IMAQ™

NI-IMAQ™ for IEEE 1394 Cameras User Manual

Image Acquisition Software

NATIONAL INSTRUMENTS™

June 2003 Edition
Part Number 370362B-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 (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949, Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838, Czech Republic 420 2 2423 5774, 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 1800 300 800, Norway 47 0 66 90 76 60, Poland 48 0 22 3390 150, Portugal 351 210 311 210, Russia 7 095 238 7139, 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

CVI™, IMAQ™, LabVIEW™, National Instruments™, NI™, ni.com™, and NI-IMAQ™ are trademarks of National Instruments Corporation.

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

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.

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 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, and code excerpts.

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

Contents

Chapter 1
Introduction to NI-IMAQ for IEEE 1394 Cameras

Chapter 2
Programming with NI-IMAQ for IEEE 1394 Cameras

Chapter 3
Programming with NI-IMAQ for IEEE 1394 Cameras VIs

Chapter 4
Software Overview for C

Appendix A
Technical Support and Professional Services

Glossary

Index

1

Introduction to NI-IMAQ for IEEE 1394 Cameras

This chapter describes the NI-IMAQ for IEEE 1394 Cameras software, lists the compatible application development environments (ADEs), describes the fundamentals of creating NI-IMAQ applications for Windows 2000/XP/Me/98, describes the files used to build these applications, and explains where to find sample programs.

About the NI-IMAQ Software

NI-IMAQ for IEEE 1394 Cameras gives you the ability to use industrial digital video cameras with the NI-IMAQ driver software and IMAQ Vision. You can use cameras with the following output formats:

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

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

NI-IMAQ for IEEE 1394 Cameras complies with the Industrial Infrastructure Development Centre (IIDC) specification and controls all available modes of the digital camera.

Application Development Environments

This release of NI-IMAQ for IEEE 1394 Cameras supports the following ADEs for Windows 2000/XP/Me/98:

- LabVIEW version 6.0 and later
- LabVIEW Real-Time 7.0 and later

- LabWindows™/CVI™ version 5.0 and later
- Microsoft Visual C/C++ version 6.0 and later
- Microsoft Visual Basic 6.0 and later

Note Although the NI-IMAQ for IEEE 1394 Cameras software has been tested and found to work with these ADEs, other ADEs may also work.

Configuring Your IEEE 1394 Camera

You can use National Instruments Measurement & Automation Explorer (MAX) to configure your IEEE-1394 camera. Refer to the *Measurement & Automation Explorer Help for NI-IMAQ for IEEE 1394 Cameras* for information about configuring your IEEE 1394 camera.

The camera information is saved in a camera file, which the NI-IMAQ for IEEE 1394 Cameras VIs and functions use to select a camera and supported attributes.

Fundamentals of Building Applications with NI-IMAQ for IEEE 1394 Cameras

Architecture

A block diagram of the NI-IMAQ for IEEE 1394 Cameras architecture shown in Figure 1-1 illustrates the low-level and mid-level architecture for IMAQ devices.

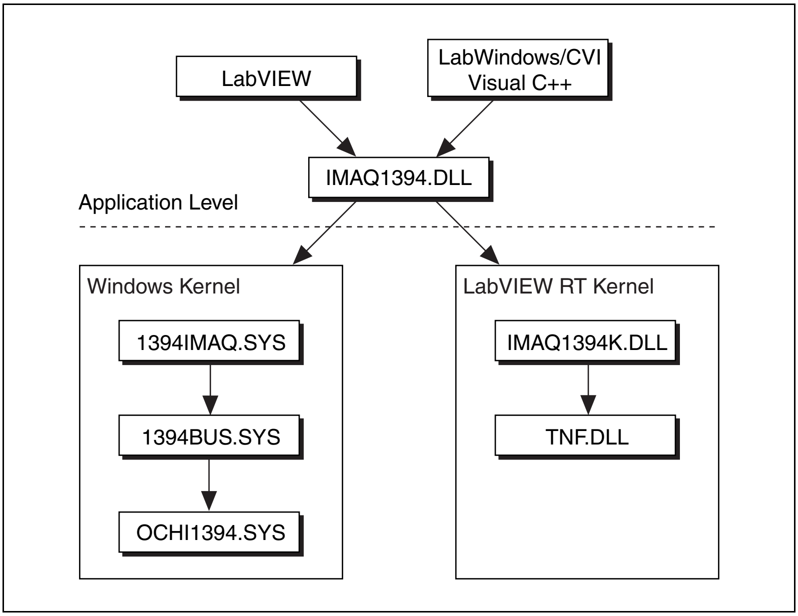

Figure 1-1. NI-IMAQ for IEEE 1394 Cameras Architecture

The NI-IMAQ Libraries

The NI-IMAQ for IEEE 1394 Cameras for Windows 2000/XP/Me/98 function libraries are dynamic link libraries (DLLs), which means that NI-IMAQ for IEEE 1394 Cameras routines are not linked into the executable files of applications. Only the information about the NI-IMAQ routines in the NI-IMAQ import libraries is stored in the executable files.

Import libraries contain information about their DLL-exported functions. They indicate the presence and location of the DLL routines. Depending on the development tools you use, you can give the DLL routines information through import libraries or through function declarations. Your NI-IMAQ for IEEE 1394 Cameras software kit contains function prototypes for all routines.

Creating an Application Using LabVIEW or LabVIEW Real-Time

When you install NI-IMAQ for IEEE 1394 Cameras, a subpalette of VIs appears on the NI-IMAQ functions palette. If you are using LabVIEW 7.0 or later, NI-IMAQ for IEEE 1394 Cameras also installs an image display control on your controls palette.

Create NI-IMAQ for IEEE 1394 Cameras 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.

Creating an Application Using C

This section outlines the process for developing NI-IMAQ for IEEE 1394 Cameras applications using C for Windows 2000/XP/Me/98. Detailed instructions on 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.

When programming, use the following guidelines:

- Include the `niimaq1394.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 `niimaq1394.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 for IEEE-1394`. 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 for IEEE 1394 Cameras sample programs. These programs are installed in the `sample` subdirectory under the target installation folder, if you elected to install the sample files.

2 Programming with NI-IMAQ for IEEE 1394 Cameras

This chapter contains an overview of the NI-IMAQ for IEEE 1394 Cameras library, a description of the programming flow of NI-IMAQ for IEEE 1394 Cameras, and programming examples. Flowcharts are included for the following operations: snap, grab, and sequence.

Introduction

The NI-IMAQ for IEEE 1394 Cameras application programming interface (API) is divided into two groups—high-level functions and low-level functions. With the high-level functions, you can write programs quickly without having to learn the details of the low-level API and driver. The low-level functions give you finer granularity and control over your image acquisition process, but you must understand the API and driver in greater detail.

Note The high-level functions call low-level functions and use certain attributes that are listed in the high-level function description of the *NI-IMAQ for IEEE 1394 Cameras Function Reference Help*. Changing the value of these attributes while using low-level functions will affect the operation of the high-level functions.

High-Level Functions

The high-level function set supports three basic types of image acquisition.

- *Snap* acquires a single frame to a buffer.
- *Grab* performs an acquisition that loops continually on a buffer. Obtain a copy of the latest 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.

The high-level function set also allows triggered acquisitions.

Note For LabVIEW RT systems, all high-level and low-level acquisitions are asynchronous. For Windows systems, all high-level acquisitions are synchronous.

Low-Level Functions

The low-level function set supports all types of acquisition. You can use low-level functions to start a synchronous or asynchronous acquisition.

Establishing Interface Connections

To acquire images using the high-level or low-level functions, you must first learn how to establish a connection to a camera. Refer to the *Camera Functions* and *Camera Attributes* sections for information about how to manage cameras, then refer to the high-level or low-level samples for information about acquiring images.

Camera Functions

Use camera functions to query the number of available cameras, establish a connection to, control access to, and initialize cameras. All parameters configured in MAX for an IMAQ 1394 interface are associated with a camera name. You can have one interface associated with more than one camera name, which allows you to have several different configurations for one device. Use the camera name to refer to the IMAQ 1394 device in your programming environment. Camera name information is stored in an interface (`.icd`) file.

NI-IMAQ for IEEE 1394 Cameras specifies all interfaces by a name. By default, the system creates default names for the number of cameras in your system. These names observe the convention shown in Table 2-1.

Table 2-1. Camera Naming Convention

Camera Name	IMAQ 1394 Device Installed
`cam0`	Device 0
`cam1`	Device 1
...	...
`cam`*n*	Device *n*

You can edit existing cameras or create new cameras in MAX. You also can use MAX to configure the default state of a particular camera.

Before you can acquire image data successfully, you must open a camera with IMAQ 1394 Initialize. IMAQ 1394 Initialize requires a camera name and outputs a session for this interface. NI-IMAQ for IEEE 1394 Cameras then uses this session to reference this camera when using other NI-IMAQ functions.

For example, use IMAQ 1394 Initialize to open a camera named `cam0`. When the program is finished with the camera, close the camera using IMAQ 1394 Close.

For a complete list of the available camera functions, refer to the *NI-IMAQ for IEEE 1394 Cameras Help*.

Camera Functions for LabVIEW Real-Time

Every camera has a `.iid` interface file and a `.icd` camera file. The interface files store information about which physical camera is connected to an interface name. The camera files store all the settable attributes. Each interface file can only be used by a single camera, whereas camera files can be shared between identical cameras.

After calling IMAQ 1394 Initialize, NI-IMAQ for IEEE 1394 Cameras assigns a default interface name when a new camera is detected. The interface names are assigned in the order the cameras are connected. Default interface and camera files are created during initialization. You can edit existing camera files or create new camera files in MAX.

When you open an interface name, the camera with the unique serial number described by the interface file `cam`*`n`*`.iid` opens where *`n`* is the reference to the camera. An error messages returns if the camera is not present. If the camera is not present and a camera of the same make and model is present, as described in the interface file, the new camera opens. The interface file updates to use the new camera. The camera file described by interface file opens and all the user attributes are set in the driver.

Camera Attributes

Use camera attributes to control camera-specific features such as brightness and shutter speed directly from NI-IMAQ for IEEE 1394 Cameras. You also can set camera attributes through the feature tab in MAX. All of the configured parameters for a camera are stored in a camera (`.icd`) file. This file is linked to a specific camera.

The following attributes are defined in the *1394 Based Digital Camera Specification*—Brightness, Auto_Exposure, Sharpness, White_Balance, Hue, Saturation, Gamma, Shutter, Gain, Iris, Focus, Temperature, Zoom, Pan, Tilt, and Optical Filter.

The LabVIEW, C, and Visual Basic APIs all provide get and set attribute functions to modify camera attributes. If your camera does not implement every attribute specified, the functions return an error.

Scalable Image Size

IEEE 1394 digital cameras support a predefined set of image sizes which you can select through the **Video** attributes in MAX. Refer to your camera documentation for a list of supported formats.

If you are using LabVIEW, the NI-IMAQ for IEEE 1394 Cameras software recognizes the predefined formats and automatically allocates enough memory to accommodate the image. When you use C or C++ with NI-IMAQ for IEEE 1394 Cameras functions, you must know the size of the image for the selected format and mode to allocate enough memory to contain the image. You can obtain the size of the image using the **Image Width**, **Image Height**, and **BytesPerPixel** attributes.

Some IEEE 1394 cameras support the Scalable Image Format (format 7), which allows you to define the size of the acquired image. If you use this format, you must input the image size using the **Rectangle** parameter in C and C++, as shown in Figure 2-1. The size and position of the sub-image you are acquiring must be a multiple of the attributes **Unit Width** and **Unit Height**, or the driver acquires the smallest sub-image that contains the ROI you defined.

The **Unit Width** and **Unit Height** values are camera-specific. Refer to the camera documentation or query the **Unit Width** and **Unit Height** attributes to obtain the actual values.

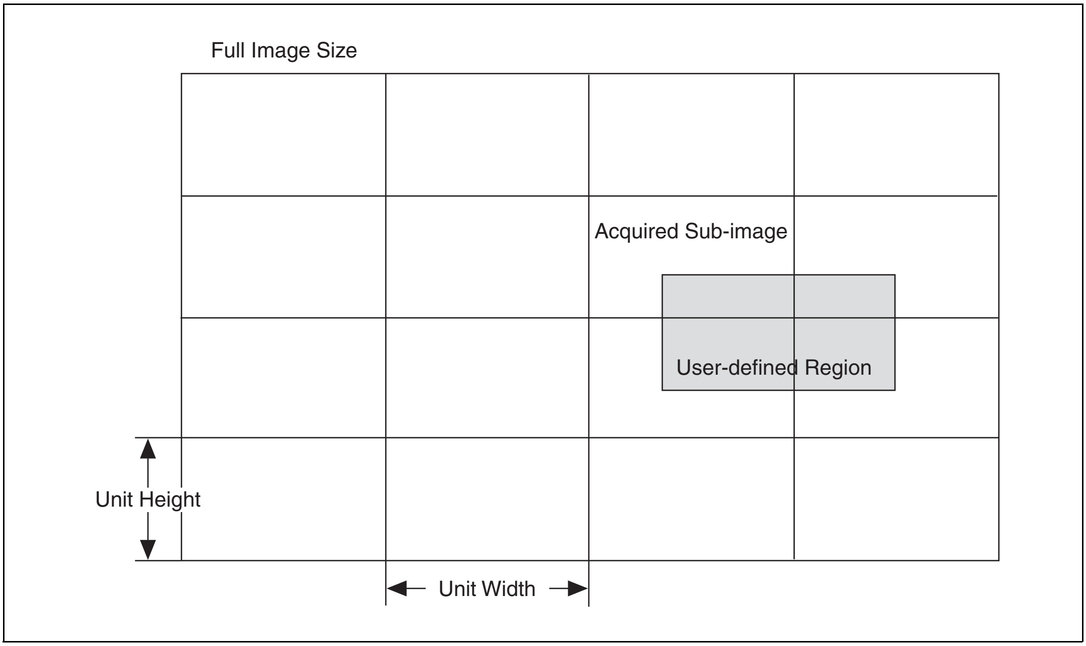

Figure 2-1. Scalable Image Format (format 7)

Introductory Programming Examples

This section introduces some examples for performing the different types of image acquisition.

Note The error codes returned by NI-IMAQ for IEEE 1394 Cameras are not included in the examples. Always check the return code for errors in your program.

If you have IMAQ Vision for CVI and C installed on your computer, you can take advantage of the additional Image functions installed with NI-IMAQ for IEEE 1394 Cameras. These functions use the IMAQ Vision memory management feature, in which you must first create an image using `imaqCreate` and then pass that image to an acquisition function.

If you are using CVI or C and do not have IMAQ Vision installed on your computer, you must manually allocate the memory for your image. Use the attributes **ImageWidth**, **ImageHeight**, and **BytesPerPixel** to determine how much memory you should allocate.

You can find code examples discussed in the section in the following directories:

- LabVIEW—`LabVIEW\examples\examples\imaq`
- CVI—`CVI\samples\imaq1394`
- C—`NI-IMAQ for IEEE 1394\examples\MSVC`
- Visual Basic—`NI-IMAQ for IEEE 1394\examples\VB`

High-Level Snap Functions

A *snap* acquires a single image into a memory buffer. Use this function to acquire a single frame to a buffer. If you do not have a valid session, a temporary session is created using `cam0`.

When you invoke a snap, it initializes the IMAQ 1394 device and acquires the next incoming video frame to a buffer. Use a snap for low-speed or single-capture applications where ease of programming is essential. Figure 2-2 illustrates a typical snap programming order.

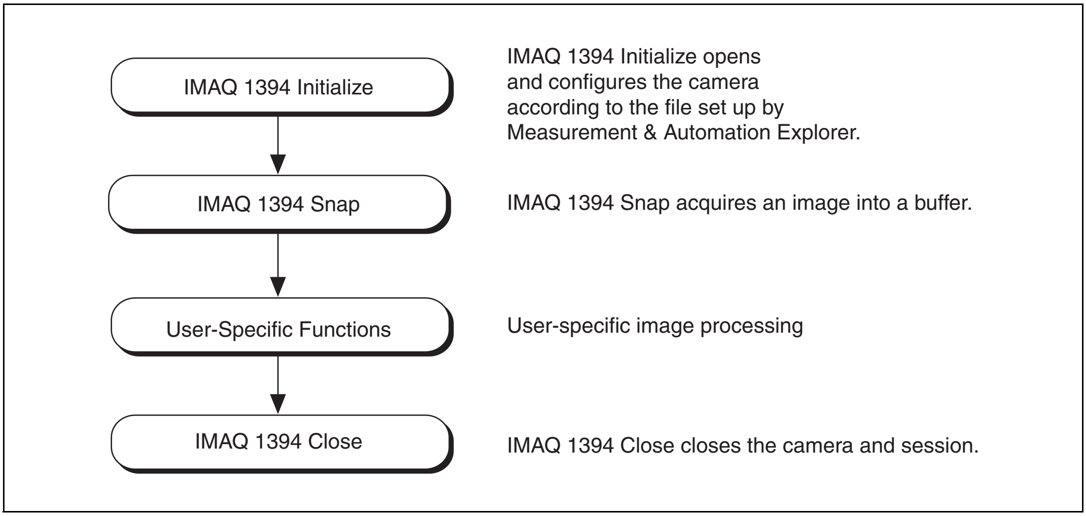

Figure 2-2. Snap Programming Flowchart

The snap examples demonstrate how to perform a single snap. The example opens a session on a camera and then performs a single snap. When you open a session, the ROI is set to the size of the video mode you selected in MAX.

The sample then calls a process function to analyze the image. When the program is finished, it calls IMAQ 1394 Close with the camera handle. This instructs NI-IMAQ to free all of the resources associated with this camera, which releases the session.

High-Level Grab Functions

A *grab* is a continuous high-speed acquisition of data to a single buffer in memory. You can 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. If you do not have a valid session, IMAQ 1394 Grab Setup creates a session using `cam0`.

Calling IMAQ 1394 Grab Setup initializes a session for a grab acquisition. After IMAQ 1394 Grab Setup, 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 2-3 illustrates a typical grab programming order.

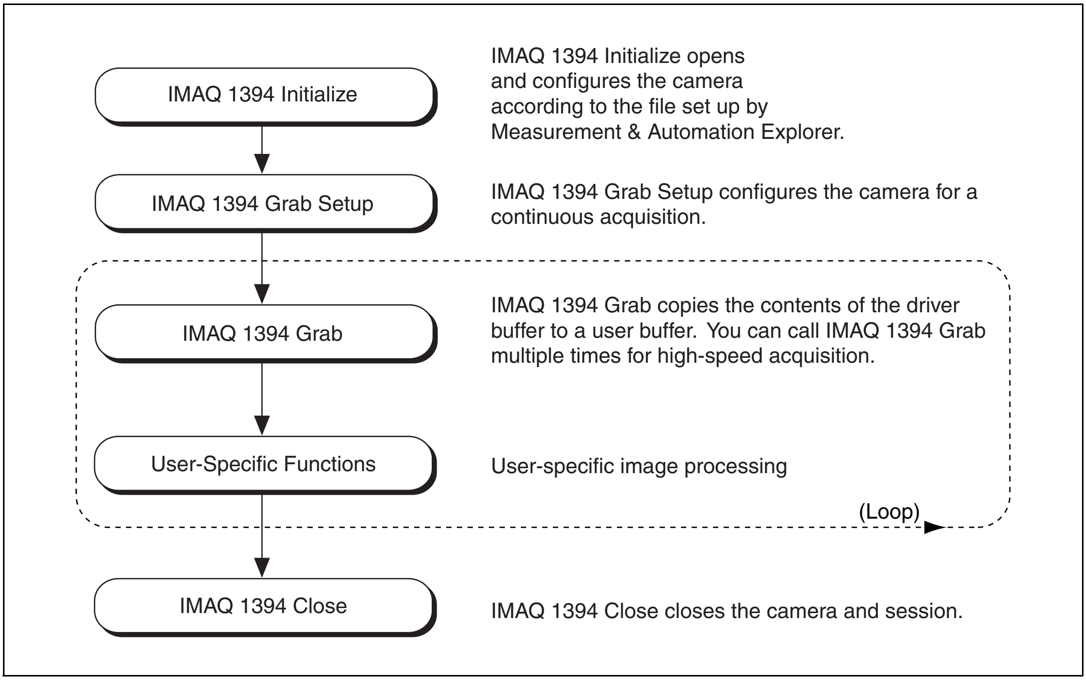

Figure 2-3. Grab Programming Flowchart

The grab examples demonstrate how to perform a grab. The example performs multiple grabs until the user stops the grab from the user interface.

High-Level Sequence Functions

A *sequence* initiates a variable-length transfer to multiple buffers. You can specify the number of buffers. IMAQ 1394 Sequence is synchronous and returns when all images have been acquired. If you do not specify a session, a temporary session is created using `cam0`.

Use a sequence in applications where you need to perform processing on multiple images. Figure 2-4 illustrates a typical sequence programming order.

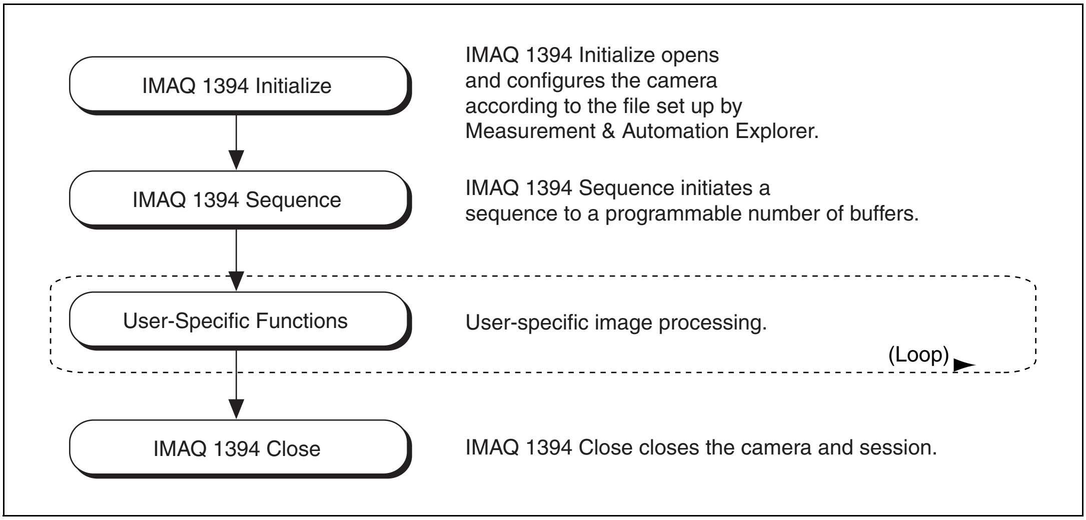

Figure 2-4. Sequence Programming Flowchart

The sequence example demonstrates how to perform a sequence acquisition using IMAQ 1394 Sequence. The example sets up a sequence that uses a user-defined number of buffers. The acquisition is started at setup time and the setup call is synchronous.

Advanced Programming Examples

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

Performing a Snap Using Low-Level Functions

The low-level snap examples demonstrate how to perform a snap acquisition using low-level calls. The examples set up a one-shot, single-frame acquisition. The program acquires the image, displays it, and closes the session.

Performing a Grab Using Low-Level Functions

The low-level grab examples demonstrate how to perform a grab acquisition using low-level calls. The examples set up a continuous acquisition to a single buffer. The main processing loop of the code shows how to copy the buffer to an analysis buffer.

Performing a Sequence Acquisition Using Low-Level Functions

The low-level sequence examples demonstrate how to perform a sequence acquisition using low-level calls. The examples set up a sequence acquisition to multiple buffers. The program starts the image acquisition asynchronously. The main processing loop of the code shows how to process each buffer acquired in sequential order.

3

Programming with NI-IMAQ for IEEE 1394 Cameras VIs

This chapter describes how to use the NI-IMAQ for IEEE 1394 Cameras VIs in LabVIEW.

Introduction

The NI-IMAQ for IEEE 1394 Cameras VI Library, a series of virtual instruments (VIs) for using LabVIEW with your IMAQ 1394 device, is included with your NI-IMAQ for IEEE 1394 Cameras software.

IMAQ Vision for LabVIEW is an image processing and analysis library that consists of more than 400 VIs. If you have not purchased the IMAQ Vision image processing and analysis libraries, you can use the four IMAQ Vision VIs included with your NI-IMAQ for IEEE 1394 Cameras software. If you use these basic functions, you can later upgrade your programs to use IMAQ Vision processing capabilities without any changes to your image acquisition VIs.

Before you start building your IMAQ application, you should know the following basic IMAQ for LabVIEW knowledge and concepts:

- Location of the NI-IMAQ for IEEE 1394 Cameras examples
- Location of the NI-IMAQ for IEEE 1394 Cameras VIs in LabVIEW
- Common NI-IMAQ for IEEE 1394 Cameras VI parameters
- Buffer management
- NI-IMAQ for IEEE 1394 Cameras acquisition types
- Acquisition VIs
- Triggering
- Image display

- Camera attributes
- Error handling
- Error code format

Location of NI-IMAQ for IEEE 1394 Cameras Examples

The NI-IMAQ VI for IEEE-1394 Cameras 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 select **Windows»Show VI Info** for a text description of the example.

Location of the NI-IMAQ for IEEE 1394 Cameras VIs

You can find the NI-IMAQ VIs in the **Functions** palette from your LabVIEW block diagram. Select **NI Measurements»Vision»IMAQ IEEE-1394**, as shown in Figure 3-1.

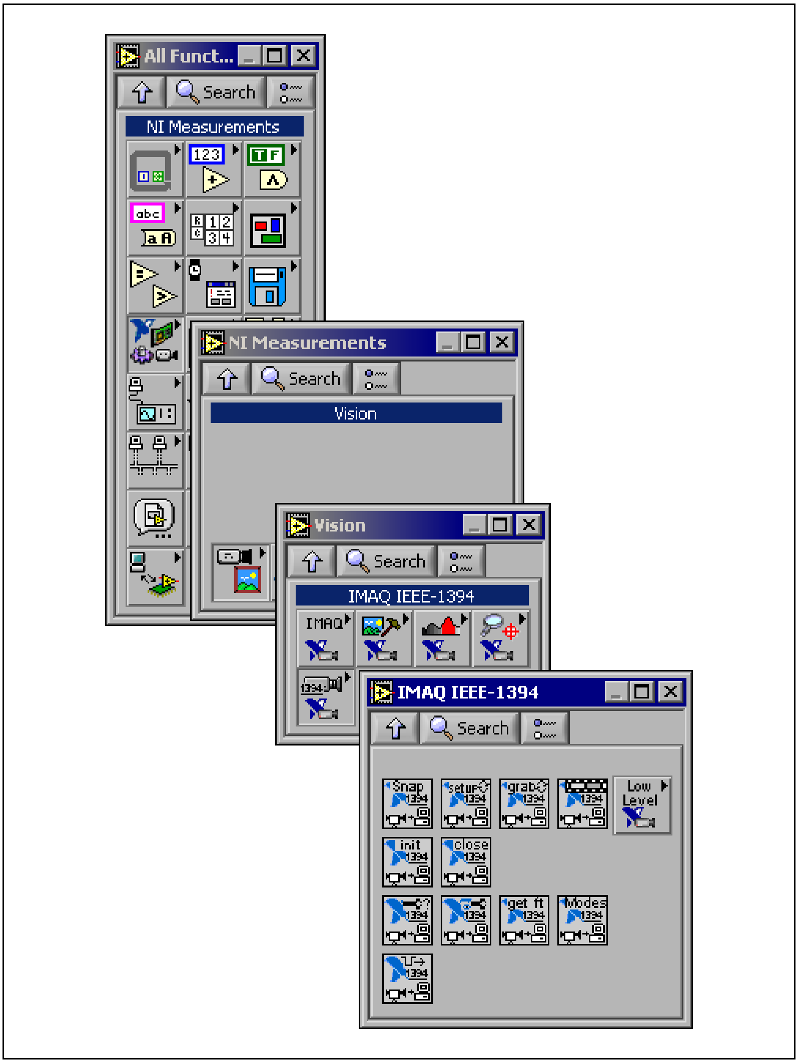

Figure 3-1. NI-IMAQ for IEEE 1394 Cameras Functions Palette

The most commonly used VIs are on the **IMAQ for IEEE-1394** palette. You can find VIs for basic acquisition and changing attributes. The **Vision»IMAQ for IEEE-1394»IMAQ Low Level** palette contains VIs for more advanced applications.

Refer to the *NI-IMAQ for IEEE 1394 VI Reference Help* for more information about using the VIs.

Common NI-IMAQ for IEEE 1394 Cameras VI Parameters

IMAQ1394 Session is a unique identifier that specifies the Interface file used for the acquisition. It is produced by the IMAQ1394 Init VI and used as an input to all other NI-IMAQ VIs. The NI-IMAQ for IEEE 1394 Cameras VIs use IMAQ1394 Session Out, which is identical to IMAQ1394 Session, to simplify dataflow programming. IMAQ1394 Session Out is similar to the duplicate file sessions provided by the file I/O VIs. The high-level acquisition VIs—IMAQ1394 Snap, IMAQ1394 Grab Setup, and IMAQ1394 Sequence—require you to wire IMAQ1394 Session In only if you are using an interface other than the default `cam0`, if you are using multiple cameras, or if you need to set IMAQ 1394 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 IMAQ Create. Refer to the *Buffer Management* section for more information about using buffers. **Image In** receives the image buffer. **Image Out** returns the captured image.

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, and Bottom. If **Region of Interest** is not wired, the entire image acquisition window is captured. You configure the default acquisition window using MAX.

Note For Windows, use **Region of Interest** input only when your camera is configured to use Scalable Image Format (format 7). For LabVIEW RT, you can use **Region of Interest** for other formats.

Buffer Management

IMAQ Create and IMAQ Dispose manage image buffers in LabVIEW. IMAQ Create, shown in Figure 3-2, 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 16-bit monochrome images, and **RGB** for RGB color images.

Note If **Image Type** is set to a value incompatible with the current video mode, it automatically changes to a compatible value.

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 wire to **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.

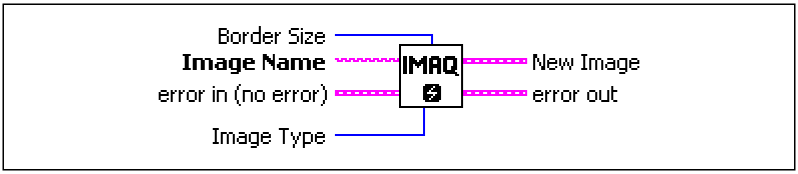

Figure 3-2. IMAQ Create

IMAQ Dispose, shown in Figure 3-3, frees the memory allocated for the image buffer. Call this VI only after the image is no longer required for processing.

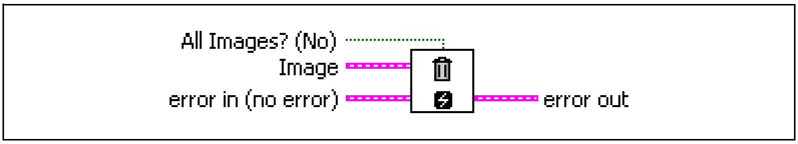

Figure 3-3. IMAQ Dispose

NI-IMAQ for IEEE 1394 Cameras Acquisition Types

Three NI-IMAQ image acquisition types are available in LabVIEW—snap, grab, and sequence. 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 to a buffer. Use a snap for low-speed or single-capture applications.

Use IMAQ1394 Snap for snap applications. Figure 3-4 shows a simplified block diagram for using IMAQ1394 Snap.

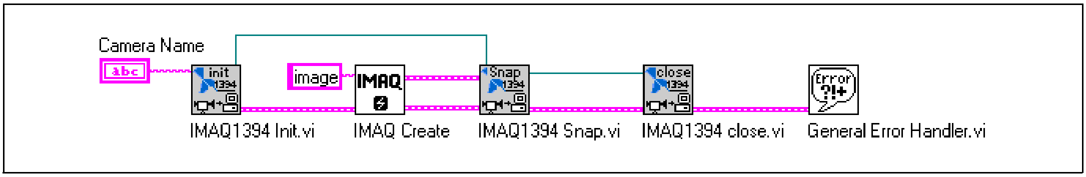

Figure 3-4. Acquiring an Image Using Snap

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—IMAQ1394 Grab Setup and IMAQ1394 Grab Acquire—for a grab acquisition in LabVIEW. Call IMAQ1394 Grab Setup once to initialize the acquisition and start capturing the image to an internal software buffer. You can call IMAQ1394 Grab Acquire multiple times to copy the image currently stored in the internal buffer to a LabVIEW image buffer. After the program finishes copying images, call IMAQ1394 Close once to shut down the acquisition.

Figure 3-5 shows a simplified block diagram for using IMAQ1394 Grab Setup and IMAQ1394 Grab Acquire.

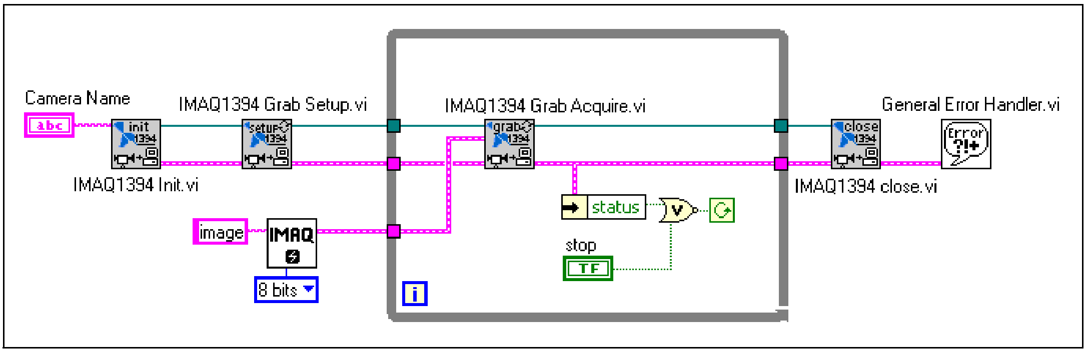

Figure 3-5. Acquiring Images Using Grab

Sequence

A sequence initiates a variable-length and variable-delay transfer to multiple buffers. Use a sequence for applications that process multiple images.

Use IMAQ1394 Sequence for sequence applications. IMAQ1394 Sequence starts, acquires, and releases a sequence acquisition. IMAQ1394 Sequence does not return until the entire sequence is acquired.

Figure 3-6 shows a simplified block diagram for using IMAQ1394 Sequence. Place IMAQ Create inside a For Loop to create an array of images for the Images in input to IMAQ1394 Sequence. To Decimal and Concatenate create a unique name for each image in the array.

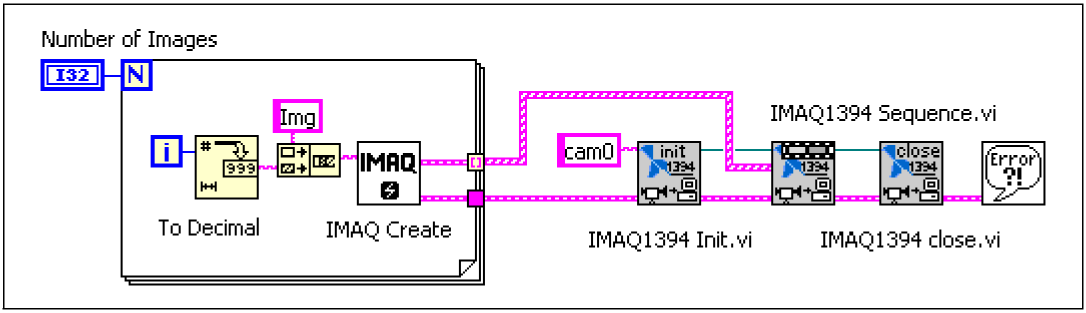

Figure 3-6. Acquiring Images Using Sequence

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. VIs are included for snap, grab, and sequence as described in the *NI-IMAQ for IEEE 1394 Cameras Acquisition Types* section. You can find examples of using the high-level acquisition VIs in the `examples\imaq\IMAQ1394examples.llb`.

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 in conjunction with the event VIs to construct advanced IMAQ applications.

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

1. Call IMAQ1394 Init to initialize the board and create an IMAQ1394 Session.
2. Call IMAQ1394 Configure Occurrence if you want to implement an asynchronous acquisition.
3. Call IMAQ1394 Start Acquisition.
4. Call IMAQ1394 Get Image.
5. After an acquisition, release the resources associated with the acquisition using IMAQ1394 Close. IMAQ1394 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 IMAQ1394 Stop Acquisition. Use IMAQ1394 Clear Acquisition to clear the acquisition and reconfigure.

Examples of the low-level acquisition VIs are included in the `examples\imaq\IMAQ1394examples.llb`.

Triggering

Often you may need 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. In these cases, use a triggered image acquisition.

Timeout specifies the amount of time, in milliseconds, to wait for the trigger. Figure 3-7 shows how to use IMAQ1394 Configure Trigger to perform a snap acquisition based on a trigger.

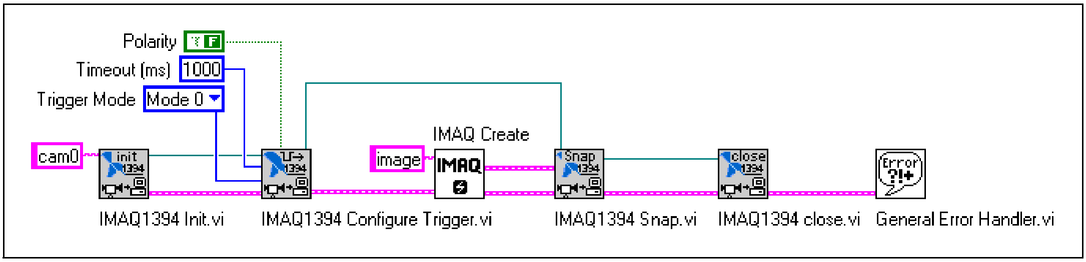

Figure 3-7. IMAQ Triggering

Image Display

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

If you have LabVIEW 7.0 or later, you 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 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 control terminal. Figure 3-8 illustrates using an image control to display an image using an image control. For more information about image controls, refer to the *IMAQ Vision VI Reference Help*.

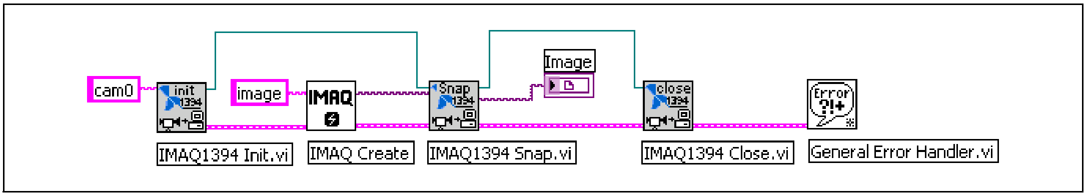

Figure 3-8. Displaying an Image Using an Image Control

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. You also can use IMAQ WindDraw when you need more image size and location control. IMAQ WindDraw, located at **Vision»Vision Utilities»External Display**, displays an image in a separate image window. Figure 3-9 illustrates using IMAQ WindDraw to display an image acquired using IMAQ1394 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 for LabVIEW User Manual*.

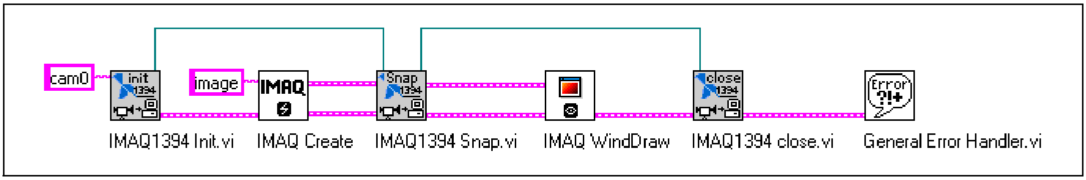

Figure 3-9. Displaying an Image Using IMAQ WindDraw

If you have LabVIEW RT, you can use IMAQ RT Video Out, located at **Vision»Vision Utilities»IMAQ RT**. IMAQ RT Video Out displays an image on the monitor connected to your RT device. Use IMAQ Video Out Display Mode, located at **Vision»Vision Utilities»IMAQ RT** to configure the monitor for display. Figure 3-10 illustrates configuring the monitor and displaying an image acquired with IMAQ1394 Snap.

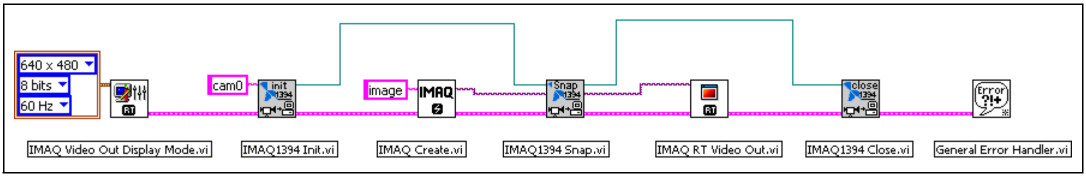

Figure 3-10. Displaying an Image Using RT Video Out

Note The Video Out VIs function only on RT devices with an Intel i815 video chipset. These devices include the NI-1450 series and PXI-817*x* controllers.

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 RGB image on a picture control, place the picture control on the front panel of your VI. Use IMAQ ColorImageToArray to copy an image from an image buffer into a LabVIEW array. Then you can wire this array to Draw True-Color Pixmap. 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 3-11 illustrates using a picture control to display an RGB image acquired with IMAQ1394 Snap.

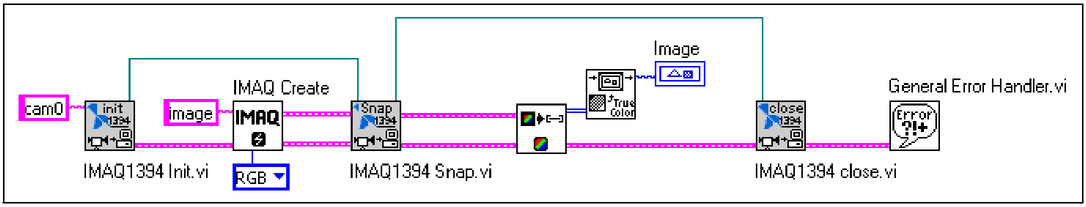

Figure 3-11. Displaying an RGB Image Using a Picture Control

Camera Attributes

Camera attributes allow you to control camera-specific features such as brightness and shutter speed directly from NI-IMAQ for IEEE 1394 Cameras. You can also set camera attributes through the feature tab in MAX. All of the configured parameters for a camera are stored in a camera (`.icd`) file. This file is linked to a specific camera.

The following attributes are defined in the *IEEE 1394 Based Digital Camera Specifications*—Brightness, Auto_Exposure, Sharpness, White_Balance, Hue, Saturation, Gamma, Shutter, Gain, Iris, Focus, Temperature, Zoom, Pan, Tilt, and Optical Filter. Attributes are camera-specific. Refer to your camera documentation for information about supported attributes

To modify these attributes in LabVIEW, use IMAQ1394 Attribute. Set the **Get/Set** parameter to FALSE to read the current value of the attribute, and TRUE to write the new value of the attribute.

Error Handling

Every NI-IMAQ for IEEE 1394 Cameras VI contains an **error in** input cluster and an **error out** output cluster, as shown in Figure 3-12. The clusters 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-IMAQ for IEEE 1394 Cameras function.

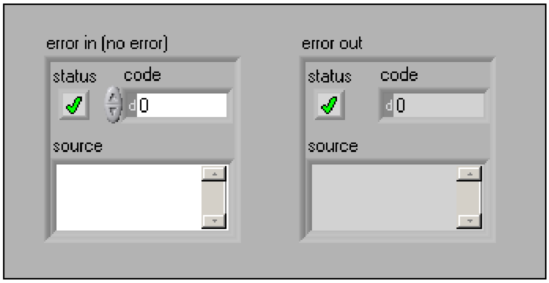

Figure 3-12. Error Clusters

You can use the Simple Error Handler VI, located on the **Functions» Time&Dialog** 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 3-13 shows how to wire an NI-IMAQ for 1394 Cameras VI to the Simple Error Handler VI.

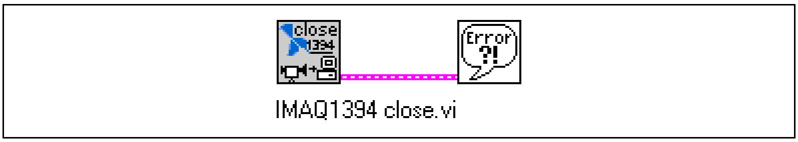

Figure 3-13. Error Checking Using the Simple Error Handler VI

Error Code Format

Error format for all NI-IMAQ for IEEE 1394 Cameras VIs is the same, 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.

4

Software Overview for C

This chapter describes the classes of NI-IMAQ for IEEE 1394 Cameras functions and briefly describes each function.

Introduction

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

- Generic functions
- High-level functions
    - Snap functions
    - Grab functions
    - Sequence functions
    - Trigger function
- Low-level functions
    - Acquisition functions
    - Attribute functions
    - Utility functions

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

Generic Functions

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

`imaq1394CameraOpen`	Opens a session on a camera by name.
`imaq1394Close`	Closes a session and unlocks and releases all buffers.

High-Level Functions

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

Snap Functions

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

`imaq1394SnapImage`	Performs a single frame acquisition in an image buffer, which is allocated using IMAQ Vision memory management.
`imaq1394Snap`	Performs a single frame acquisition in a memory buffer, which is allocated without using IMAQ Vision memory management.

Grab Functions

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

`imaq1394SetupGrab`	Configures and starts a continuous acquisition.
`imaq1394GrabImage`	Acquires the most current frame into the specified IMAQ Vision image buffer. Call this function only after calling `imaq1394SetupGrab`.
`imaq1394Grab`	Acquires the most current frame into a previously allocated buffer. Call this function only after calling `imaq1394SetupGrab`.

Sequence Functions

Sequence functions start and stop a continuous acquisition of multiple frames.

`imaq1394SetupSequenceImage`	Configures and starts a session for acquiring a full sequence into the list of buffers managed by IMAQ Vision.
`imaq1394SetupSequence`	Configures and starts a session for acquiring a full sequence in the buffer list.

Trigger Function

Trigger function controls the trigger mode of the IEEE 1394 camera.

`imaq1394TriggerConfigure`	Configures an acquisition to start based on an external trigger.

Low-Level Functions

Use low-level functions when you require more direct control of the acquisition of the images.

Acquisition Functions

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

`imaq1394ConfigureAcquisition`	Configures the acquisition session mode (continuous or one-shot).
`imaq1394StartAcquisition`	Starts acquisition synchronously or asynchronously.
`imaq1394StopAcquisition`	Stops an asynchronous acquisition or synchronous continuous acquisition immediately.
`imaq1394GetBuffer`	Copies a frame buffer to a user-specified buffer.

`imaq1394ClearAcquisition`	Clears a previously configured acquisition.
`imaq1394GetImage`	Copies a session's image data to an IMAQ Vision image.
`imaq1394InstallCallback`	Configures an asynchronous acquisition and installs a callback function that will be called when an image is acquired.

Attribute Functions

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

`imaq1394GetAttribute`	Returns an attribute for a session.
`imaq1394SetAttribute`	Sets an attribute for a session.
`imaq1394AttributeInquiry`	Queries the camera to check that it supports the specified attribute.
`imaq1394GetVideoModes`	Retrieves a list of video formats, modes, and frame rates supported by the camera.
`imaq1394GetFeatures`	Retrieves a list of features supported by the camera.

Utility Functions

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

`imaq1394Plot`	Plots a buffer to a window given a handle to a window.
`imaq1394PlotDC`	Plots a buffer to a window given a handle to a device context.
`imaq1394SaveBuffer`	Saves a buffer of a session to disk in BMP, TIFF, or PNG format.
`imaq1394ShowError`	Returns a null-terminated string describing the error code.

A

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 include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at `ni.com/support`. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting `ni.com/support`. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit `ni.com/training` for self-paced tutorials, videos, and interactive CDs. 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

A

acquisition window	The image size specific to a video standard or camera resolution.
active pixel region	The region of pixels actively being stored. 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.
API	Application programming interface.
AQ_DONE	Signals that the acquisition of a frame or field is completed.
AQ_IN_PROGRESS	Signals that the acquisition of video data is in progress.
area	A rectangular portion of an acquisition window or frame that is controlled and defined by software.
array	Ordered, indexed set of data elements of the same type.
aspect ratio	The ratio of a picture or image's width to its height.

B

b	Bit. One binary digit, either 0 or 1.
B	Byte. Eight related bits of data, an 8-bit binary number. Also denotes the amount of memory required to store 1 byte of data.
buffer	Temporary storage for acquired data.

C

cache	High-speed processor memory that buffers commonly used instructions or data to increase processing throughput.
chroma	The color information in a video signal.
chrominance	*See* chroma.

compiler	A software utility that converts a source program in a high-level programming language, such as Basic, C, or Pascal, into an object or compiled program in machine language. Compiled programs run 10 to 1,000 times faster than interpreted programs. *See also* interpreter.
conversion device	Device that transforms a signal from one form to another. For example, analog-to-digital converters (ADCs) for analog input and digital-to-analog converters (DACs) for analog output.
CPU	Central processing unit.

D

D/A	Digital-to-analog.
DAC	Digital-to-analog converter. An electronic device, often an integrated circuit, that converts a digital number into a corresponding analog voltage or current.
DAQ	Data acquisition. (1) Collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing; (2) Collecting and measuring the same kinds of electrical signals with A/D or DIO boards plugged into a computer, and possibly generating control signals with D/A and/or DIO boards in the same computer.
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.
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.
driver	Software that controls a specific hardware device, such as an IMAQ or DAQ device.

E

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.

frame — A complete image. In interlaced formats, a frame is composed of two fields.

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.

GUI — Graphical user interface. An intuitive, easy-to-use means of communicating information to and from a computer program by means of graphical screen displays. GUIs can resemble the front panels of instruments or other objects associated with a computer program.

H

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

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.

Hz — Hertz. Frequency in units of 1/second.

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.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that control specific plug-in computer boards. Instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW.
interlaced	A video frame composed of two interleaved fields. The number of lines in a field are half the number of lines in an interlaced frame.
interpreter	A software utility that executes source code from a high-level language such as Basic, C or Pascal, by reading one line at a time and executing the specified operation. *See also* compiler.

K

k	Kilo. The standard metric prefix for 1,000, or 10^3, used with units of measure such as volts, hertz, and meters.
K	Kilo. The prefix for 1,024, or 2^{10}, used with B in quantifying data or computer memory.
kbytes/s	A unit for data transfer that means 1,000 or 10^3 bytes/s.
Kword	1,024 words of memory.

L

library	A file containing compiled object modules, each comprised of one of more functions, that can be linked to other object modules that make use of these functions.
line count	The total number of horizontal lines in the picture.
LSB	Least significant bit.

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

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

N

NI-IMAQ	Driver software for National Instruments IMAQ hardware.
noninterlaced	A video frame where all the lines are scanned sequentially, instead of divided into two frames as in an interlaced video frame.

O

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

P

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.

picture aspect ratio — The ratio of the active pixel region to the active line region. For standard video signals like RS-170 or CCIR, the full-size picture aspect ratio normally is 4/3 (1.33).

pixel — Picture element. The smallest division that makes up the video scan line. For display on a computer monitor, a pixel's optimum dimension is square (aspect ratio of 1:1, or the width equal to the height).

pixel aspect ratio — The ratio between the physical horizontal size and the vertical size of the region covered by the pixel. An acquired pixel should optimally be square, thus the optimal value is 1.0, but typically it falls between 0.95 and 1.05, depending on camera quality.

protocol — The exact sequence of bits, characters, and control codes used to transfer data between computers and peripherals through a communications channel.

pts — Points.

R

RAM — Random-access memory.

real time — A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.

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.

ROM — Read-only memory.

S

s — Seconds.

syntax — Set of rules to which statements must conform in a particular programming language.

T

transfer rate — The rate, measured in bytes/s, at which data is moved from source to destination after software initialization and set up operations. The maximum rate at which the hardware can operate.

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

U

UV plane — *See* YUV.

V

V	Volts.
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

I

K

L

N

O

P

S