



# NI-IMAQ I/O Visual Basic Reference Help

July 2007, 371554B-01

The NI-IMAQ I/O Visual Basic Reference Help contains reference information for developing Visual Basic code to control IMAQ I/O devices.

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window.

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

[Conventions](#)—formatting and typographical conventions in this help file

[Important Information](#)


[Technical Support and Professional Services](#)

To comment on National Instruments documentation, refer to the [National Instruments Web site](#).

© 2005—2007 National Instruments Corporation. All rights reserved.


# 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 on 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.
- italic* Italic text denotes variables or cross references. This font also denotes text that is a placeholder for a word or value that you must supply.
- monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.

## 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.

## NI-IMAQ I/O Open/Close Methods

Use the NI-IMAQ I/O open and close methods to open and close sessions on an NI-IMAQ I/O device.

[imaqIOOpen](#)

[imaqIOClose](#)

# imaqIOOpen Method

## Syntax

**imaqIOOpen** deviceName, id

## **Purpose**

Opens a reference to an NI-IMAQ I/O device.



## Parameters

**id** As [IMAQIO\\_SESSION](#)

(Output Parameter) The ID of the session to associated with the **deviceName**.

**deviceName** As [String](#)

The name of the device to open a session on. This is the VISA name that is defined in Measurement & Automation Explorer (MAX). This name is similar to RIO#::INSTR, where # represents the number of the RIO device, depending on how many RIO devices are installed in the system.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOClose](#)

# **imaqIOClose Method**

## **Syntax**

**imaqIOClose id**

## **Purpose**

Closes a reference to an NI-IMAQ I/O device.

## Parameters

**id** As IMAQIO\_SESSION

ID of the session to close.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOOpen](#)



## NI-IMAQ I/O Static I/O Methods

Use the NI-IMAQ I/O static I/O methods to manipulate the signals on an NI-IMAQ I/O device.

[imaqIOGetAttribute](#)

[imaqIOSetAttribute](#)

[imaqIODrive](#)

[imaqIORead](#)

# **imaqIOGetAttribute Method**

## **Syntax**

**imaqIOGetAttribute** id, attribute, value

## **Purpose**

Reads one of the NI-IMAQ I/O device attributes.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to get an attribute for.

**attribute** As [IMAQIOAttribute](#)

The attribute to retrieve.

**value** As [Long](#)

(Output Parameter) The value of the attribute retrieved.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOSetAttribute](#)

## **imaqIODrive Method**

### **Syntax**

**imaqIODrive** id, signalType, signalNumber, value

## **Purpose**

Drives general-purpose digital outputs with the static I/O on the NI-IMAQ I/O device.



## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to drive.

**signalType** As [IMAQIOSignalType](#)

The type of signal to drive.

**signalNumber** As [Long](#)

The line number of the signal you want to drive.

**value** As [IMAQIOSignalState](#)

The state you want to drive the line to.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOGetShutdownState](#)

[imaqIOSetShutdownState](#)

# **imaqIOSetAttribute Method**

## **Syntax**

**imaqIOSetAttribute** id, attribute, value

## Purpose

Sets an attribute on the NI-IMAQ I/O device.



**Note** This method is reserved for future use.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to set an attribute for.

**attribute** As [IMAQIOAttribute](#)

The attribute to set.



**Note** All of the valid values for the **attribute** are currently not settable.

**value** As [Long](#)

The value to set the attribute to.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOGetAttribute](#)



## **imaqIORead Method**

### **Syntax**

**imaqIORead** id, signalType, signalNumber, value

## **Purpose**

Reads general-purpose digital inputs from the static I/O of the NI-IMAQ I/O device.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to read from.

**signalType** As [IMAQIOSignalType](#)

The type of signal to read from.

**signalNumber** As [Long](#)

The line number of the signal to read from.

**value** As [IMAQIOSignalState](#)

(Output Parameter) The signal state read from the NI-IMAQ I/O device.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imglOGetErrorTextVB](#).

## NI-IMAQ I/O Pulse Generator Methods

Use the NI-IMAQ I/O pulse generator methods to manipulate pulses on the signals of the NI-IMAQ I/O device.

[imaqIOPulseCreate](#)

[imaqIOPulseStart](#)

[imaqIOPulseStop](#)

[imaqIOPulseDispose](#)

# imaqIOPulseCreate Method

## Syntax

**imaqIOPulseCreate** id, delayTimebase, delay, widthTimebase, width, triggerSignalType, triggerSignalNumber, triggerSignalPolarity, outputSignalType, outputSignalNumber, outputSignalPolarity, pulseMode, pulseID

## **Purpose**

Configures one of the six pulse generators for the NI-IMAQ I/O device. Refer to your hardware documentation for more information about connecting and configuring the pulse generators.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to create a pulse on.

**delayTimebase** As [IMAQIOTimebase](#)

The timebase of the pulse delay.

**delay** As [Long](#)

The pulse delay.

**widthTimebase** As [IMAQIOTimebase](#)

The timebase of the pulse width.

**width** As [Long](#)

The width of the pulse.

**triggerSignalType** As [IMAQIOSignalType](#)

The type of signal to trigger on.

**triggerSignalNumber** As [Long](#)

The line number of the signal to trigger on. If you set **triggerSignalType** to IMAQIO\_SIGNAL\_STATUS, use [IMAQIO\\_STATUS\\_SIGNAL\\_NONE](#).

**triggerSignalPolarity** As [IMAQIOPolarity](#)

The polarity of the trigger.

**outputSignalType** As [IMAQIOSignalType](#)

The type of signal to create a pulse on.

**outputSignalNumber** As [Long](#)

The line number of the signal to create the pulse on.

**outputSignalPolarity** As [IMAQIOPolarity](#)

The polarity of the pulse output.

**pulseMode** As [IMAQIOPulseMode](#)

The value that indicates if the pulse is generated once or continuously.

**pulseID** As [IMAQIO\\_PULSE\\_ID](#)

(Output Parameter) The variable to hold the pulse ID. If the method



succeeds, the variable is populated with a valid PULSE\_ID that you can use in subsequent methods.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOPulseDispose](#)

[imaqIOPulseStart](#)

[imaqIOPulseStop](#)

# **imaqIOPulseStart Method**

## **Syntax**

**imaqIOPulseStart** id, pulseID

## **Purpose**

Starts a pulse generator session for the NI-IMAQ I/O device.

## Parameters

**id** As IMAQIO\_SESSION

The ID of the session to start a pulse on.

**pulseID** As IMAQIO\_PULSE\_ID

The pulse to start.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOPulseCreate](#)

[imaqIOPulseDispose](#)

[imaqIOPulseStop](#)



# **imaqIOPulseStop Method**

## **Syntax**

**imaqIOPulseStop** id, pulseID

## **Purpose**

Stops a pulse generator session for the NI-IMAQ I/O device.

## Parameters

**id** As IMAQIO\_SESSION

The ID of the session to stop a pulse on.

**pulseID** As IMAQIO\_PULSE\_ID

The pulse to stop.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOPulseCreate](#)

[imaqIOPulseDispose](#)

[imaqIOPulseStart](#)

# **imaqIOPulseDispose Method**

## **Syntax**

**imaqIOPulseDispose** id, pulseID

## **Purpose**

Disposes of a pulse ID, relinquishing its resources.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to dispose a pulse on.

**pulseID** As [IMAQIO\\_PULSE\\_ID](#)

The pulse to dispose of.



## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOPulseCreate](#)

[imaqIOPulseStart](#)

[imaqIOPulseStop](#)

## Change Detectors

Use the NI-IMAQ I/O change detector functions to detect signal state changes on the various input lines of the NI-IMAQ I/O device.

[imaqIOChangeDetectConfigure](#)

[imaqIOChangeDetectQuery](#)

[imaqIOChangeDetectReset](#)

# imaqIOChangeDetectConfigure

## Syntax

imaqIOChangeDetectConfigure, id, triggerType, triggerNumber, detectMode,  
filter

## **Purpose**

Configures the signal state change detector logic of an input line on the NI-IMAQ I/O device. Use this function to arm the detection of rising edges, falling edges, or both on the specified input trigger line.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to enable or disable shutdown on.

**triggerType** As [IMAQIOSignalType](#)

The type of trigger line to detect a change on.

**triggerNumber** As [IMAQIOSignalType](#)

The trigger number to detect a change on.

**detectMode** As [IMAQIOChangeDetectMode](#)

The change detection mode. This specifies what type of edge to be sensitive to.

**filter** As [Long](#)

Used to suppress high-frequency noise from the change detection. After receiving an edge, the trigger line must be held steady in the new logic state for  $\text{filter} \times (25) \text{ ns}$  in order for the device to register a successful change detection. Set this parameter to 0 to disable filtering.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[Change Detectors](#)

[imaqIOChangeDetectQuery](#)

[imaqIOChangeDetectReset](#)



# **imaqIOChangeDetectQuery**

## **Syntax**

imaqIOChangeDetectQuery, id, triggerType, triggerNumber, changed

## Purpose

Queries the NI-IMAQ I/O device for information about whether the given trigger line has changed. For this function to behave properly, you must have previously configured the change detector for the same trigger line using [imaqIOChangeDetectConfigure](#). After a change has been registered for a trigger line, this function will continually report that the trigger line has changed until you reset it by calling [imaqIOChangeDetectReset](#).

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to enable or disable shutdown on.

**triggerType** As [IMAQIOSignalType](#)

The type of trigger line to detect a change on.

**triggerNumber** As [IMAQIOSignalType](#)

The trigger number to detect a change on.

**changed** As [Long](#)

A pointer to a variable to receive the trigger changed information. If no change was detected, the variable will be set to 0. Otherwise, the variable will be set to 1.

---

## See Also

[Change Detectors](#)

[imaqIOChangeDetectConfigure](#)

[imaqIOChangeDetectRead](#)

# **imaqIOChangeDetectReset**

## **Syntax**

imaqIOChangeDetectReset, id, triggerType, triggerNumber

## **Purpose**

Resets the change detection mechanism for the given trigger line. This rearms the change detector logic for detection of a new trigger edge.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to enable or disable shutdown on.

**triggerType** As [IMAQIOSignalType](#)

The type of trigger line to detect a change on.

**triggerNumber** As [IMAQIOSignalType](#)

The trigger number to detect a change on.

---

## See Also

[Change Detectors](#)

[imaqIOChangeDetectConfigure](#)

[imaqIOChangeDetectQuery](#)



## NI-IMAQ I/O Watchdog Methods

Use the NI-IMAQ I/O watchdog methods to set up a timer for the NI-IMAQ I/O device.

[imaqIOWatchdogConfigure](#)

[imaqIOWatchdogArm](#)

[imaqIOWatchdogClear](#)

[imaqIOWatchdogWhack](#)

# **imaqIOWatchdogConfigure Method**

## **Syntax**

**imaqIOWatchdogConfigure** id, timeout, expirationAction

## **Purpose**

Configures a watchdog timer on the NI-IMAQ I/O device.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to configure the watchdog on.

**timeout** As [Long](#)

The time, in milliseconds, that can occur without a reset before the watchdog occurs.

**expirationAction** As [IMAQIOExpirationAction](#)

The action to take when the watchdog occurs.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOWatchdogArm](#)

[imaqIOWatchdogClear](#)

[imaqIOWatchdogWhack](#)

# imaqIOWatchdogArm Method

## Syntax

`imaqIOWatchdogArm id`

## **Purpose**

Arms the watchdog timer on the NI-IMAQ I/O device.



## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to arm the watchdog on.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgqIOGetErrorTextVB](#).

---

## See Also

[imaqIOWatchdogClear](#)

[imaqIOWatchdogConfigure](#)

[imaqIOWatchdogWhack](#)

# **imaqIOWatchdogClear Method**

## **Syntax**

**imaqIOWatchdogClear id**

## **Purpose**

Clears the watchdog timer on the NI-IMAQ I/O device.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to clear the watchdog on.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOWatchdogArm](#)

[imaqIOWatchdogConfigure](#)

[imaqIOWatchdogWhack](#)



# **imaqIOWatchdogWhack Method**

## **Syntax**

**imaqIOWatchdogWhack id**

## **Purpose**

Resets the watchdog timer on the NI-IMAQ I/O device.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to reset the watchdog timer on.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOWatchdogArm](#)

[imaqIOWatchdogClear](#)

[imaqIOWatchdogConfigure](#)

## NI-IMAQ I/O Shutdown States Methods

Use the NI-IMAQ I/O shutdown states methods to enable/disable and get/set shutdown states on an NI-IMAQ I/O device.

[imaqIOSetShutdownState](#)

[imaqIOGetShutdownState](#)

[imaqIOEnableShutdown](#)

[imaqIOQueryShutdown](#)

## **imaqIOSetShutdownState Method**

### **Syntax**

**imaqIOSetShutdownState** id, signalType, signalNumber, value

## **Purpose**

Sets the shutdown state of one of the output signals on the NI-IMAQ I/O device.



## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to set the shutdown state on.

**signalType** As [IMAQIOSignalType](#)

The type of signal to set the shutdown state on.

**signalNumber** As [Long](#)

The line number to set the shutdown state on.

**value** As [IMAQIOSignalState](#)

The shutdown state to set the device to.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOEnableShutdown](#)

[imaqIOQueryShutdown](#)

[imaqIOGetShutdownState](#)

## **imaqIOGetShutdownState Method**

### **Syntax**

**imaqIOGetShutdownState** id, signalType, signalNumber, value

## **Purpose**

Gets the shutdown state of one of the output signals on the NI-IMAQ I/O device.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to get the shutdown state for.

**signalType** As [IMAQIOSignalType](#)

The type of signal to get the shutdown state for.

**signalNumber** As [Long](#)

The line number of the signal to get the shutdown state for.

**value** As [IMAQIOSignalState](#)

(Output Parameter) The state of the NI-IMAQ I/O device.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOEnableShutdown](#)

[imaqIOQueryShutdown](#)

[imaqIOSetShutdownState](#)



# **imaqIOEnableShutdown Method**

## **Syntax**

**imaqIOEnableShutdown** id, enableShutdown

## Purpose

Enables or disables the shutdown functionality of the NI-IMAQ I/O device.

When the shutdown functionality is enabled, the device reaches a shutdown state under two conditions:

- If ISO\_IN\_11 is Low
- If the Watchdog Timer expires and the expiration action is IMAQIO\_EXPIRATION\_ACTION\_SHUTDOWN

When the device is in a shutdown state, it asserts the outputs with the values that have been specified through [imaqIOSetShutdownState](#). The default value for the outputs is Disabled for TTL Outputs, and Low for ISO Outputs. Shutdown is a fatal condition. Clear shutdown by powering off and then powering on your system.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to enable or disable shutdown on.

**enableShutdown** As [Long](#)

1 enables shutdown, and 0 disables shutdown.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

---

## See Also

[imaqIOGetShutdownState](#)

[imaqIOQueryShutdown](#)

[imaqIOSetShutdownState](#)

# imaqIOQueryShutdown Method

## Syntax

**imaqIOQueryShutdown** id, enableShutdown

## **Purpose**

Queries the shutdown functionality of the NI-IMAQ I/O device to determine if it is enabled or disabled.

## Parameters

**id** As [IMAQIO\\_SESSION](#)

The ID of the session to query shutdown on.

**shutdownEnabled** As [Long](#)

(Output Parameter) The return data that indicates if the shutdown functionality of the NI-IMAQ I/O device is enabled. 1 indicates that the shutdown functionality is enabled, and 0 indicates that the shutdown functionality is disabled.



## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imgIIOGetErrorTextVB](#).

---

## See Also

[imaqIOEnableShutdown](#)

[imaqIOGetShutdownState](#)

[imaqIOSetShutdownState](#)

## **NI-IMAQ I/O Error Handling Method**

Use the NI-IMAQ I/O error handling method to retrieve error information.

[imaqIOGetErrorTextVB](#)

# **imaqIOGetErrorTextVB Method**

## **Syntax**

**imaqIOGetErrorTextVB** errorCode, errorText

## **Purpose**

Gets the error text associated with an error code.

## Parameters

**errorCode** As [IMAQIO\\_ERR](#)

The error code to get text for.

**errorText** As [String](#)

(Output Parameter) Returns the error text.

## Return Value

This function returns 0 on success. On failure, this function returns an error code. For information about the error code, call [imaqIOGetErrorTextVB](#).

## NI-IMAQ I/O Constants and Enumerations

The following NI-IMAQ I/O constants and enumerations provide the valid values for parameters in the NI-IMAQ I/O methods:

[IMAQIO\\_ERR](#)

[IMAQIO\\_PULSE\\_ID](#)

[IMAQIO\\_SESSION](#)

[IMAQIOAttribute](#)

[IMAQIOExpirationAction](#)

[IMAQIOPolarity](#)

[IMAQIOPulseMode](#)

[IMAQIOSignalState](#)

[IMAQIOSignalType](#)

[IMAQIOTimebase](#)



## IMAQIOAttribute Constants

IMAQIOAttribute constants are the constants for the **attribute** parameter of the [imaqIOGetAttribute](#) and [imaqIOSetAttribute](#) methods in the NI-IMAQ I/O library.



**Note** Currently, these values are not settable.

- `IMAQIO_ATTRIBUTE_PRODUCT_SELECT_CURRENT`—Gets the value of the current product selection line. ISO Inputs 4 thru 0 are the product selection lines.
- `IMAQIO_ATTRIBUTE_PRODUCT_SELECT_LATCHED`—Gets the value of the currently latched product selection line. ISO Inputs 4 thru 0 are the product selection lines. ISO Input 5 latches in the data on ISO Inputs 4 thru 0.
- `IMAQIO_ATTRIBUTE_QUADRATURE_ENCODER`—Gets the value of the quadrature encoder. The quadrature encoder uses ISO Input 6 for its Phase A input and ISO Input 7 for its Phase B input.
- `IMAQIO_ATTRIBUTE_WATCHDOG_STATUS`—Gets the value of the watchdog status. 1 indicates that no errors occurred, and 0 indicates that a timeout occurred.
- `IMAQIO_ATTRIBUTE_ISOPOWER_PRESENT`—Gets the ISO power status. A value of 1 indicates that ISO power is present. A value of 0 indicates that ISO power is absent.

## IMAQIOChangeDetectMode

IMAQIOChangeDetectMode constants are the constants for the [imaqIOChangeDetectConfigure](#) method in the NI-IMAQ I/O library.

- IMAQ\_IO\_CHANGE\_DETECT\_ANY\_EDGE—Detects a change on any edge.
- IMAQ\_IO\_CHANGE\_DETECT\_RISING\_EDGE—Detects a change on a rising edge.
- IMAQ\_IO\_CHANGE\_DETECT\_FALLING\_EDGE—Detects a change on a falling edge.

## IMAQIOExpirationAction Constants

IMAQIOExpirationAction constants are the constants for the **expirationAction** parameter of the [imaqIOWatchdogConfigure](#) method in the NI-IMAQ I/O library. Use these constants to specify what action to take when the watchdog timer expires.

- IMAQIO\_EXPIRATION\_ACTION\_INDICATOR\_ONLY—Displays a message when the watchdog occurs.
- IMAQIO\_EXPIRATION\_ACTION\_ASSERT\_TTL\_OUT\_0—Drives the TTL Out 0 line when the watchdog occurs.
- IMAQIO\_EXPIRATION\_ACTION\_SHUTDOWN—Shuts down the NI-IMAQ I/O device when the watchdog occurs. Refer to [imaqIOEnableShutdown](#) for information about enabling shutdown.

## IMAQIOPolarity Constants

IMAQIOPolarity constants are the constants for the [imaqIOPulseCreate](#) method in the NI-IMAQ I/O library.

## **outputSignalPolarity**

For the **outputSignalPolarity** parameter, these constants behave as follows:

- **IMAQIO\_POLARITY\_HIGH\_TRUE**—Drives the line high during the pulse assertion, which is dictated by the **width** parameter.
- **IMAQIO\_POLARITY\_LOW\_TRUE**—Drives the line low during the pulse assertion, which is dictated by the **width** parameter.

## **triggerSignalPolarity**

For the **triggerSignalPolarity** parameter, these constants behave as follows:

- `IMAQIO_POLARITY_HIGH_TRUE`—Triggers the pulse when the signal is high.
- `IMAQIO_POLARITY_LOW_TRUE`—Triggers the pulse when the signal is low.

## IMAQIOPulseMode Constants

IMAQIOPulseMode constants are the constants for the [imaqIOPulseCreate](#) method in the NI-IMAQ I/O library.

- PULSE\_MODE\_TRAIN—Pulse is generated continuously after the trigger is asserted. Choose this option to generate a continuous pulse train that is inactive for the time specified in the delay parameter, and active for the time specified in the width parameter. When the pulse train is started, it continues periodically until you call [imaqIOPulseStop](#), [imaqIOPulseDispose](#), or [imaqIOPClose](#). This mode is valid only if you set **triggerSignalType** to IMAQIO\_SIGNAL\_STATUS, and [triggerSignalNumber](#) is set to [IMAQIO\\_STATUS\\_SIGNAL\\_NONE](#).
- PULSE\_MODE\_SINGLE—This option is not supported.
- PULSE\_MODE\_SINGLE\_REARM—Pulse occurs one time on each trigger occurrence. Choose this option to generate a rearmed single shot pulse. On every occurrence of the trigger, the output line stays inactive for the time specified in the delay parameter, and becomes active for the time specified in the width parameter. When the pulse is started, output toggles for each occurrence of the trigger until you call [imaqIOPulseStop](#), [imaqIOPulseDispose](#), or [imaqIOPClose](#). This mode works only when the application is configured to trigger on the TTL In or ISO In lines.

## IMAQIOSignalState Constants

IMAQIOSignalState constants are the constants for the **value** parameter of the following methods:

- [imaqIODrive](#)
- [imaqIORead](#)
- [imaqIOGetShutdownState](#)
- [imaqIOSetShutdownState](#)

- 
- IMAQIO\_SIGNAL\_STATE\_HIGH—Drives the line high when the signal is True.
  - IMAQIO\_SIGNAL\_STATE\_LOW—Drives the line low when the signal is True.
  - IMAQIO\_SIGNAL\_STATE\_HI\_Z—Disables output on the line. This option is valid only for TTL output signals.



# IMAQIOSignalType Enumeration

IMAQIOSignalType enumerations are the constants for the **signalType** parameter on the methods listed in the following table:

| Attribute             | Description  | Applicable Methods   | Notes  |
|-----------------------|--|--|--|
| IMAQIO_SIGNAL_ISO_IN  | Pulse trigger inputs, a shutdown input, and a quadrature encoder input | <a href="#">imaqIORead</a><br><a href="#">imaqIOPulseCreate (triggerSignalType)</a><br><a href="#">imaqIOChangeDetectConfigure</a><br><a href="#">imaqIOChangeDetectQuery</a><br><a href="#">imaqIOChangeDetectReset</a> | For the <b>triggerSignalType</b> parameter of the <code>imaqIOPulseCreate</code> method, this input is used as a pulse trigger for signal lines 5, 8, or |
| IMAQIO_SIGNAL_ISO_OUT | General-purpose outputs  | <a href="#">imaqIODrive</a><br><a href="#">imaqIORead</a><br><a href="#">imaqIOPulseCreate (outputTriggerType)</a><br><a href="#">imaqIOSetShutdownState</a><br><a href="#">imaqIOGetShutdownState</a>                   |  |
| IMAQIO_SIGNAL_STATUS  | Internal timing signals, including the immediate trigger mode          | <a href="#">imaqIOPulseCreate (triggerSignalType)</a>  |  |
| IMAQIO_SIGNAL_TTL_IN  | General-purpose inputs   | <a href="#">imaqIORead</a><br><a href="#">imaqIOPulseCreate (triggerSignalType)</a><br><a href="#">imaqIOChangeDetectConfigure</a><br><a href="#">imaqIOChangeDetectQuery</a><br><a href="#">imaqIOChangeDetectReset</a> | For the <b>triggerSignalType</b> parameter of the <code>IMAQIOPulseCreate</code> method, this input is only for signal lines 1.                          |
| IMAQIO_SIGNAL_TTL_OUT | General-purpose outputs  | <a href="#">imaqIODrive</a><br><a href="#">imaqIORead</a><br><a href="#">imaqIOPulseCreate (outputTriggerType)</a><br><a href="#">imaqIOSetShutdownState</a><br><a href="#">imaqIOGetShutdownState</a>                   | For the <b>outputTriggerSignalType</b> parameter of the <code>imaqIOPulseCreate</code> method, this output signal is for signal lines 1, 2, 3, 4, 8, and |

# IMAQIOStatusSignal Enumeration

## Purpose

This enumeration provides options you can use for the signal number when you set signal type to status.

- `IMAQIO_STATUS_SIGNAL_NONE`—Sets the signal number to none, which results in an immediate trigger.

## IMAQIOTimebase Enumeration

IMAQIOTimebase enumerations are the constants for the **delayTimebase** and **widthTimebase** parameters on the [imaqIOPulseCreate](#) method.

You can use the following constants with this data type:

- IMAQIO\_TIMEBASE\_ENCODER\_TICKS—Defines the delay or width in encoder counts.
- IMAQIO\_TIMEBASE\_MICROSECONDS—Defines the delay or width in microseconds.



**Note** The IMAQIO\_TIMEBASE\_ENCODER\_TICKS enumeration is not applicable to the **widthTimebase** parameter.

## IMAQIO\_ERR

IMAQIO\_ERR is the typedef for the **errorCode** parameter of the [imaqIOGetErrorTextVB](#) method. The **errorCode** parameter is returned by each method in the NI-IMAQ I/O library.

## IMAQIO\_PULSE\_ID

IMAQIO\_PULSE\_ID is the structure that defines the functionality of the **pulseID** parameter of the following methods:

- [imaqIOPulseCreate](#)
- [imaqIOPulseDispose](#)
- [imaqIOPulseStart](#)
- [imaqIOPulseStop](#)

# IMAQIO\_SESSION

IMAQIO\_SESSION is the typedef structure that defines the functionality of the **id** parameter of the following methods:

- [imaqIOClose](#)
- [imaqIODrive](#)
- [imaqIOEnableShutdown](#)
- [imaqIOGetAttribute](#)
- [imaqIOOpen](#)
- [imaqIORead](#)
- [imaqIOGetShutdownState](#)
- [imaqIOPulseCreate](#)
- [imaqIOPulseDispose](#)
- [imaqIOPulseStart](#)
- [imaqIOPulseStop](#)
- [imaqIOQueryShutdown](#)
- [imaqIOSetAttribute](#)
- [imaqIOSetShutdownState](#)
- [imaqIOWatchdogArm](#)
- [imaqIOWatchdogClear](#)
- [imaqIOWatchdogConfigure](#)
- [imaqIOWatchdogWhack](#)

# Data Types for NI-imaqIO

| <b>Visual Basic</b> | <b>C/C++</b>  | <b>Description</b>                                 |
|---------------------|---------------|--|
| Long                | long          | 32-bit signed integer.                             |
| Picture             | LPPICTUREDISP | An OLE Automation Interface to a picture or image. |
| String              | BSTR          | A string.  |
| Void                | void          | No data.   |

# Error Codes

| Error Code | Status Name   | Description   |
|------------|---|---|
| -301521    | IMAQIO_ERR_INVALID_PULSE_MODE                       | The pulse mode is invalid.  |
| -301520    | IMAQIO_ERR_ATTRIBUTE_NOT_SETTABLE                   | The attribute is not settable.  |
| -301519    | IMAQIO_ERR_TIMEOUT_OUT_OF_RANGE                     | The timeout value is out of range. This watchdog timer supports only 16-bit values. |
| -301518    | IMAQIO_ERR_INVALID_WATCHDOG_TIMER_EXPIRATION_ACTION | The watchdog timer expiration action is invalid.                                    |
| -301517    | IMAQIO_ERR_INVALID_ATTRIBUTE_TO_READ                | The attribute read is invalid.  |
| -301516    | IMAQIO_ERR_INVALID_PULSE_ID                         | The pulse identifier is invalid.  |
| -301515    | IMAQIO_ERR_INVALID_TIMEBASE                         | The timebase is invalid.  |
| -301514    | IMAQIO_ERR_INVALID_POLARITY                         | The polarity is invalid.  |
| -301513    | IMAQIO_ERR_PULSE_OUTPUT_ALREADY_IN_USE              | The pulse output is already in use.   |
| -301512    | IMAQIO_ERR_INVALID_PULSE_TRIGGER_SIGNAL             | The pulse trigger signal is invalid.  |
| -301511    | IMAQIO_ERR_INVALID_PULSE_OUTPUT_SIGNAL              | The pulse output signal is invalid.   |
| -301510    | IMAQIO_ERR_INVALID_SIGNAL_STATE_FOR_THIS_CHANNEL    | The signal state is invalid for this channel.                                       |



|         |  |  |
|---------|--|--|
| -301509 | IMAQIO_ERR_INVALID_SIGNAL_FOR_THIS_OPERATION | The signal is invalid for this operation.    |
| -301508 | IMAQIO_ERR_PULSE_ALREADY_RUNNING             | The pulse is already running.                |
| -301507 | IMAQIO_ERR_INVALID_SESSION                   | The session is invalid.                      |
| -301506 | IMAQIO_ERR_INTERNAL_ERROR                    | An internal error has occurred.              |
| -301505 | IMAQIO_ERR_ERROR_NOT_FOUND                   | The specified error is not found.            |
| -301504 | IMAQIO_ERR_ERROR_TEXT_TOO_SHORT              | Text buffer is too small for the error text. |
| -301503 | IMAQIO_ERR_OUT_OF_MEMORY                     | Out of memory.                               |
| -301502 | IMAQIO_ERR_INVALID_POINTER                   | The pointer is invalid.                      |
| -301501 | IMAQIO_ERR_DEVICE_IN_USE                     | The device is in use.                        |
| -301500 | IMAQIO_ERR_DEVICE_NOT_FOUND                  | The device was not found.                    |

# Important Information

[Warranty](#)

[Copyright](#)

[Trademarks](#)

[Patents](#)

[Warning Regarding Use of NI Products](#)

## Warranty

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

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

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

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

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

## Copyright

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

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

## Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about [National Instruments trademarks](#).

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

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

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

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

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

## Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the patents.txt file on your CD, or [ni.com/patents](http://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 National Instruments Web site at [ni.com](http://ni.com) for technical support and professional services:

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

For information about other [technical support options](#) in your area, visit [ni.com/services](http://ni.com/services) or [contact](#) your local office at [ni.com/contact](http://ni.com/contact).

- [Training and Certification](#)—Visit [ni.com/training](http://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](http://ni.com/alliance).

If you searched [ni.com](http://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](http://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 0 662 45 79 90 0 |
| Belgium                   | 32 0 2 757 00 20    |
| Brazil                    | 55 11 3262 3599     |
| Canada                    | 800 433 3488        |
| China                     | 86 21 6555 7838     |
| Czech Republic            | 420 224 235 774     |
| Denmark                   | 45 45 76 26 00      |
| Finland                   | 358 0 9 725 725 11  |
| France                    | 33 0 1 48 14 24 24  |
| Germany                   | 49 0 89 741 31 30   |
| India                     | 91 80 51190000      |
| Israel                    | 972 0 3 6393737     |
| Italy                     | 39 02 413091        |
| Japan                     | 81 3 5472 2970      |
| Korea                     | 82 02 3451 3400     |
| Lebanon                   | 961 0 1 33 28 28    |
| Malaysia                  | 1800 887710         |
| Mexico                    | 01 800 010 0793     |
| Netherlands               | 31 0 348 433 466    |
| New Zealand               | 0800 553 322        |
| Norway                    | 47 0 66 90 76 60    |
| Poland                    | 48 22 3390150       |
| Portugal                  | 351 210 311 210     |
| Russia                    | 7 095 783 68 51     |
| Singapore                 | 1800 226 5886       |
| Slovenia                  | 386 3 425 4200      |
| South Africa              | 27 0 11 805 8197    |
| Spain                     | 34 91 640 0085      |
| Sweden                    | 46 0 8 587 895 00   |
| Switzerland               | 41 56 200 51 51     |
| Taiwan                    | 886 02 2377 2222    |
| Thailand                  | 662 992 7519        |
| United Kingdom            | 44 0 1635 523545    |
| United States (Corporate) | 512 683 0100        |