most of the intricacies of register-level programming are abstracted by the driver. The driver is responsible for manipulating camera features and activating/deactivating the video data stream.

Some cameras implement additional registers that are not contained in the IIDC 1.31 or GigE Vision 1.0 specifications. These advanced camera features are not natively supported by the camera driver. To use these advanced features, you must use the low-level, register-level access tools to communicate with the camera.

GigE Vision cameras have all features defined in an XML file which normally eliminates the need for direct register programming.

The NI-IMAQdx software provides the following register-level primitives:

- Read Register—Reads 32-bits of data from a specified memory location
- Write Register—Writes 32-bits of data to a specified memory location
- Read Memory—Reads an array of bytes from a specified memory location and range
- Write Memory—Writes an array of bytes to a specified memory location

Usage

To perform a register-level access, specify a memory location (or offset) and data storage. IEEE 1394 memory locations are specified as 48-bit values. The upper 20 bits are filled in by the driver. The low-level register primitives accept the lower 28-bit offset. The memory storage contains the result/desired data when transferring. GigE Vision memory locations are specified as 32-bit values.

Basic Example

The isonchronous enable register indicates active video transmission. To read the ISO_EN register (0x614), calculate the memory offset by adding the specified offset to the base register. The base register is 0xF0F00000 for most IEEE 1394 cameras.

0xF0F00000 + 0x614 = 0xF0F00614

The value is read, and the result is placed in the specified memory location.

read register (0xF0F00614) = <*iso_en*>

where <*iso_en*> = (0x80000000 or 0x00000000).

If bit 0 has a value of 0x80000000, the bit is on, and the camera is transmitting video data. If bit 0 has a value of 0x00000000, the camera is not currently transmitting data.

Advanced Example

The advanced feature described in this example is specific to Basler IEEE 1394 cameras. The advanced feature replaces the live video feed with a static test pattern.

According to the user documentation for the Basler A601f camera, the TEST_IMAGE register is located at advanced offset 0x0098. You can enable a static test pattern by setting bit 17 of the TEST_IMAGE register. To get the advanced base register, first read ADVANCED_FEATURE_INQ register (0x480). Add the specified offset to the base register—0xF0F00000 for most IEEE 1394 cameras.

0xF0F00000 + 0x480 = 0xF0F00480

Read the value into storage.

read register (0xF0F00480) = <*advanced_feature_inq*>

where <*advanced_feature_inq*> = 0x800000.

Now, calculate the offset to the advanced feature offset. You need to multiply the previous result by 4 to convert the quadlet offset value to byte offset.

(0xF0F00000 + (<*advanced feature offset*> × 4) + 0x98) = newly calculated offset

byte swap (1 << 17) = newly calculated register mask

write register (0xF2F00098, 0x00002000)

Now the camera is set to the test pattern.

Caveats

This section discusses caveats to consider when programming registers using the NI-IMAQdx software.

Endianness

Data that spans multiple bytes, such as a quadlet, may be written left-to-right or right-to-left. The method with which data is written is called *endianness*. Two types of endianness exist: *big endian* and *little endian*.

The ethernet and IEEE 1394 bus transports data using the big endian method. However, Windows and LabVIEW RT host machines accept little endian data. To correct for this discrepancy, NI-IMAQdx byte-swaps every quadlet that is read or written with low-level register primitives.

Byte Array

Many cameras allow register-level access to more than 32 bits of data per communication request. In most cases, you can safely write and read a large, contiguous block of data to and from the connected camera. Some cameras fail when trying to access large payloads. If the camera does not successfully transfer an array of bytes, attempt to transfer the smaller packets of data one at a time.

Timing

Many cameras are responsive to successive register accesses. In most cases, you can safely read and write registers as quickly as possible. Some cameras lock up under stressed conditions. The camera driver inserts an artificial delay between register accesses. You can change this artificial delay in the registry under the following registry key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\niimaqdxk\Parameters\AsyncTransferDelay`.

The key specifies the millisecond value to delay before each transaction. After changing the value, reboot the host computer to enable the changes.

Note Changing this delay affects the entire driver, not just register-level access.

Invalid Memory Location

The NI-IMAQdx software allows access to register locations that do not exist. If an error occurs while accessing the register, check the validity of the register location.

B

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 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 Discussion Forums at `ni.com/forums`. National Instruments Application Engineers make sure every question receives an answer.

    For information about other technical support options in your area, visit `ni.com/services` or contact your local office at `ni.com/contact`.
- **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, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit `ni.com/alliance`.

If you searched `ni.com` and could not find the answers you need, contact your local office or NI corporate headquarters. 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

A

acquisition window	The image size specific to a video standard or camera resolution.
address	Value that identifies a specific location (or series of locations) in memory.
API	Application programming interface.
area	A rectangular portion of an acquisition window or frame that is controlled and defined by software.
array	Ordered, indexed set of data elements of the same type.
aspect ratio	The ratio of a picture or image's width to its height.
asynchronous	(1) Independent in time from any other event. (2) Communication mechanism on the IEEE 1394 bus, which guarantees delivery of the message but does not guarantee timing.

B

big endian	Describes computers that store bytes of memory by placing the most significant byte at the memory location with the lowest address, the next significant byte at the next memory location, and so on.
buffer	Temporary storage for acquired data.

C

camera session	A process-safe handle to a camera.
chroma	The color information in a video signal.

D

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).
DLL	Dynamic Link Library—A software module in Microsoft Windows containing executable code and data that can be called or used by Windows applications or other DLLs; functions and data in a DLL are loaded and linked at run time when they are referenced by a Windows application or other DLLs.
driver	Software that controls a specific hardware device, such as a camera.

E

endianness	The convention describing the ordering of bytes in memory or the sequence in which bytes are transmitted.
external trigger	A voltage pulse from an external source that triggers an event such as A/D conversion.

F

FIFO	First-In First-Out—The first data stored in the memory buffer is the first data sent to the acceptor. FIFOs are used on image acquisition devices to temporarily store incoming data until that data can be retrieved.
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's brightness level and the voltage level needed to produce that brightness.
Gigabit Ethernet	Describes technologies which transmit Ethernet packets at a rate of a gigabit per second.

GigE Vision	A camera interface standard developed using the Gigabit Ethernet communication protocol.
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 abstraction layer	Separates software API capabilities, such as general acquisition and control functions, from hardware-specific information.
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 color spectrum. *See also* RGB.

I

I/O	Input/Output—The transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.
IEEE	Institute of Electrical and Electronics Engineers.
internal buffer	A page-locked buffer. *See also* page-locked buffer.

L

library	A file containing compiled object modules, each comprised of one of more functions, that can be linked to other object modules that make use of these functions.
little endian	Describes computers that store bytes of memory by placing the least significant byte at the memory location with the lowest address, the second least significant byte at the next memory location, and so on.
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.

M

MAX	Measurement & Automation Explorer—A controlled, centralized configuration environment that allows you to configure all of your NI devices.

N

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

P

page-locked buffer	Memory page that is marked as non-pagable by the virtual file system. Page-locked buffers remain in physical memory and do not cause page faults
pixel	Picture element. The smallest division that makes up the video scan line. For display on a computer monitor, a pixel's optimum dimension is square (aspect ratio of 1:1, or the width equal to the height).
process-safe handle	A handle that allows only one process to access a camera at any given time.
protocol	The exact sequence of bits, characters, and control codes used to transfer data between computers and peripherals through a communications channel.

Q

quadlet	A 32-bit (four-byte) word.

R

real time	A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.
resolution	(1) The number of rows and columns of pixels. An image composed of m rows and n columns has a resolution of $n \times m$. 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: 8 bits for red, 8 bits for green, 8 bits for blue, and 8 bits for the alpha value (unused).
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.

S

sequence	Performs an acquisition that acquires a specified number of buffers, then stops.
snap	Acquires a single image to a buffer.
syntax	Set of rules to which statements must conform in a particular programming language.

T

timeout	Length of time, in milliseconds, that the driver waits for an image from the camera before returning an error
transfer rate	The rate, measured in bytes/s, at which data is moved from source to destination after software initialization and set up operations. The maximum rate at which the hardware can operate.
trigger	Any event that causes or starts some form of data capture.

U

user buffer	A memory buffer created by the user as a destination for the image. In LabVIEW, this is created with the IMAQ Create VI.
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.

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