NATIONAL INSTRUMENTS™

NI-IMAQdx Function Reference Help

June 2008, 371968C-01

NI-IMAQdx driver software gives you the ability to acquire images from Gig E Vision IEEE 1394 industrial digital video cameras. This help file describes the functions included in the NI-IMAQdx driver software.

For more information about this help file, refer to the following topics:

To comment on National Instruments documentation, refer to the National Instruments Web site.

Related Documentation

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

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

- *Deployment Policy for NI-IMAQdx Note to Users*—Contains information about the deployment policy for NI-IMAQdx driver software.
- *Measurement & Automation Explorer Help for NI-IMAQdx*—Describes how to configure NI-IMAQdx driver software, NI image acquisition devices, and cameras using Measurement & Automation Explorer.
- *NI-IMAQdx Help*—Contains fundamental programming concepts for NI-IMAQdx driver software.
- *NI Vision Acquisition Software Release Notes*—Contains information about new functionality, minimum system requirements, and installation instructions for NI-IMAQdx driver software.

Activating Your Software

How do I activate my software?

Use the NI Activation Wizard to obtain an activation code for your software. You can launch the NI Activation Wizard two ways:

- Launch the product and choose to activate your software from the list of options presented.
- Launch NI License Manager by selecting **Start»All Programs»National Instruments»NI License Manager**. Click the **Activate** button in the toolbar.

Note You do not need to activate your software if it is managed by NI Volume License Manager as a part of a Volume License Agreement.

What is activation?

Activation is the process of obtaining an activation code to enable your software to run on your computer. An *activation code* is an alphanumeric string that verifies the software, version, and computer ID to enable features on your computer. Activation codes are unique and are valid on only one computer.

What is the NI Activation Wizard?

The NI Activation Wizard is a part of NI License Manager that steps you through the process of enabling software to run on your machine.

What information do I need to activate?

You need your product serial number, user name, and organization. The NI Activation Wizard determines the rest of the information. Certain activation methods may require additional information for delivery. This information is used only to activate your product. Complete disclosure of National Instruments licensing privacy policy is available at ni.com/activate/privacy. If you optionally choose to register your software, your information is protected under the National Instruments privacy policy, available at ni.com/privacy.

How do I find my product serial number?

You can find your serial number on the proof-of-ownership and registration card that you received with your product, as shown in the

following example.

What is a Computer ID?

The computer ID contains unique information about your computer. National Instruments requires this information to enable your software. You can find your computer ID through the NI Activation Wizard or by using NI License Manager, as follows:

1. Launch NI License Manager by selecting **Start»Programs»National Instruments»NI License Manager**.
2. Click the **Display Computer Information** button in the toolbar.

For more information about product activation and licensing refer to ni.com/activate.

Using Help

Conventions

This help file uses the following conventions:

< >	Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, DBIO<3..0>.
»	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, emphasis, or an introduction to a key concept.
green	Underlined text in this color denotes a link to a help topic, help file, or Web address.
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.

Navigating Help (Windows Only)

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

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

Searching Help (Windows Only)

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

Wildcards

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

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

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

Nested Expressions

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

Boolean Expressions

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

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

Search Options

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

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

Printing Help File Topics (Windows Only)

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

1. Right-click the book.
2. Select **Print** from the shortcut menu to display the **Print Topics** dialog box.
3. Select the **Print the selected heading and all subtopics** option.

   Note Select **Print the selected topic** if you want to print the single topic you have selected in the **Contents** tab.

4. Click the **OK** button.

Printing PDF Documents

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

LabWindows/CVI Function Tree

The following table shows the LabWindows/CVI function panel that corresponds to each NI-IMAQdx function.

Class/Panel Name	Function Name	Descriptio
High-Level Acquisition		
Snap	IMAQdxSnap	Configures, starts, a and unconfigures a acquisition.
Configure Grab	IMAQdxConfigureGrab	Configures and star acquisition. A grab p an acquisition that l continually on a ring buffers.
Grab	IMAQdxGrab	Acquires the most c frame into **image**.
Sequence	IMAQdxSequence	Configures, starts, a stops, and unconfig sequence acquisitic this function to capt multiple images.
Low-Level Session		
Reset Ethernet Camera Address	IMAQdxResetEthernetCameraAddress	Use this function to Ethernet cameras c network with a loca This function will be and will return wher is complete or after specified timeout.
Discover Ethernet Cameras	IMAQdxDiscoverEthernetCameras	Initiates a round of camera discovery. U function to find Ethe

		cameras on the net a remote subnet.
Enumerate Cameras	IMAQdxEnumerateCameras	Returns a list of all on the host comput
Reset Camera	IMAQdxResetCamera	Performs a manual a camera. Stops an acquisitions.
Open Camera	IMAQdxOpenCamera	Opens a camera, q camera for its capa loads a camera cor file, and creates a u reference to the car
Close Camera	IMAQdxCloseCamera	Stops an acquisitio progress, releases associated with an acquisition, and clo specified Camera S
Low-Level Acquisition		
Configure Acquisition	IMAQdxConfigureAcquisition	Configures a low-le acquisition previous opened with IMAQdxOpenCame
Start Acquisition	IMAQdxStartAcquisition	Starts an acquisitio previously configure IMAQdxConfigureA
Get Image	IMAQdxGetImage	Acquires the specifi into **image**. Call this only after calling IMAQdxConfigureA
Get Image Data	IMAQdxGetImageData	Copies the raw data specified frame into Call this function or calling IMAQdxConfigureA

Stop Acquisition	IMAQdxStopAcquisition	Stops an acquisitio previously started w IMAQdxStartAcquis
Unconfigure Acquisition	IMAQdxUnconfigureAcquisition	Unconfigures an ac previously configure IMAQdxConfigureA
Low-Level Attribute		
Enumerate Video Modes	IMAQdxEnumerateVideoModes	Returns a list of vid supported by the ca
Enumerate Attributes	IMAQdxEnumerateAttributes2	Gets the attributes by the camera.
Get Attribute	IMAQdxGetAttribute	Gets the current va camera attribute.
Set Attribute	IMAQdxSetAttribute	Sets the value for a attribute.
Get Attribute Minimum	IMAQdxGetAttributeMinimum	Gets the minimum camera attribute.
Get Attribute Maximum	IMAQdxGetAttributeMaximum	Gets the maximum camera attribute.
Get Attribute Increment	IMAQdxGetAttributeIncrement	Gets the increment camera attribute.
Get Attribute Type	IMAQdxGetAttributeType	Gets the attribute ty camera.
Is Attribute Readable	IMAQdxIsAttributeReadable	Gets the read perm for a camera attribu
Is Attribute Writable	IMAQdxIsAttributeWritable	Gets the write perm for a camera attribu
Enumerate	IMAQdxEnumerateAttributeValues	Gets the values sup

Attribute Values		the camera attribute
Get Attribute Tooltip	IMAQdxGetAttributeTooltip	Gets the tooltip for t camera attribute.
Get Attribute Units	IMAQdxGetAttributeUnits	Gets the attribute u camera.
Get Attribute Visibility	IMAQdxGetAttributeVisibility	Gets the visibility fo camera attribute.
Get Attribute Description	IMAQdxGetAttributeDescription	Gets the descriptior camera attribute.
Get Attribute Display Name	IMAQdxGetAttributeDisplayName	Gets the display na camera attribute.
Write Attributes	IMAQdxWriteAttributes	Saves a configurati a camera.
Read Attributes	IMAQdxReadAttributes	Loads a configurati a camera.
Low-Level Event		
Register Frame Done Event	IMAQdxRegisterFrameDoneEvent	Configures the NI-II driver to execute a function when a frai event occurs.
Register Plug and Play Event	IMAQdxRegisterPnpEvent	Configures the NI-II driver to execute a function when a plu play event occurs.
Low-Level Register		
Write	IMAQdxWriteRegister	Accesses registers

Register		camera and writes a value to the camera byte-swapped for b alignment before tra
Read Register	IMAQdxReadRegister	Accesses registers camera and reads a value from the cam is byte-swapped for endian alignment at transfer.
Write Memory	IMAQdxWriteMemory	Accesses registers camera and writes a the camera.
Read Memory	IMAQdxReadMemory	Accesses registers camera and reads a from the camera.
Low-Level Utility		
Get Error String	IMAQdxGetErrorString	Returns a string des the error code.

IMAQdxCloseCamera

Format

rval = IMAQdxCloseCamera (IMAQdxSession id);

Purpose

Stops an acquisition in progress, releases resources associated with an acquisition, and closes the specified session.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxConfigureAcquisition

Format

rval = IMAQdxConfigureAcquisition (IMAQdxSession id, unsigned int continuous, unsigned int bufferCount)

Purpose

Configures a low-level acquisition previously opened with IMAQdxOpenCamera. Specify the acquisition type using the **continuous** and **bufferCount** parameters.

Snap	Continuous = 0	Buffer Count = 1
Sequence	Continuous = 0	Buffer Count > 1
Grab	Continuous = 1	Buffer Count ³ 1

Use IMAQdxUnconfigureAcquisition to unconfigure the acquisition.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
continuous	unsigned int	Specifies whether the acquisition is continuous or one-shot.
bufferCount	unsigned int	For a one-shot acquisition, this parameter specifies the number of images to acquire. For a continuous acquisition, this parameter specifies the number of buffers the driver uses internally.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxConfigureGrab

Format

rval = IMAQdxConfigureGrab (IMAQdxSession id);

Purpose

Configures and starts an acquisition. A grab performs an acquisition that loops continually on a ring of buffers. Use a grab for high-speed image acquisition. Use IMAQdxGrab to copy an image out of the buffer. If you call this function before calling IMAQdxOpenCamera, IMAQdxConfigureGrab uses cam0 by default. Use IMAQdxUnconfigureAcquisition to unconfigure the acquisition.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxDiscoverEthernetCameras

Format

rval = IMAQdxDiscoverEthernetCameras (const char *address, unsigned int timeout);

Purpose

Detects Ethernet cameras on a network. Use this function to detect Ethernet cameras on a network with a remote subnet. During discovery, this function is blocked and returns after the specified timeout. The address specifies the destination address for the discovery command. The default address is 255.255.255.255. Call this function before calling IMAQdxEnumerateCameras or IMAQdxOpenCamera.

Parameters

Parameter	Type	Description
address	const char *	Address specifies the destination address for the discovery command. The default address is 255.255.255.255.
timeout	unsigned int	Timeout specifies the time, in milliseconds, allowed for the Ethernet camera discovery to complete. The default timeout is 1000 ms.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxEnumerateAttributes2

Format

rval = IMAQdxEnumerateAttributes2 (IMAQdxSession id, IMAQdxAttributeInformation attributeInformationArray[], unsigned int *count, const char *root, IMAQdxAttributeVisibility visibility)

Purpose

Gets the attributes supported by the camera. If you do not know in advance the number of features, complete the following steps:

1. Call this function with the **attributeInformationArray** parameter set to NULL. The necessary size is then stored in **count**.
2. Allocate **attributeInformationArray** with the given size.
3. Call this function again using the previously allocated array.

Parameters

Parameter	Type	Descrip
id	IMAQdxSession	A valid S obtain us IMAQdx
attributeInformationArray	IMAQdxAttributeInformationArray []	Contains attribute which th the came paramet needed l paramet
count	unsigned int (passed by reference)	Contains used to s user pas **attribute** paramet contains
root	const char *	Specifies attribute Specify a enumera tree.
visibility	IMAQdxAttributeVisibility	Specifies attribute attributes visibility Available IMAQdx Specify IMAQdx to return

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxEnumerateAttributeValues

Format

rval = IMAQdxEnumerateAttributeValues(IMAQdxSession id, const char *name, IMAQdxEnumItem list [], unsigned int *size);

Purpose

Gets the values supported by the camera attribute.

Note This function applies only to attributes of type IMAQdxAttributeTypeEnum. Use IMAQdxGetAttributeType to get your attribute type.

If you do not know in advance the number of attribute values, complete the following steps:

1. Call this function with the **list** parameter set to NULL. The necessary size is then stored in **size**.
2. Allocate **list** with the given size.
3. Call this function again using the previously allocated array.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	cont char *	The name of the attribute whose values you want to enumerate. Refer to Attribute Name for a list of attributes.
list	IMAQdxEnumItem []	The list of attribute values for the attribute specified by name. Set this parameter to NULL to get the size needed by the array in the **size** parameter.
size	unsigned int (passed by reference)	The size of attribute values for the attribute specified by name. If the user passes NULL as the **list** parameter, this parameter contains the needed size.

Parameter Discussion

name specifies the attribute name whose value you want to obtain. In the LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxEnumerateCameras

Format

rval = IMAQdxEnumerateCameras (IMAQdxCameraInformation cameraInformationArray[], unsigned int *count, unsigned int connectedOnly);

Purpose

Returns a list of all cameras on the host computer. If you do not know in advance the number of cameras, complete the following steps:

1. Call this function with the **cameraInformationArray** parameter set to NULL. The necessary size is then stored in **count**.
2. Allocate **cameraInformationArray** with the given size.
3. Call this function again using the previously allocated array.

Parameters

Parameter	Type	Description
cameraInformationArray	IMAQdxCameraInformation []	An array of **IMAQdxCamer** structure eleme the interfaces s the system are this parameter t get the size nee array in the **cou** parameter.
count	unsigned int (passed by reference)	The size of the store the camer information. If th passes NULL a **cameraInforma** parameter, this contains the ne
connectedOnly	unsigned int	If the **connecte** is **true**, then the cameraInformat contains camer currently conne host computer. **connectedOnl** **false**, then the cameraInformat contains camer currently conne were previously to the host com

Parameter Discussion

The IMAQdxCameraInformation structure contains information about currently and previously connected interfaces. Once enumerated, check the Flags member of the **IMAQdxCameraInformation** structure. If the value of Flags is 0, the camera is not currently connected. If the value of Flags is 1, the camera is currently connected.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxEnumerateVideoModes

Format

rval = IMAQdxEnumerateVideoModes (IMAQdxSession id, IMAQdxVideoMode videoModeArray[], unsigned int *count, unsigned int *currentMode);

Purpose

Returns a list of video modes supported by the camera.

Note This function applies only to cameras of bus type IMAQdxBusTypeFireWire. Use IMAQdxGetAttribute with attribute IMAQdxAttributeBusType to get your bus type.

If the number of video modes is not known in advance, complete the following steps:

1. Call this function with the **videoModeArray** parameter set to NULL. The necessary size is then stored in **videoModeArraySize**.
2. Allocate the **videoModeArray** with the given size.
3. Call this function again using with the previously allocated array.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
videoModeArray	IMAQdxVideoMode []	Contains an array of video modes supported by the current camera. Set this parameter to NULL to get the size needed by the array in the **count** parameter.
count	unsigned int (passed by reference)	The size of the array used to store the video modes. If the user passes NULL as the **videoModeArray** parameter, this parameter then contains the needed size.
currentMode	unsigned int (passed by reference)	The index into the **videoModeArray** of the current mode used by the camera.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttribute

Format

rval = IMAQdxGetAttribute(IMAQdxSession id, char *name, IMAQdxValueType type, void *value);

Purpose

Gets the current value for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	char *	The name of the attribute whose value you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	void * (passed by reference)	The value of the specified attribute when the function returns.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, IMAQdxValueTypeF64, IMAQdxValueTypeString, IMAQdxValueTypeEnumItem, and IMAQdxValueTypeBool.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeDescription

Format

rval = IMAQdxGetAttributeDescription(IMAQdxSession id, const char *name, char *description, unsigned int length)

Purpose

Gets the description for the camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute for which you want to get the description. Refer to Attribute Name for a list of attributes.
description	char *	A pointer to an area of memory reserved for a tooltip. The reserved memory must be at least the size specified by the **length** parameter.
length	unsigned int	The maximum length of the C string passed as the **description** parameter.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In the LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeDisplayName

Format

rval = IMAQdxGetAttributeDisplayName(IMAQdxSession id, const char *name, char *displayName, unsigned int length)

Purpose

Gets the display name for the camera attribute. The display name is a human readable version of the attribute name.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose tooltip you want to get. Refer to Attribute Name for a list of attributes.
display name	char *	A pointer to an area of memory reserved for a display name. The reserved memory must be at least the size specified by the **length** parameter.
length	unsigned int	The maximum length of the C string passed as the **display name** parameter.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeIncrement

Format

rval = IMAQdxGetAttributeIncrement(IMAQdxSession id, char *name, IMAQdxValueType type, void *value);

Purpose

Gets the increment for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	char *	The name of the attribute whose increment you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	void * (passed by reference)	The increment of the specified attribute when the function returns.

Parameter Discussion

name specifies the attribute whose increment you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, and IMAQdxValueTypeF64.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeMaximum

Format

rval = IMAQdxGetAttributeMaximum(IMAQdxSession id, char *name, IMAQdxValueType type, void *value);

Purpose

Gets the maximum for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	char *	The name of the attribute whose maximum you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	void * (passed by reference)	The maximum of the specified attribute when the function returns.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, and IMAQdxValueTypeF64.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeMinimum

Format

rval = IMAQdxGetAttributeMinimum(IMAQdxSession id, const char *name, IMAQdxValueType type, void *value);

Purpose

Gets the minimum for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose minimum you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	void * (passed by reference)	The minimum of the specified attribute when the function returns.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, and IMAQdxValueTypeF64.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeTooltip

Format

rval = IMAQdxGetAttributeTooltip(IMAQdxSession id, const char *name, char *tooltip, unsigned int length);

Purpose

Gets the tooltip for the camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose tooltip you want to get. Refer to Attribute Name for a list of attributes.
tooltip	char *	A pointer to an area of memory reserved for a tooltip. The reserved memory must be at least the size specified by the **length** parameter.
length	unsigned int	The maximum length of the C string passed as the **tooltip** parameter.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeType

Format

rval = IMAQdxGetAttributeType (IMAQdxSession id, const char *name, IMAQdxAttributeType *type);

Purpose

Gets the attribute type for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose value you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxAttributeType (passed by reference)	The type of the attribute whose value you want to get.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, IMAQdxValueTypeF64, IMAQdxValueTypeString, IMAQdxValueTypeEnumItem, and IMAQdxValueTypeBool.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxGetAttributeUnits

Format

rval = IMAQdxGetAttributeUnits (IMAQdxSession id, const char *name, char *units, unsigned int length);

Purpose

Gets the attribute units for a camera.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose units you want to get. Refer to Attribute Name for a list of attributes.
units	char *	A pointer to an area of memory reserved for an error string. The reserved memory must be at least the size specified by the **length** parameter.
length	unsigned int	The maximum length of the C string passed as the **units** parameter.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetAttributeVisibility

Format

rval = IMAQdxGetAttributeVisibility(IMAQdxSession id, const char *name, IMAQdxAttributeVisibility* visibility)

Purpose

Gets the visibility for the camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can c using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose vis you want to get. Refer to Attribute N a list of attributes.
visibility	IMAQdxAttributeVisibility (passed by reference)	On return contains the visibility for t current attribute. Choose from one following options: • IMAQdxAttributeVisibilitySir • IMAQdxAttributeVisibilityInt • IMAQdxAttributeVisibilityAd

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxGetErrorString

Format

rval = IMAQdxGetErrorString (IMAQdxError error, char *message, unsigned int messageLength);

Purpose

Returns a string describing the error code.

Parameters

Parameter	Type	Description
error	IMAQdxError	A valid NI-IMAQdx error code. Refer to the Error Codes topic in this help file for a complete error code list.
message	char *	A pointer to an area of memory reserved for an error string. The reserved memory must be at least the size specified by the **messageLength** parameter.
messageLength	unsigned int	The maximum length of the C string passed as the **message** parameter.

Return Value

Refer to Error Codes for a complete error code list.

IMAQdxGetImage

Format

rval = IMAQdxGetImage (IMAQdxSession id, Image *image, IMAQdxBufferNumberMode mode, unsigned int desiredBufferNumber, unsigned int *actualBufferNumber);

Purpose

Acquires the specified frame into **image**. Call this function only after calling IMAQdxConfigureAcquisition. If the image type does not match the video format of the camera, the function changes the image type to a suitable format.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, v using IMAQdxOpen
image	Image *	The image that rece pixel data.
mode	IMAQdxBufferNumberMode	The buffer number r retrieve. Set this par IMAQdxBufferNumb the next buffer, or se IMAQdxBufferNumb acquired buffer, or s IMAQdxBufferNumb to acquire a specific number.
desiredBufferNumber	unsigned int	The cumulative buff image to retrieve. Th needed if mode is se IMAQdxBufferNumb
actualBufferNumber	unsigned int (passed by reference)	On return, the actua number of the image

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetImageData

Format

rval = IMAQdxGetImageData (IMAQdxSession id, void *buffer, unsigned int bufferSize, IMAQdxBufferNumberMode mode, unsigned int desiredBufferNumber, unsigned int *actualBufferNumber);

Purpose

Copies the raw data of the specified frame into **buffer**. Call this function only after calling IMAQdxConfigureAcquisition.

Note This function allows you to access raw image data. For many uncompressed formats like YUV or RGB, **buffer** is not compatible with NI Vision. To use the NI Vision functions, use IMAQdxGetImage instead of this function.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, v using IMAQdxOpen
buffer	void *	The buffer that conta the image when the
bufferSize	unsigned int	The maximum size
mode	IMAQdxBufferNumberMode	The buffer number r retrieve. Set this par IMAQdxBufferNumb the next buffer, or se IMAQdxBufferNumb acquired buffer, or s IMAQdxBufferNumb to acquire a specific number.
desiredBufferNumber	unsigned int	The cumulative buff image to retrieve. Th needed if mode is se IMAQdxBufferNumb
actualBufferNumber	unsigned int (passed by reference)	On return, the actua number of the image

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGrab

Format

rval = IMAQdxGrab (IMAQdxSession id, Image *image, unsigned int waitForNextBuffer, unsigned int *actualBufferNumber);

Purpose

Acquires the most current frame into **image**. Call this function only after calling IMAQdxConfigureGrab. If the image type does not match the video format of the camera, this function changes the image type to a suitable format.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
image	Image *	The image that receives the captured pixel data.
waitForNextBuffer	unsigned int	If the **waitForNextBuffer** value is **true**, the driver will wait for the next available buffer. If the **waitForNextBuffer** value is **false**, the driver will not wait for the next available buffer, and will instead return the last acquired buffer.
actualBufferNumber	unsigned int (passed by reference)	On return, the actual cumulative buffer number of the image retrieved.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxIsAttributeReadable

Format

rval = IMAQdxIsAttributeReadable (IMAQdxSession id, const char *name, unsigned int *readable);

Purpose

Gets the read permissions for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the camera attribute whose read permission you want to get. Refer to Attribute Name for a list of attributes.
readable	unsigned int (passed by reference)	Returns true if the attribute is readable, otherwise false.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxIsAttributeWritable

Format

rval = IMAQdxIsAttributeWritable (IMAQdxSession id, const char *name, unsigned int *writable);

Purpose

Gets the write permissions for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the camera attribute whose write permission you want to get. Refer to Attribute Name for a list of attributes.
writable	unsigned int (passed by reference)	Returns true if the attribute is writable, otherwise false.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxOpenCamera

Format

rval = IMAQdxOpenCamera (const char *name, IMAQdxCameraControlMode mode, IMAQdxSession *id);

Purpose

Opens a camera, queries the camera for its capabilities, loads a camera configuration file, and creates a unique reference to the camera. Use IMAQdxCloseCamera when you are finished with the reference.

Parameters

Parameter	Type	Description
name	const char *	The name of the camera you want to open. **name** (cam0, cam1, ..., cam*N*) must match the configuration file name you use to configure the camera in MAX. You can also open a camera using its 64-bit serial number (uuid:XXXXXXXXXXXXXXXX where the number following uui must be a 64-bit hexadecimal number representing the intern serial number of the camera. **Note** Specify " uuid:*serial number in hexadecimal representation*" for the camera name when opening in listening mode. The serial number must match the serial number used in MAX.
mode	IMAQdxCameraControlMode	**Camera Control Mode** is the control mode of the camera used during image broadcastin Open a camera in controller mode to actively configure and acquire image data. Open a camera in listener mode to passively acquire image data from a session that was opene in controller mode on a differen host or target computer. The default value is **Controller**.
id	IMAQdxSession (passed by reference)	On return, a valid Session ID.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxReadAttributes

Format

rval = IMAQdxReadAttributes (IMAQdxSession id, const char* filename)

Purpose

Reads attributes from file and applies to current session. This function is only required if you wish to load parameters. By default the attributes are loaded from file when the camera is opened.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
Filename	const char*	The filename to load the attributes from. Specify NULL to load from the default camera file.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxReadMemory

Format

rval = IMAQdxReadMemory (IMAQdxSession id, unsigned int offset, const char *values, unsigned int count);

Purpose

Accesses registers on the camera and reads a string from the camera.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
offset	unsigned int	The register location to access. Refer to the camera documentation for more information about camera-specific register ranges. Use attribute IMAQdxAttributeBaseAddress to obtain the base address for the camera.
values	const char *	Specifies the string read from the memory offset.
count	unsigned int	Specifies the maximum length of the string read from the memory offset.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxReadRegister

Format

rval = IMAQdxReadRegister (IMAQdxSession id, unsigned int offset, unsigned int *value);

Purpose

Accesses registers on the camera and reads a 32-bit value from the camera. Data is byte-swapped for little endian alignment after transfer.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
offset	unsigned int	The register location to access. Refer to the camera documentation for more information about camera-specific register ranges. Use attribute IMAQdxAttributeBaseAddress to obtain the base address for the camera.
value	unsigned int (passed by reference)	Specifies the value to read from the memory offset.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxRegisterFrameDoneEvent

Format

rval = IMAQdxRegisterFrameDoneEvent (IMAQdxSession id, unsigned int bufferInterval, FrameDoneEventCallbackPtr callbackFunction, void *callbackData);

Purpose

Configures the NI-IMAQdx driver to execute a callback function when a frame done event occurs.

Note Make sure that the code inside the callback is thread safe since the callback executes in a different thread.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
bufferInterval	unsigned int	The number of images to acquire before executing the callback function. Specify a buffer interval of 1 to receive a callback for every buffer.
callbackFunction	FrameDoneEventCallbackPtr	The address of the callback function.
callbackData	void *	A pointer to user-defined data passed to the event function.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxRegisterPnpEvent

Format

rval = IMAQdxRegisterPnpEvent (IMAQdxSession id, IMAQdxPnpEvent event, PnpEventCallbackPtr callbackFunction, void *callbackData);

Purpose

Configures the NI-IMAQdx driver to execute a callback function when a plug and play event occurs.

Note Make sure that the code inside the callback is thread safe since the callback executes in a different thread.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you c using IMAQdxOpenCamera.
event	IMAQdxPnpEvent	The plug and play event to mor following events are valid: – IMAQdxPnpEventCame Callback fired when a n is attached. – IMAQdxPnpEventCame Callback fired when the detached. – IMAQdxPnpEventBusR Callback fired when a F reset occurs.
callbackFunction	PnpEventCallbackPtr	The address of the callback fun
callback Data	void *	A pointer to user-defined data p event function.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxResetEthernetCameraAddress

Format

rval = IMAQdxError ResetEthernetCameraAddress(const char* name, const char* address, const char* subnet, const char* gateway, u32 timeout)

Purpose

Force a new static IP address for the specified camera. Use IMAQdxResetEthernetCameraAddress when the camera is configured for a different subnet than your network. The code execution will suspend for the current thread and will resume after the specified timeout or as soon as it completes. Call this function before calling IMAQdxDiscoverEthernetCameras. Resetting the Ethernet Address is optional for cameras not on the local subnet.

Parameters

Parameter	Type	Description
name	const char *	The name of the camera you want to open. **name** (cam0, cam1, ..., cam*N*) must match the configuration file name you used to configure the camera in MAX. You can also open a camera using its 64-bit serial number (uuid:XXXXXXXXXXXXXXXX), where the number following uuid must be a 64-bit hexadecimal number representing the internal serial number of the camera. **Note** Specify "uuid:*serial number in hexadecimal representation*" for the camera name when opening in listening mode. The serial number must match the serial number used in MAX.
address	const char*	Network address for the camera.
subnet	const char*	Subnet mask for the camera.
gateway	const char*	Gateway for the camera.
timeout	unsigned int	Time, in milliseconds, allowed for the Ethernet camera to reset its network address. The default timeout is 1000 ms.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxResetCamera

Format

rval = IMAQdxResetCamera (const char *name, unsigned int resetALL);

Purpose

Performs a manual reset on a camera. Stops any ongoing acquisitions.

Parameters

Parameter	Type	Description
name	const char *	The name of the camera you want to open. **name** (cam0, cam1, ..., cam*N*) must match the configuration file name you used to configure the camera in MAX. You can also open a camera using its 64-bit serial number (uuid:XXXXXXXXXXXXXXXX), where the number following uuid must be a 64-bit hexadecimal number representing the internal serial number of the camera. **Note** Specify "uuid:*serial number in hexadecimal representation*" for the camera name when opening in listening mode. The serial number must match the serial number used in MAX.
resetALL	unsigned int	If the **resetALL** value is **false**, then only the specified camera will be reset. If the **resetALL** value is **true**, then all of the connected cameras will be reset.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxSequence

Format

rval = IMAQdxSequence (IMAQdxSession id, Image *images[], unsigned int count);

Purpose

Configures, starts, acquires, stops, and unconfigures a sequence acquisition. Use this function to capture multiple images. If you call this function before calling IMAQdxOpenCamera, IMAQdxSequence uses cam0 by default.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
images	Image *[]	The image array that receives the captured pixel data.
count	unsigned int	The number of images in the image array. This value must be less than or equal to the number of allocated images in the image array.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxSetAttribute

Format

rval = IMAQdxSetAttribute (IMAQdxSession id, const char *name, IMAQdxValueType type, ...);

Purpose

Sets the value for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	const char *	The name of the attribute you want to set. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the attribute value you want to set.
...	variable argument	Data is passed by value. The data type should match **type**.

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <Spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, IMAQdxValueTypeF64, IMAQdxValueTypeString, IMAQdxValueTypeEnumItem, and IMAQdxValueTypeBool.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxSnap

Format

rval = IMAQdxSnapImage (IMAQdxSession id, Image *image);

Purpose

Configures, starts, acquires, and unconfigures a snap acquisition. Use a snap for low-speed or single-capture applications where ease of programming is essential. If you call this function before calling IMAQdxOpenCamera, IMAQdxSnap uses cam0 by default. If the image type does not match the video format of the camera, this function changes the image type to a suitable format.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
image	Image *	The image that receives the captured pixel data.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxStartAcquisition

Format

rval = IMAQdxStartAcquisition (IMAQdxSession id);

Purpose

Starts an acquisition that was previously configured with IMAQdxConfigureAcquisition. Use IMAQdxStopAcquisition to stop the acquisition.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxStopAcquisition

Format

rval = IMAQdxStopAcquisition (IMAQdxSession id);

Purpose

Stops an acquisition previously started with IMAQdxStartAcquisition.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxUnconfigureAcquisition

Format

rval = IMAQdxUnconfigureAcquisition (IMAQdxSession id);

Purpose

Unconfigures an acquisition previously configured with IMAQdxConfigureAcquisition.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxWriteAttributes

Format

rval = IMAQdxWriteAttributes (IMAQdxSession id, const char* filename)

Purpose

Writes current attributes to the camera file. This function is only required if you wish to save parameters.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID.
Filename	Const char*	The filename to load the attributes from. Specify NULL to load from the default camera file. Specify a valid filename to override the default camera file. The driver locates camera files in the <NI-IMAQdx\Data> folder if no path information is specified.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString or, if you are using Microsoft Visual Basic, IMAQdxGetErrorStringCW.

IMAQdxWriteMemory

Format

rval = IMAQdxWriteMemory (IMAQdxSession id, unsigned int offset, char *value, unsigned int count);

Purpose

Accesses registers on the camera and writes a string to the camera.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
offset	unsigned int	The register location to access. Refer to the camera documentation for more information about camera-specific register ranges. Use attribute IMAQdxAttributeBaseAddress to obtain the base address for the camera.
value	char *	Specifies the string to write to the memory offset.
count	unsigned int	Specifies the length of the string to write to the memory offset.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxWriteRegister

Format

rval = IMAQdxWriteRegister(IMAQdxSession id, unsigned int offset, unsigned int value);

Purpose

Accesses registers on the camera and writes a 32-bit value to the camera. Data is byte-swapped for big endian alignment before transfer.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
offset	unsigned int	The register location to access. Refer to the camera documentation for more information about camera-specific register ranges. Use attribute IMAQdxAttributeBaseAddress to obtain the base address for the camera.
value	unsigned int	Specifies the value to write to the memory offset.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxEnumerateAttributes2CW

Format

IMAQdxEnumerateAttributes2CW(id As IMAQdxSession, attributeInformationArray() As IMAQdxAttributeInformation, root As String, visibility As IMAQdxAttributeVisibility) As IMAQdxError

Purpose

Gets the attributes supported by the camera.

Parameters

Parameter	Type	Descrip
id	IMAQdxSession	A valid S obtain us IMAQdx
attributeInformationArray	IMAQdxAttributeInformationArray []	Contains attribute which th the came
root	String	Specifies attribute Specify a enumera tree.
visibility	IMAQdxAttributeVisibility	Specifies attribute attributes visibility Available IMAQdx Specify IMAQdx to return

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxEnumerateAttributeValuesCW

Format

IMAQdxEnumerateAttributeValuesCW (id As IMAQdxSession, name As String, enumItemArray() As IMAQdxEnumItem) As IMAQdxError

Purpose

Gets the values supported by the camera attribute.

Note This function applies only to attributes of type IMAQdxAttributeTypeEnum. Use IMAQdxGetAttributeType to get your attribute type.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose values you want to enumerate. Refer to Attribute Name for a list of attributes.
enumItemArray	IMAQdxEnumItem []	The list of attribute values for the attribute specified by name.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxEnumerateCamerasCW

Format

IMAQdxEnumerateCamerasCW (cameraInformationArray() As IMAQdxCameraInformation, connectedOnly As VARIANT_BOOL) As IMAQdxError

Purpose

Returns a list of all cameras on the host computer.

Parameters

Parameter	Type	Description
cameraInformationArray	IMAQdxCameraInformation []	An array of **IMAQdxCamer** structure eleme the interfaces s the system are
connectedOnly	VARIANT_BOOL	If the **connecte** is **true**, then the cameraInformat contains camer currently conne host computer. **connectedOnl** **false**, then the cameraInformat contains camer currently conne were previously to the host com

Parameter Discussion

The IMAQdxCameraInformation structure contains information about currently and previously connected interfaces. Once enumerated, check the Flags member of the **IMAQdxCameraInformation** structure. If the value of Flags is 0, the camera is not currently connected. If the value of Flags is 1, the camera is currently connected.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxEnumerateVideoModesCW

Format

IMAQdxEnumerateVideoModesCW(id As IMAQdxSession, videoModeArray() As IMAQdxVideoMode, currentMode As Long) As IMAQdxError

Purpose

Returns a list of video modes supported by the camera.

Note This function applies only to cameras of bus type IMAQdxBusTypeFireWire. Use IMAQdxGetAttributeCW with attribute IMAQdxAttributeBusType to get your bus type.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
videoModeArray	IMAQdxVideoModeArray[]	Contains an array of video modes supported by the current camera.
currentMode	Long (passed by reference)	The index of the current mode used by the camera in **videoModeArray**.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeCW

Format

IMAQdxGetAttributeCW (id As IMAQdxSession, name As String, type As IMAQdxValueType, value As VARIANT) As IMAQdxError

Purpose

Gets the current value for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose value you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	VARIANT (passed by reference)	The value of the specified attributes when the function returns.

Parameter Discussion

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, IMAQdxValueTypeF64, IMAQdxValueTypeString, IMAQdxValueTypeEnumItem, and IMAQdxValueTypeBool.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeDescriptionCW

Format

IMAQdxGetAttributeDescriptionCW (id As IMAQdxSession, name As String, description As String) As IMAQdxError

Purpose

Gets the description for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose description you want to get. Refer to Attribute Name for a list of attributes.
description	String (passed by reference)	The description of the specified attributes when the function returns.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeDisplayNameCW

Format

IMAQdxGetAttributeDisplayNameCW (id As IMAQdxSession, name As String, displayName As String) As IMAQdxError

Purpose

Gets the display name for the camera attribute. The display name is a human readable version of the attribute name.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose display name you want to get. Refer to Attribute Name for a list of attributes.
displayName	String (passed by reference)	The display name of the specified attributes when the function returns.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeIncrementCW

Format

IMAQdxGetAttributeIncrementCW (id As IMAQdxSession, name As String, type As IMAQdxValueType, value As VARIANT) As IMAQdxError

Purpose

Gets the increment for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose increment you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	VARIANT (passed by reference)	The increment of the specified attributes when the function returns.

Parameter Discussion

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, and IMAQdxValueTypeF64.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeMaximumCW

Format

IMAQdxGetAttributeMaximumCW (id As IMAQdxSession, name As String, type As IMAQdxValueType, value As VARIANT) As IMAQdxError

Purpose

Gets the maximum for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose maximum you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	VARIANT (passed by reference)	The maximum of the specified attributes when the function returns.

Parameter Discussion

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, and IMAQdxValueTypeF64.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeMinimumCW

Format

IMAQdxGetAttributeMinimumCW (id As IMAQdxSession, name As String, type As IMAQdxValueType, value As VARIANT) As IMAQdxError

Purpose

Gets the minimum for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose minimum you want to get. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the value you want to get.
value	VARIANT (passed by reference)	The minimum of the specified attributes when the function returns.

Parameter Discussion

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, and IMAQdxValueTypeF64.

Note The value type must be compatible with the attribute type. Refer to the *NI-IMAQdx Help* for more information about camera attributes.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeTooltipCW

Format

IMAQdxGetAttributeTooltipCW (id As IMAQdxSession, name As String, tooltip As String) As IMAQdxError

Purpose

Gets the tooltip for the camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose tooltip you want to get. Refer to Attribute Name for a list of attributes.
tooltip	String (passed by reference)	The tooltip of the specified attributes when the function returns.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeUnitsCW

Format

IMAQdxGetAttributeUnitsCW (id As IMAQdxSession, name As String, unit As String) As IMAQdxError

Purpose

Gets the attribute units for a camera.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute whose units you want to get. Refer to Attribute Name for a list of attributes.
units	String (passed by reference)	The units of the specified attributes when the function returns.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetAttributeVisibility

Format

rval = IMAQdxGetAttributeVisibility(IMAQdxSession id, const char *name, IMAQdxAttributeVisibility* visibility)

Purpose

Gets the visibility for the camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can c using IMAQdxOpenCamera.
name	const char *	The name of the attribute whose vis you want to get. Refer to Attribute N a list of attributes.
visibility	IMAQdxAttributeVisibility (passed by reference)	On return contains the visibility for t current attribute. Choose from one following options:: • IMAQdxAttributeVisibilitySir • IMAQdxAttributeVisibilityInt • IMAQdxAttributeVisibilityAd

Parameter Discussion

name specifies the attribute whose value you want to obtain. In LabWindows/CVI function panel, when you click the control or press <Enter>, <spacebar>, or <Ctrl-down arrow>, a dialog box opens containing a hierarchical list of the available attributes. Attributes whose values cannot be obtained are dimmed. You can access function help text for each attribute by double-clicking an attribute or by selecting the attribute and pressing <Enter>.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGetErrorStringCW

Format

IMAQdxGetErrorStringCW (errorCode As IMAQdxError, errorMessage As String) As IMAQdxError

Purpose

Returns a string describing the error code.

Parameters

Parameter	Type	Description
errorCode	IMAQdxError	A valid NI-IMAQdx error code. Refer to Error Codes for a complete error code list.
errorMessage	String	The string describing the error that occurred.

Return Value

Refer to Error Codes for a complete error code list.

IMAQdxGetImageCW

Format

IMAQdxGetImageCW (id As IMAQdxSession, image As CWIMAQImage, mode as IMAQdxBufferNumberMode, desiredBufferNumber As Long, actualBufferNumber As Long) As IMAQdxError

Purpose

Acquires the specified frame into **image**. If the image type does not match the video format of the camera, the function changes the image type to a suitable format.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, v with the function IMA
image	CWIMAQImage	The image that rece pixel data.
mode	IMAQdxBufferNumberMode	The buffer number r retrieve. Set this par IMAQdxBufferNumb the next buffer, or se IMAQdxBufferNumb acquired buffer, or s IMAQdxBufferNumb to acquire a specific number.
desiredBufferNumber	Long	The cumulative buff image to retrieve. Th needed if mode is se IMAQdxBufferNumb
actualBufferNumber	Long (passed by reference)	The actual cumulati the image retrieved.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxGetImageDataCW

Format

rval = IMAQdxGetImageDataCW (id As IMAQdxSession, buffer As VARIANT, mode As IMAQdxBufferNumberMode, desiredBufferNumber As Long, actualBufferNumber As Long);

Purpose

Copies the raw data of the specified frame into **buffer**.

Note This function allows you to access raw image data. For many uncompressed formats like YUV or RGB, **buffer** is not compatible with NI Vision. To use the NI Vision functions, use IMAQdxGetImage instead of this function.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, v using IMAQdxOpen
buffer	VARIANT *	The data of the spec the function returns.
mode	IMAQdxBufferNumberMode	The buffer number r retrieve. Set this par IMAQdxBufferNumb the next buffer, or se IMAQdxBufferNumb acquired buffer, or s IMAQdxBufferNumb to acquire a specific number.
desiredBufferNumber	Long	The cumulative buff image to retrieve. Th needed if mode is se IMAQdxBufferNumb
actualBufferNumber	Long (passed by reference)	On return, the actua number of the image

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorString.

IMAQdxGrabCW

Format

IMAQdxGrabCW(id As IMAQdxSession, image As CWIMAQImage, waitForNextBuffer As Long, actualBufferNumber As Long) As IMAQdxError

Purpose

Acquires the most current frame into **image**. Call this function only after calling IMAQdxConfigureGrab. If the image type does not match the video format of the camera, this function changes the image type to a suitable format.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
image	CWIMAQImage	The image that receives the captured pixel data.
waitForNextBuffer	Long	If the **waitForNextBuffer** value is **true**, the driver will wait for the next available buffer. If the **waitForNextBuffer** value is **false**, the driver will not wait for the next available buffer, and will instead return the last acquired buffer.
actualBufferNumber	Long (passed by reference)	On return, the actual cumulative buffer number of the image retrieved.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxIsAttributeReadableCW

Format

IMAQdxIsAttributeReadableCW (id As IMAQdxSession, name As String, readable as VARIANT_BOOL) As IMAQdxError

Purpose

Gets the read permissions for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the camera attribute whose read permission you want to get. Refer to Attribute Name for a list of attributes.
readable	VARIANT_BOOL (passed by reference)	Returns true if the attribute is readable, otherwise false.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxIsAttributeWritableCW

Format

IMAQdxIsAttributeWritableCW (id As IMAQdxSession, name As String, writable as VARIANT_BOOL) As IMAQdxError

Purpose

Gets the write permissions for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the camera attribute whose write permission you want to get. Refer to Attribute Name for a list of attributes.
writable	VARIANT_BOOL (passed by reference)	Returns true if the attribute is writable, otherwise false.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxSequenceCW

Format

IMAQdxSequenceCW (id As IMAQdxSession, images() As CWIMAQImage, count As Long) As IMAQdxError

Purpose

Configures, starts, acquires, stops, and unconfigures a sequence acquisition. Use this function to capture multiple images. If you call this function before calling IMAQdxOpenCamera, IMAQdxSequence uses cam0 by default.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
images	CWIMAQImage []	The image array that receives the captured pixel data.
count	Long	The number of images in the image array. This value must be less than or equal to the number of allocated images in the image array.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxSetAttributeCW

Format

IMAQdxSetAttributeCW (id As IMAQdxSession, name As String, type As IMAQdxValueType, value As VARIANT) As IMAQdxError

Purpose

Sets the value for a camera attribute.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
name	String	The name of the attribute you want to set. Refer to Attribute Name for a list of attributes.
type	IMAQdxValueType	The type of the attribute you want to set.
value	VARIANT (passed by reference)	The value of the specified attribute.

Parameter Discussion

type specifies the type of the value parameter. The following types are supported: IMAQdxValueTypeU32, IMAQdxValueTypeI64, IMAQdxValueTypeF64, IMAQdxValueTypeString, IMAQdxValueTypeEnumItem, and IMAQdxValueTypeBool.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

IMAQdxSnapCW

Format

IMAQdxSnapCW (id As IMAQdxSession, image As CWIMAQImage) As IMAQdxError

Purpose

Configures, starts, acquires, and unconfigures a snap acquisition. Use a snap for low-speed or single-capture applications where ease of programming is essential. If you call this function before calling IMAQdxOpenCamera, IMAQdxSnap uses cam0 by default. If the image type does not match the video format of the camera, this function changes the image type to a suitable format.

Parameters

Parameter	Type	Description
id	IMAQdxSession	A valid Session ID, which you can obtain using IMAQdxOpenCamera.
image	CWIMAQImage	The image that receives the captured pixel data.

Return Value

On success, this function returns IMAQdxErrorSuccess. On failure, this function returns an error code. You can obtain a more detailed error message with IMAQdxGetErrorStringCW.

Attributes by Name

The following table, sorted by attribute name, describes the attributes you can use with the attribute functions.

Attribute Name	
IMAQdxAttributeBaseAddress	CameraInformation::Base/
IMAQdxAttributeBusType	CameraInformation::BusTy
IMAQdxAttributeModelName	CameraInformation::Mode
IMAQdxAttributeSerialNumberHigh	CameraInformation::Serial
IMAQdxAttributeSerialNumberLow	CameraInformation::Serial
IMAQdxAttributeVendorName	CameraInformation::Vendo

IMAQdxAttributeHostIPAddress	CameraInformation::HostII
IMAQdxAttributeIPAddress	CameraInformation::IPAdd
IMAQdxAttributePrimaryURLString	CameraInformation::Prima
IMAQdxAttributeSecondaryURLString	CameraInformation::Secor
IMAQdxAttributeAcqInProgress	StatusInformation::AcqInP
IMAQdxAttributeLastBufferCount	StatusInformation::LastBu
IMAQdxAttributeLastBufferNumber	StatusInformation::LastBu

IMAQdxAttributeLostBufferCount	StatusInformation::LostBu
IMAQdxAttributeLostPacketCount	StatusInformation::LostPa
IMAQdxAttributeRequestedResendPackets	StatusInformation::Reques
IMAQdxAttributeReceivedResendPackets	StatusInformation::Receiv
IMAQdxAttributeBayerGainB	AcquisitionAttributes::Baye

IMAQdxAttributeBayerGainG	AcquisitionAttributes::Baye
IMAQdxAttributeBayerGainR	AcquisitionAttributes::Baye
IMAQdxAttributeBayerPattern	AcquisitionAttributes::Baye
IMAQdxAttributeStreamChannelMode	AcquisitionAttributes::Cont
IMAQdxAttributeDesiredStreamChannel	AcquisitionAttributes::Cont
IMAQdxAttributeFrameInterval	AcquisitionAttributes::Fram

IMAQdxAttributeIgnoreFirstFrame	AcquisitionAttributes::Igno
IMAQdxAttributeOffsetX	OffsetX
IMAQdxAttributeOffsetY	OffsetY
IMAQdxAttributeWidth	Width
IMAQdxAttributeHeight	Height
IMAQdxAttributePixelFormat	PixelFormat
IMAQdxAttributePacketSize	PacketSize
IMAQdxAttributePayloadSize	PayloadSize
IMAQdxAttributeSpeed	AcquisitionAttributes::Spee

IMAQdxAttributeShiftPixelBits	AcquisitionAttributes::Shift
IMAQdxAttributeSwapPixelBytes	AcquisitionAttributes::Swa
IMAQdxAttributeOverwriteMode	AcquisitionAttributes::Over

IMAQdxAttributeTimeout	AcquisitionAttributes::Time
IMAQdxAttributeVideoMode	AcquisitionAttributes::Vide
IMAQdxAttributeBitsPerPixel	AcquisitionAttributes::BitsP
IMAQdxAttributeReserveDualPackets	AcquisitionAttributes::Rese
IMAQdxAttributeReceiveTimestampMode	AcquisitionAttributes::Rece

IMAQdxAttributeActualPeakBandwidth	AcquisitionAttributes::Adva
IMAQdxAttributeDesiredPeakBandwidth	AcquisitionAttributes::Adva
IMAQdxAttributeDestinationMode	AcquisitionAttributes::Adva
IMAQdxAttributeDestinationMulticastAddress	AcquisitionAttributes::Adva
IMAQdxAttributeLostPacketMode	AcquisitionAttributes::Adva

IMAQdxAttributeMemoryWindowSize	AcquisitionAttributes::Adva
IMAQdxAttributeResendsEnabled	AcquisitionAttributes::Adva
IMAQdxAttributeResendThresholdPercentage	AcquisitionAttributes::Adva
IMAQdxAttributeResendBatchingPercentage	AcquisitionAttributes::Adva

IMAQdxAttributeMaxResendsPerPacket	AcquisitionAttributes::Adva
IMAQdxAttributeResendResponseTimeout	AcquisitionAttributes::Adva
IMAQdxAttributeNewPacketTimeout	AcquisitionAttributes::Adva
IMAQdxAttributeMissingPacketTimeout	AcquisitionAttributes::Adva

IMAQdxAttributeResendTimerResolution	AcquisitionAttributes::Adva

Error Codes

The following table describes the error codes used in NI-IMAQdx.

Error Code	Status Name	Descripti
-1074360320	IMAQdxErrorSystemMemoryFull	Not enougl memory
-1074360319	IMAQdxErrorInternal	Internal err
-1074360318	IMAQdxErrorInvalidParameter	Invalid parameter
-1074360317	IMAQdxErrorInvalidPointer	Invalid poir
-1074360316	IMAQdxErrorInvalidInterface	Invalid camera session
-1074360315	IMAQdxErrorInvalidRegistryKey	Invalid registry key
-1074360314	IMAQdxErrorInvalidAddress	Invalid address
-1074360313	IMAQdxErrorInvalidDeviceType	Invalid dev type
-1074360312	IMAQdxErrorNotImplemented	Not implemente yet
-1074360311	IMAQdxErrorCameraNotFound	Camera nc found
-1074360310	IMAQdxErrorCameraInUse	Camera is already in use.
-1074360309	IMAQdxErrorCameraNotInitialized	Camera is initialized.
-1074360308	IMAQdxErrorCameraRemoved	Camera ha been removed.
-1074360307	IMAQdxErrorCameraRunning	Acquisition progress.

-1074360306	IMAQdxErrorCameraNotRunning	No acquisition progress.
-1074360305	IMAQdxErrorAttributeNotSupported	Attribute no supported the camera
-1074360304	IMAQdxErrorAttributeNotSettable	Unable to s attribute.
-1074360303	IMAQdxErrorAttributeNotReadable	Unable to g attribute.
-1074360302	IMAQdxErrorAttributeOutOfRange	Attribute value is ou range.
-1074360301	IMAQdxErrorBufferNotAvailable	Requested buffer is unavailable
-1074360300	IMAQdxErrorBufferListEmpty	Buffer list is empty. Add one or mor buffers.
-1074360299	IMAQdxErrorBufferListLocked	Buffer list is already locked. Reconfigur acquisition and try aga
-1074360298	IMAQdxErrorBufferListNotLocked	No buffer li Reconfigur acquisition and try aga
-1074360297	IMAQdxErrorResourcesAllocated	Transfer engine resources already allocated. Reconfigur

		acquisition and try aga
-1074360296	IMAQdxErrorResourcesUnavailable	Insufficient transfer engine resources.
-1074360295	IMAQdxErrorAsyncWrite	Unable to perform asynchron register wri
-1074360294	IMAQdxErrorAsyncRead	Unable to perform asynchron register rea
-1074360293	IMAQdxErrorTimeout	Timeout
-1074360292	IMAQdxErrorBusReset	Bus reset occurred during a transaction
-1074360291	IMAQdxErrorInvalidXML	Unable to load camer XML file.
-1074360290	IMAQdxErrorFileAccess	Unable to read/write f file.
-1074360289	IMAQdxErrorInvalidCameraURLString	Camera ha malformed URL string.
-1074360288	IMAQdxErrorInvalidCameraFile	Invalid camera file
-1074360287	IMAQdxErrorGenICamError	Unknown Genicam error.
-1074360286	IMAQdxErrorFormat7Parameters	For format The

		combinatio of speed, image position, image size and color coding is incorrect.
-1074360285	IMAQdxErrorInvalidAttributeType	The attribu type is not compatible with the passed variable typ
-1074360284	IMAQdxErrorDLLNotFound	The DLL could not b found.
-1074360283	IMAQdxErrorFunctionNotFound	The functio could not b found.
-1074360282	IMAQdxErrorLicenseNotActivated	License no activated.
-1074360281	IMAQdxErrorCameraNotConfiguredForListener	The camer is not configured properly to support a listener.
-1074360280	IMAQdxErrorCameraMulticastNotAvailable	Unable to configure tl system for multicast support.
-1074360279	IMAQdxErrorBufferHasLostPackets	The requested buffer has lost packet

		and the us requested error to be generated.
-1074360278	IMAQdxErrorGiGEVisionError	Unknown GiGE Visio error.
-1074360277	IMAQdxErrorNetworkError	Unknown network er
-1074360276	IMAQdxErrorCameraUnreachable	Unable to connect to the camera
-1074360275	IMAQdxErrorHighPerformanceNotSupported	High performanc acquisition not suppor on the specified network interface. Connect th camera to network interface running the high performanc driver.

Glossary

A B C D E F G H I L M N O P Q R S T U V W Y

A

A/D	Analog-to-digital.
AC	Alternating current.
acquisition window	The image size specific to a video standard or camera resolution.
active line region	The region of lines actively being stored; defined by a line start (relative to vertical sync signal) and a line count.
active pixel region	The region of pixels actively being stored; defined by a pixel start (relative to the horizontal sync signal) and a pixel count.
ADC	Analog-to-digital converter. An electronic device, often an integrated circuit, that converts an analog voltage to a digital number.
address	Character code that identifies a specific location (or series of locations) in memory.
ANSI	American National Standards Institute.
antichrominance filter	Removes the color information from the video signal.
API	Application programming interface.
area	A rectangular portion of an acquisition window or frame that is controlled and defined by software.
array	Ordered, indexed set of data elements of the same type.
ASIC	Application-specific integrated circuit. A proprietary semiconductor component designed and manufactured to perform a set of specific functions for a specific customer.
aspect ratio	The ratio of a picture or image's width to its height.
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

back porch	The area of the video signal between the rising edge of the horizontal sync signal and the active video information.
Bayer encoding	Method to produce color images with a single imaging sensor, as opposed to three individual sensors for the red, green, and blue components of light.
Bayer pattern	Color filter array pattern that can appear in four variations, depending on the current left and top offsets of the acquisition window: GBGB GRGR BGBG RGRG RGRG BGBG GRGR GBGB
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.
black reference level	The level that represents the darkest an image can get. **See also** white reference level.
BMP	Bitmap. Image file format commonly used for 8-bit and color images (extension .bmp).
buffer	Temporary storage for acquired data.
bus	The group of conductors that interconnect individual circuitry in a computer, such as the PCI bus; typically the expansion vehicle to which I/O or other devices are connected.

C

cache	High-speed processor memory that buffers commonly used instructions or data to increase processing throughput.
camera session	A process-safe handle to a camera.
CCIR	Comite Consultatif International des Radiocommunications. A committee that developed standards for color video signals.
chrominance	The color information in a video signal.
CMOS	Complementary metal-oxide semiconductor.
CompactPCI	Refers to the core specification defined by the PCI Industrial Computer Manufacturer's Group (PICMG).
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.
CSYNC	Composite sync signal. A combination of the horizontal and vertical sync pulses.

D

D/A	Digital-to-analog.
DAC	Digital-to-analog converter; an electronic device, often an integrated circuit, that converts a digital number into a corresponding analog voltage or current.
DAQ	Data acquisition. (1) Collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing. (2) Collecting and measuring the same kinds of electrical signals with A/D or DIO devices plugged into a computer, and possibly generating control signals with D/A and/or DIO devices in the same computer.
DC	Direct current.
default setting	A default parameter value recorded in the driver; in many cases, the default input of a control is a certain value (often 0) that means use the current default setting.
DLL	Dynamic link library. A software module in Microsoft Windows containing executable code and data that can be called or used by Windows applications or other DLLs; functions and data in a DLL are loaded and linked at run time when they are referenced by a Windows application or other DLLs.
DMA	Direct memory access. A method by which data can be transferred to and from computer memory from and to a device or memory on the bus while the processor does something else; DMA is the fastest method of transferring data to/from computer memory.
DRAM	Dynamic RAM.
driver	Software that controls a specific hardware device such as an image acquisition device.
dynamic range	The ratio of the largest signal level a circuit can handle to the smallest signal level it can handle (usually taken to be the noise level), normally expressed in decibels.

E

EEPROM	Electrically erasable programmable read-only memory. ROM that can be erased with an electrical signal and reprogrammed.
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

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 of the odd-numbered lines, and the second field contains all of the even-numbered lines.
FIFO	First-in first-out memory buffer. The first data stored is the first data sent to the acceptor; FIFO buffers are used on image acquisition devices to temporarily store incoming data until that data can be retrieved.
flash ADC	An ADC whose output code is determined in a single step by a bank of comparators and encoding logic.
frame	A complete image; in interlaced formats, a frame is composed of two fields.
front porch	The area of a video signal between the start of the horizontal blank and the start of the horizontal sync.
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

gain	Applied value to compensate for discrepancies in the filter for a particular color.
gamma	The nonlinear change in the difference between the video signal's brightness level and the voltage level needed to produce that brightness.
genlock	Circuitry that aligns the video timing signals by locking together the horizontal, vertical, and color subcarrier frequencies and phases and generates a pixel clock to clock pixel data into memory for display or into another circuit for processing.
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.
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, cables, and so on.
hardware abstraction layer	Separates software API capabilities, such as general acquisition and control functions, from hardware-specific information.
HSYNC	Horizontal sync signal. The synchronization pulse signal produced at the beginning of each video scan line that keeps a video monitor's horizontal scan rate in step with the transmission of each new line.
hue	Represents the dominant color of a pixel. The hue function is a continuous function that covers all the possible colors generated using the R, G, and B primaries. **See also** RGB.

I

I/O	Input/output. The transfer of data to/from a computer system involving communications channels, operator interface devices, or data acquisition and control interfaces.
IEEE	Institute of Electrical and Electronics Engineers.
INL	Integral nonlinearity. A measure, in LSB, of the worst-case deviation from the ideal A/D or D/A transfer characteristic of the analog I/O circuitry.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that controls specific plug-in computer boards; instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW.
interlaced	A video frame composed of two interleaved fields; the number of lines in a field are half the number of lines in an interlaced frame.
internal buffer	A page-locked buffer. **See also** page-locked buffer.
interpreter	A software utility that executes source code from a high-level language, such as Java or Basic, by reading one line at a time and executing the specified operation. In contrast, a compiler converts all source code to executable machine code before execution. Compiled languages give significantly higher performance than interpreted languages. Examples of compiled languages are C, C++, and LabVIEW, while Java and Basic are generally interpreted languages. **See also** compiler.
interrupt	A computer signal indicating that the CPU should suspend its current task to service a designated activity.
interrupt level	The relative priority at which a device can interrupt.
IRE	A relative unit of measure (named for the Institute of Radio Engineers). 0 IRE corresponds to the blanking level of a video signal, 100 IRE to the white level. Note that for CIR/PAL video the black level is equal to the blanking level or 0 IRE, while for RS-170/NTSC video, the black level is at 7.5 IRE.
IRQ	Interrupt request. **See also** interrupt.

L

library	A file containing compiled object modules, each comprised of one or 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.
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.
LSB	Least significant bit.
luminance	The brightness information in the video picture. The luminance signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.
LUT	Lookup table. A selection in Measurement & Automation Explorer (MAX) for Vision that contains formulas that let you implement simple imaging operations such as contrast enhancement, data inversion, gamma manipulation, or other nonlinear transfer functions.

M

MAX	Measurement & Automation Explorer. The National Instruments Windows-based graphical configuration utility you can use to configure NI software and hardware, execute system diagnostics, add new channels and interfaces, and view the devices and instruments you have connected to your computer. MAX is installed on the desktop during the National Instruments driver software installation.
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 image acquisition hardware.
NI-IMAQdx	National Instruments driver software for IEEE 1394 and GigE Vision cameras.
noninterlaced	A video frame where all the lines are scanned sequentially, rather than being divided into two frames as in an interlaced video frame.
NTSC	National Television Standards Committee. The committee that developed the color video standard used primarily in North America, which uses 525 lines per frame. **See also** PAL.
NVRAM	Nonvolatile RAM. RAM that is not erased when a device loses power or is turned off.

O

one-shot	Applies to pulse generation and acquisitions. A one-shot pulse or acquisition happens only once.

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.
PAL	Phase Alternation Line. One of the European video color standards; uses 625 lines per frame. **See also** NTSC.
PCI	Peripheral Component Interconnect. A high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA; it is achieving widespread acceptance as a standard for PCs and workstations and offers a theoretical maximum transfer rate of 133 Mbytes/s.
PCIe	PCI Express. A high-performance expansion bus architecture originally developed by Intel to replace PCI. PCIe offers a theoretical maximum transfer rate that is dependent upon lane width. A x1 link theoretically provides 250 MB/s in each direction—to and from the device. Once overhead is accounted for, a x1 link can provide approximately 200 MB/s of input capability and 200 MB/s of output capability. Increasing the number of lanes in a link increases maximum throughput by approximately the same factor.
PCLK	Pixel clock signal. Times the sampling of pixels on a video line.
PGIA	Programmable gain instrumentation amplifier.
picture aspect ratio	The ratio of the active pixel region to the active line region; for standard video signals such as RS-170 or CCIR, the full-size picture aspect ratio typically 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; however, typically it falls between 0.95 and 1.05, depending on camera quality.
pixel clock	Divides the incoming horizontal video line into pixels.
pixel count	The total number of pixels between two horizontal sync signals; the pixel count determines the frequency of the pixel

Q

quadlet	A 32-bit (four-byte) word.
quadrature encoder	An encoding technique for a rotating device where two tracks of information are placed on the device, with the signals on the tracks offset by 90 degrees from each other. The phase difference indicates the position and direction of rotation.

R

RAM	Random-access memory.
real time	A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.
relative accuracy	A measure in LSB of the accuracy of an ADC; it includes all nonlinearity and quantization errors but does not include offset and gain errors of the circuitry feeding the ADC.
resolution	The smallest signal increment that can be detected by a measurement system; resolution can be expressed in bits, in proportions, or in percent of full scale. For example, a system has 12-bit resolution, one part in 4,096 resolution, and 0.0244 percent of full scale.
RGB	Red, green, and blue. The three primary colors used to represent a color picture. An RGB camera is a camera that delivers three signals, one for each primary.
ribbon cable	A flat cable in which the conductors are side by side.
ring	Performs an acquisition that loops continually on a specified number of buffers.
ROI	Region of interest. (1) An area of the image that is graphically selected from a window displaying the image. This area can be used focus further processing; (2) A hardware-programmable rectangular portion of the acquisition window.
ROM	Read-only memory.
RS-170	The U.S. standard used for black-and-white television.
RTSI bus	Real-Time System Integration Bus. The National Instruments timing bus that connects image acquisition and DAQ devices directly, by means of connectors on top of the devices, for precise synchronization of functions.

S

saturation	The richness of a color. A saturation of zero corresponds to no color, that is, a gray pixel. Pink is a red with low saturation.
scaling down circuitry	Circuitry that scales down the resolution of a video signal.
scatter-gather DMA	A type of DMA that allows the DMA controller to reconfigure on-the-fly.
sequence	Performs an acquisition that acquires a specified number of buffers, then stops.
snap	Acquires a single frame or field to a buffer.
SRAM	Static RAM.
StillColor	A post-processing algorithm that allows the acquisition of high-quality color images generated either by an RGB or composite (NTSC or PAL) camera using a monochrome video acquisition device.
sync	Tells the display where to put a video picture; the horizontal sync indicates the picture's left-to-right placement and the vertical sync indicates top-to-bottom placement.
syntax	Set of rules to which statements must conform in a particular programming language.
system RAM	RAM installed on a personal computer and used by the operating system, as contrasted with onboard RAM.

T

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 setup operations; the maximum rate at which the hardware can operate.
trigger	Any event that causes or starts some form of data capture.
trigger control and mapping circuitry	Circuitry that routes, monitors, and drives the external and RTSI bus trigger lines; you can configure each of these lines to start or stop acquisition on a rising or falling edge.
TTL	Transistor-transistor logic. A digital circuit composed of bipolar transistors wired in a certain manner. A typical medium-speed digital technology. Nominal TTL logic levels are 0 and 5 V.

U

user buffer	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

VCO — Voltage-controlled oscillator. An oscillator that changes frequency depending on a control signal; used in a PLL to generate a stable pixel clock.

VI — Virtual Instrument.

1. A combination of hardware and/or software elements, typically used with a PC, that has the functionality of a classic stand-alone instrument
2. A LabVIEW software module (VI), which consists of a front panel user interface and a block diagram program.

video line — A video line consists of a horizontal sync signal, back porch, active pixel region, and a front porch.

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

W

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

Y

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

Important Information

Warranty

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

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

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

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action

accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the patents.txt file on your media, 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.

Technical Support and Professional Services

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

- Support—Technical support resources at ni.com/support include the following:
  - **Self-Help Resources**—For answers and solutions, visit ni.com/support 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. Registered users also receive access to the NI Discussion Forums at ni.com/forums. NI Applications Engineers make sure every question submitted online receives an answer.
  - **Standard Service Program Membership**—This program entitles members to direct access to NI Applications Engineers via phone and email for one-to-one technical support, as well as exclusive access to on demand training modules via the Services Resource Center. NI offers complementary membership for a full year after purchase, after which you may renew to continue your benefits.

    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. 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.

Branch Offices

Office	Telephone Number
Australia	1800 300 800
Austria	43 662 457990-0
Belgium	32 (0) 2 757 0020
Brazil	55 11 3262 3599
Canada	800 433 3488
China	86 21 5050 9800
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	358 (0) 9 725 72511
France	33 (0) 1 57 66 24 24
Germany	49 89 7413130
India	91 80 41190000
Israel	972 0 3 6393737
Italy	39 02 41309277
Japan	0120-527196 / 81 3 5472 2970
Korea	82 02 3451 3400
Lebanon	961 (0) 1 33 28 28
Malaysia	1800 887710
Mexico	01 800 010 0793
Netherlands	31 (0) 348 433 466
New Zealand	0800 553 322
Norway	47 (0) 66 90 76 60
Poland	48 22 3390150
Portugal	351 210 311 210
Russia	7 495 783 6851
Singapore	1800 226 5886
Slovenia	386 3 425 42 00

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