

NI-MotionTM VI Help

July 2007, 370537H-01

This help file provides a comprehensive reference of all NI-Motion Virtual Instruments (VIs). This document is intended for use by developers of NI-Motion systems with a working knowledge of LabVIEW.

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

Using Help

Related Documentation

Glossary

Important Information

Technical Support and Professional Services

To comment on National Instruments documentation, refer to the <u>National</u> Instruments Web site.

© 1999–2007 National Instruments Corporation. All rights reserved.

Using Help

Conventions

Navigating Help

Searching Help

Printing Help File Topics

Conventions

This help file uses the following formatting and typographical conventions:

<>	Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, AO <03>.
[]	Square brackets enclose optional items—for example, [response].
»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence File»Page Setup»Options directs you to pull down the File menu, select the Page Setup item, and select Options from the last dialog box.
	This icon denotes a tip, which alerts you to advisory information.
	This icon denotes a note, which alerts you to important information.
	This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
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.
dark red	Text in this color denotes a caution.
green	Underlined text in this color denotes a link to a help topic, help file, or Web address.
italic	Italic text denotes variables, emphasis, cross-references, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.
monospace	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

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.

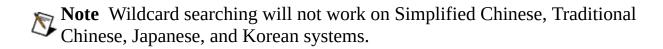
Searching Help (Windows Only)

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.



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.

Axes

An axis consists of a trajectory generator, PID or stepper control block, and some sort of output resource, either a <u>digital-to-analog converter (DAC)</u> output or a <u>stepper pulse generator output</u>. Servo axes must also have some sort of feedback resource, either an <u>encoder</u> or <u>ADC channel</u>. Closed-loop stepper axes also require a feedback resource, and can use either encoder or ADC inputs. Open-loop stepper axes do not require feedback for correct operation. The following table lists the resource IDs for axes:

Resource Name Resource ID

Axis Control	0x00(0)
Axis 1	0x01(1)
Axis 2	0x02 (2)
Axis 3	0x03(3)
Axis 4	0x04(4)
Axis 5	0x05(5)
Axis 6	0x06(6)
Axis 7	0x07(7)
Axis 8	0x08(8)
Axis 9	0x09 (9)
Axis 10	0x0A (10)
Axis 11	0x0B (11)
Axis 12	0x0C (12)
Axis 13	0x0D (13)
Axis 14	0x0E (14)
Axis 15	0x0F (15)

VIs that can operate on multiple axes simultaneously (for example, <u>Read Blend</u> <u>Status</u> and <u>Start Motion</u>) can take the axis control (0) as their resource parameter.

Using Axes Nine Through Fifteen

Complete the following steps to use axes nine through fifteen:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected axis number (in decimal) for axes nine through fifteen.

Vector Spaces

Vector spaces are logical, multidimensional groups of axes. They can be either one-dimensional, two-dimensional with x and y axes, or three-dimensional with x, y, and z axes. The total number of vector spaces NI-Motion supports in your motion control system can be calculated using the following formula:

total vector spaces = [number of activated axes/2].

Vector spaces facilitate 2D and 3D interpolated moves: linear, circular, helical, and spherical. You can send a vector space to many NI-Motion VIs to define vector position, vector velocity, vector acceleration, and so on.

Vector spaces are started, stopped, and controlled as if they were a single axis, greatly simplifying the control of coordinated vector axes. All axes in a vector space start and stop at the same time, completing the vector motion profiles programmed. The following table lists the resource IDs for vector space control.

Resource Name	Resource ID
Vector Space Control	0x10 (16)
Vector Space 1	0x11 (17)
Vector Space 2	0x12 (18)
Vector Space 3	0x13 (19)
Vector Space 4	0x14 (20)
Vector Space 5	0x15 (21)
Vector Space 6	0x16 (22)
Vector Space 7	0x17 (23)
Vector Space 8	0x18 (24)

VIs that can operate on multiple vector spaces simultaneously (for example, Start Motion) can take the vector space control (0x10) as their resource parameter.

Vector spaces are configured by mapping axes to the vector space with the Configure Vector Space VI. Vector spaces are logical, not physical, and do not

require motion resources other than those used by the axes themselves.

Using Vector Spaces Five Through Eight

Complete the following steps to use vector spaces five through eight:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected vector space number (in decimal) for vector spaces five through eight.

Motion Resources

There are four types of motion resources on the NI-Motion controller: <u>encoders</u>, <u>ADC channels</u>, <u>DAC outputs</u>, and <u>stepper outputs</u>. In general, VIs relating to motion resources (for example, <u>Read DAC</u> and <u>Read Steps Generated</u>) can be sent to the resource or the axis the resource is mapped to.

Note After they are mapped to an axis, all features and functions of a motion resource are available as part of the axis. It is not necessary to remember or use the resource number directly when accessing these features as part of the axis. Resources are referenced by axis number after they are assigned to that axis.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using these resources with a 73xx controller:

- 7330 controllers do not support DAC outputs or secondary ADCs.
- 7340 controllers do not support secondary ADCs.
- 7350 controllers support a secondary feedback device on each axis.
- 7390 controllers do not support ADC channels, DAC outputs, secondary ADCs, or secondary encoders.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using these resources with the NI SoftMotion Controller:

• The NI SoftMotion Controller supports a secondary feedback device on each axis.

• Secondary Encoders

Use secondary encoders in a dual-loop feedback system to provide a more accurate velocity feedback estimation.

Resource Name	Resource ID
Secondary Encoder 1	0X71 (113)
Secondary Encoder 2	0X72 (114)
	•
	•
	•

Secondary Encoder 15 0X7F (127)

• Secondary ADCs

Use secondary ADCs in a dual-loop feedback system to provide a more accurate velocity feedback estimation.

```
Resource Name Resource ID

Secondary ADC 1 0X91 (145)

Secondary ADC 2 0X92 (146)

.
.
Secondary ADC 15 0X9F (159)
```

Encoders

Encoder resources are primarily used for position feedback on servo and closed-loop stepper axes. When encoder resources are not mapped to an axis for use as axis feedback, you can use them for any number of other functions including position or velocity monitoring, as digital potentiometer encoder inputs, or as master encoders for master/slave and gearing applications. The following table lists the resource IDs for encoders:

Resource Name	Resource ID
Encoder Control	0x20 (32)
Encoder 1	0x21 (33)
Encoder 2	0x22 (34)
Encoder 3	0x23 (35)
Encoder 4	0x24 (36)
Encoder 5	0x25 (37)
Encoder 6	0x26 (38)
Encoder 7	0x27 (39)
Encoder 8	0x28 (40)
Encoder 9	0x29 (41)
Encoder 10	0x2A (42)
Encoder 11	0x2B (43)
Encoder 12	0x2C (44)
Encoder 13	0x2D (45)
Encoder 14	0x2E (46)
Encoder 15	0x2F (47)

VIs that can operate on multiple encoders simultaneously (for example, <u>Read High-Speed Capture Status</u>) can take the encoder control (0x20) as their resource parameter.

All encoders feature high-speed capture inputs and breakpoint outputs. These features are implemented in the encoder processor FPGA and are fully functional when an encoder is used as an independent resource or as feedback for an axis.

Using Encoders Nine Through Fifteen

Complete the following steps to use encoders nine through fifteen:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected encoder number (in decimal) for encoders nine through fifteen.

ADC Channels

You can use ADC channels as analog feedback for axes or as general-purpose analog inputs to measure sensors or potentiometers.

All ADC channels are multiplexed and automatically scanned to keep the converted ADC register values current. The following table lists the resource IDs for ADCs:

Resource Name	Resource ID
ADC 1	0x51 (81)
ADC 2	0x52 (82)
ADC 3	0x53 (83)
ADC 4	0x54 (84)
ADC 5	0x55 (85)
ADC 6	0x56 (86)
ADC 7	0x57 (87)
ADC 8	0x58 (88)
ADC 9	0x59 (89)
ADC 10	0x5A(90)
ADC 11	0x5B (91)
ADC 12	0x5C (92)
ADC 13	0x5D (93)
ADC 14	0x5E (94)
ADC 15	0x5F (95)

ADC channels do not typically provide the same level of feedback performance as encoders, but have the advantage of providing absolute rather than incremental feedback.

On 7340/7330 controllers, ADCs 5 through 8 are hard-wired to specific sources, and cannot be used as general-purpose resources or as feedback devices. Refer to the Read ADCs VI for more information.

Using ADCs Nine Through Fifteen

Complete the following steps to use ADCs nine through fifteen:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected ADC number (in decimal) for ADCs nine through fifteen.

DAC Outputs

DAC resources are typically mapped to servo axes and generate the analog control outputs from the PID loops. DAC resources that are not used by axes are available for non-axis specific applications. You can directly control an unmapped DAC as a general-purpose analog output.

The DAC outputs offer 16-bit resolution and the industry-standard ± 10 V range. Refer to the specifications of the motion controller documentation for complete DAC output specifications. The following table lists the resource IDs for DACs:

Resource Name	Resource ID
DAC 1	0x31 (49)
DAC 2	0x32 (50)
DAC 3	0x33 (51)
DAC 4	0x34 (52)
DAC 5	0x35 (53)
DAC 6	0x36 (54)
DAC 7	0x37 (55)
DAC 8	0x38 (56)
DAC 9	0x39 (57)
DAC 10	0x3A(58)
DAC 11	0x3B (59)
DAC 12	0x3C (60)
DAC 13	0x3D (61)
DAC 14	0x3E (62)
DAC 15	0x3F (63)

Using DACs Nine Through Fifteen

Complete the following steps to use DACs nine through fifteen:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected DAC number (in decimal) for DACs nine through fifteen.

Stepper Outputs

Stepper output resources generate the step pulses required for stepper axis control. They operate like the DAC output in a servo axis.

NI-Motion supports the two industry-standard stepper output configurations: Step and Direction, or CW/CCW pulses. Refer to <u>Configure Stepper Output</u> for more information about these output configurations. The following table lists the resource IDs for stepper outputs:

Resource Name Resource ID

Stepper Output 1 0x41 (65)

Stepper Output 2 0x42 (66)

Stepper Output 3 0x43 (67)

Stepper Output 4 0x44 (68)

Stepper Output 5 0x45 (69)

Stepper Output 6 0x46 (70)

Stepper Output 7 0x47(71)

Stepper Output 8 0x48 (72)

Stepper Output 9 0x49(73)

Stepper Output 10 0x4A (74)

Stepper Output 11 0x4B (75)

Stepper Output 12 0x4C (76)

Stepper Output 13 0x4D (77)

Stepper Output 14 0x4E (78)

Stepper Output 15 0x4F (79)

Using Stepper Outputs Nine Through Fifteen

Complete the following steps to use stepper outputs nine through fifteen:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected stepper output number (in decimal) for stepper outputs nine through fifteen.

General-Purpose I/O Ports

You can use the general-purpose I/O port structure for programmable logic controller (PLC) functions or for simple point I/O. Refer to <u>Analog & Digital I/O VIs</u> for information about how to configure the direction and polarity of the I/O ports and bits, set and reset individual bits, and read the logical port status.

These I/O ports are also hardware resources, but because they are never mapped to axes, they are not considered motion resources. Refer to 73xx Controller General-Purpose I/O Port IDs and NI SoftMotion Controller General-Purpose I/O Port IDs for I/O port resource numbers.

Notes



- 1. The RTSI port is not supported by the NI SoftMotion Controller.
- 2. The resource ID for the RTSI port is 0x09 (9).

Using Unlisted General-Purpose I/O Ports

Complete the following steps to use unlisted general-purpose I/O ports:

- 1. Open the desired VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the appropriate terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the constant and select **Other**.
- 5. Type the selected port number, in decimal.

Available I/O Resources

The following table lists the I/O resources by controller type.

Device	Digital I/O Lines
NI 7330	four 8-bit ports, RTSI I/O port
NI 7340	four 8-bit ports, RTSI I/O port
NI 7350	eight 8-bit ports, RTSI I/O port
NI PCI-7390	four 2-bit input ports, four 2-bit output ports, RTSI I/O port
NI SoftMotion Controller for	
Xenus	seven digital input lines, three digital output
(Copley Controls CANopen drive)	lines per axis
NI SoftMotion Controller for	
Accelnet	seven digital input lines, two digital input lines
(Copley Controls CANopen	per axis
drive)	

73xx Controller General-Purpose I/O **Port IDs**

7330, 7340, and 7350 Controller I/O Port Resource **IDs**

The following table lists the resource IDs for I/O ports on 7330, 7340, and 7350 controllers.



Note The I/O line direction is user configurable using the Set I/O Port Direction VI, so these controllers do not have separate port IDs for input and output ports.

Resource Name Resource ID

I/O Port 1	0x01 (1)
I/O Port 2	0x02(2)
I/O Port 3	0x03(3)
I/O Port 4	0x04(4)
I/O Port 5	0x05(5)
I/O Port 6	0x06 (6)
I/O Port 7	0x07(7)
I/O Port 8	0x08(8)
RTSI Port	0x09(9)

PCI-7390 Controller I/O Port Resource IDs

The following table lists the resource IDs for I/O ports on the PCI-7390 controller.

Digital Input Ports		Digital Output Ports	
Resource Name	e Resource ID	Resource Name	Resource ID
Input Port 1	0x01(1)	Output Port 1	0x81 (129)
Input Port 2	0x02(2)	Output Port 2	0x82 (130)
Input Port 3	0x03(3)	Output Port 3	0x83 (131)
Input Port 4	0x04(4)	Output Port 4	0x84 (132)

NI SoftMotion Controller General-Purpose I/O Port IDs

The following table lists the resource IDs for I/O ports on the NI SoftMotion Controller:

Digital Input Ports		Digital Output Ports	
Resource Name	Resource ID	Resource Name	Resource ID
Input Port 1	0x01(1)	Output Port 1	0x81 (129)
Input Port 2	0x02(2)	Output Port 2	0x82 (130)
Input Port 3	0x03(3)	Output Port 3	0x83 (131)
Input Port 4	0x04(4)	Output Port 4	0x84 (132)
Input Port 5	0x05(5)	Output Port 5	0x85 (133)
Input Port 6	0x06 (6)	Output Port 6	0x86 (134)
Input Port 7	0x07(7)	Output Port 7	0x87 (135)
Input Port 8	0x08(8)	Output Port 8	0x88 (136)
Input Port 9	0x09(9)	Output Port 9	0x89 (137)
Input Port 10	0x0A(10)	Output Port 10	0x8A (138)
Input Port 11	0x0B (11)	Output Port 11	0x8B (139)
Input Port 12	0x0C (12)	Output Port 12	0x8C (140)
Input Port 13	0x0D (13)	Output Port 13	0x8D (141)
Input Port 14	0x0E (14)	Output Port 14	0x8E (142)
Input Port 15	0x0F (15)	Output Port 15	0x8F (143)

Trajectory Parameters

All trajectory parameters for servo and closed-loop stepper axes are expressed in terms of quadrature encoder counts. Parameters for open-loop stepper axes are expressed in steps. For servo axes, the encoder resolution in counts per revolution determines the ultimate positional resolution of the axis.

For stepper axes, the number of steps per revolution depends upon the type of stepper driver and motor being used. For example, a stepper motor with 1.8°/step (200 steps/revolution) used in conjunction with a 10x microstep driver would have an effective resolution of 2,000 steps per revolution. Resolution on closed-loop stepper axes is limited to the steps per revolution or encoder counts per revolution, whichever is coarser.

There are two other factors that affect the way trajectory parameters are loaded to the NI motion controller versus how they are used by the trajectory generators: floating-point versus fixed-point parameter representation, and time base.

You can load some trajectory parameters as either floating-point or fixed-point values. The internal representation on the NI motion controller is always fixed-point, however. This fact is important when working with onboard variables, input, and return vectors. It also has a small effect on parameter range and resolution, as shown in the example in the <u>Velocity in RPM</u> topic.

The second factor is the time base. Velocity and acceleration values are loaded in counts/s, RPM, RPS/sec, steps/s, and so on—all functions of seconds or minutes. But the trajectory generators update target position at the Trajectory Update Rate, which is programmable with the Enable Axes VI. This means that the range for these parameters depends on the update rate selected, as shown in the example in the Velocity in RPM topic.

RPM values stored in onboard variables are in double-precision IEEE format (f64). Refer to <u>Onboard Variables and Input and Return Vectors</u>, for information about the number of variables required to hold an RPM value.

NI 73xx Velocity in RPM

NI 73xx Velocity in Counts/s or Steps/s

NI 73xx Acceleration in Counts/s²

NI 73xx Acceleration in RPS/s

NI 73xx Velocity Override in Percent

NI 73xx Arc Angles in Degrees

ValueMotion to NI-Motion Conversion

To aid in converting a ValueMotion application to NI-Motion, the following table lists each VI in the ValueMotion API and gives the nearest NI-Motion VI. Some ValueMotion VIs are related to two NI-Motion VIs. In these cases, the first VI is the closest match to the ValueMotion VI, but the second NI-Motion VI, in combination with the first, is required to completely mimic the functionality of the original ValueMotion VI.

Refer to the individual VIs in both the ValueMotion and NI-Motion VI online help for detailed information about the functional and syntactic differences.

NI-Motion Equivalent
Acquire Trajectory Data
Flush Return Data Buffer
Read Return Data Buffer
Configure Gear Master
Enable Gearing
Enable Limits
Enable Home Inputs
Enable Breakpoint Output
Configure Breakpoint
Wait on Condition
Find Reference Home
Find Reference Index
Get Motion Board Information
Get Motion Board Information
Initialize Controller
Stop Motion
Stop Motion
Load Acceleration/Deceleration
Not supported
Load Breakpoint Position

Load Breakpoint Repeat Period Load Breakpoint Modulus

VI Execution Times

For selected VI execution times, select your controller from the following list:

- NI 7330
- <u>NI 7334</u>
- <u>NI 7340</u>
- <u>NI 7342</u>
- NI 7344
- <u>NI 7350</u>
- <u>NI 7390</u>

Communication between the Host Computer and the NI Motion Controller

The host computer communicates with the NI motion controller through a number of I/O port addresses on the ISA or PCI bus.

At the controller's base address is the primary bidirectional data transfer port. This port supports FIFO data passing in both send and readback directions. The NI motion controller has both a command buffer for incoming commands and a Return Data Buffer (RDB) for return data.

At offsets from the controller's base address are two read-only status registers. The flow of communications between the host and the NI motion controller is controlled by handshaking bits in the Communications Status Register (CSR). The MCS register provides instantaneous motion status of all axes.

Board Identification Parameter

Packets, Handshaking, and FIFO Buffers

Return Data Buffer

Board Identification Parameter

All VIs in NI-Motion have Board ID as the first parameter. Board ID is defined as boardID (u8)—the board identification number assigned by Measurement & Automation Explorer.

Board ID is a unique number assigned to each motion controller in the system by Measurement & Automation Explorer. After this assignment is made, all NI-Motion VIs use this board ID to send and receive commands and data to or from a specific NI-Motion controller. Most NI-Motion VIs also have a Board ID Out terminal which is provided for flow control. You can string together NI-Motion VIs by wiring the Board ID Out terminal of one VI to the Board ID terminal of the next VI.

Packets, Handshaking, and FIFO Buffers

This topic briefly describes how commands and data are passed between the host computer and the NI motion controller. This information is provided for reference purposes. The NI-Motion software provides drivers, DLLs, and C function libraries that handle the host-to-controller communications for you.

Data passed to or from the NI motion controller is handled in a packet format. A packet consists of a packet identifier word, command and data content, and a packet terminator word. This approach to communications enhances the integrity of data communications, speeds the processing of the transferred command and data, and organizes operation into powerful, high-level functions.

Each word in a packet is sent over the bus after checking the Ready-to-Receive (RTR) handshaking bit in the CSR. Refer to the Read Communication Status VI for the status cluster and more information about the status reported in the CSR.

Command and data packets are checked for packet format errors as the controller receives them. If the controller detects a packet error, it immediately reports the error by setting an error bit in the CSR. When the packet is received without error, the command and data is stored in a FIFO buffer.

This FIFO can hold up to 16 commands. The NI-Motion RTOS processes commands when it is not busy with higher priority tasks. In the unlikely event that the FIFO fills up before any commands can be processed, the host is held off with a Not-Ready-to-Receive condition.

Each command is processed and a determination is made whether to execute the command immediately, or store it away in a program to execute later. Commands are also checked for data and modal (sequence) errors at this time. Modal errors are flagged by setting the Error Message bit in the CSR. This modal error is functionally different from the packet communication error previously described. Refer to Errors and Error Handling for more information.

Return Data Buffer (73xx Controllers Only)

Data or status requested by the host is buffered in the Return Data Buffer (RDB) in FIFO mode. The FIFO is 16 commands deep.

When data exists in the RDB, the Ready to Send bit in the Communication Status Register (CSR) is set.

You can use the RDB either as a temporary buffer that holds a single return data packet, or as a small FIFO buffer. When the requested data is available in the RDB, use Read Return Data Buffer to retrieve it. You also can request multiple data items and leave them in the buffer for retrieval at a later time. If the RDB becomes full and cannot accommodate requested return data, NI-Motion generates an error and sets the Error Message bit in the CSR.

Errors and Error Handling

To minimize the possibility of erroneous system operation, VIs, packets, and data are checked for errors at multiple levels within the NI-Motion software and within the firmware that resides on the NI motion controller.

In a perfect system, errors should not be generated. However, during application development and debugging, errors are unfortunately quite common. NI-Motion offers an extensive error handling structure and utility VIs to allow you to quickly diagnose any error-generating situation. Refer to the Error & Utility VIs for information about specific VIs and Error Codes for a listing of errors returned by NI-Motion.

Error Codes

The following table lists all NI–Motion error codes, symbolic names, and a brief description of the associated error conditions.

Refer to the <u>Error & Utility VIs</u> for detailed descriptions of the VIs for error handling. Refer to <u>Errors and Error Handling</u> for more information about errors and error handling techniques.

Error Code	Symbolic Name	Description
-70001	NIMC_noError NIMC_readyToReceiveTimeoutError	No error. Ready to Receive Timeout. The controller is still not ready to receive commands after the specified timeout period. This error may occur if the controller is busy
		processing previous commands. If thi error persists, even when the controller should not be busy, contact National Instruments.
		Either this function is not supported by this type of controller or the controller
-70002	NIMC_currentPacketError	received an incomplete command packet and cannot

Error Codes

he following table lists all NI–Motion error codes, symbolic names, and a rief description of the associated error conditions.

efer to the <u>Error & Utility VIs</u> for detailed descriptions of the VIs for error andling. Refer to <u>Errors and Error Handling</u> for more information about errors and error handling techniques.

rror Code	Symbolic Name	Description
0	NIMC_noError	No error.
		Ready to Receive
		Timeout. The
		controller is still
		not ready to receive
		commands after
		the specified
		timeout period.
		This error may
70001	NIMC_readyToReceiveTimeoutError	occur if the
7 0001	TVIIVIO_ICUAY TORCCCIVC TIMICOULLITOI	controller is busy
		processing
		previous commands. If this
		error persists,
		even when the
		controller should
		not be busy,
		contact National
		Instruments.
		Either this
		function is not
		supported by this
		type of controller or the controller
70002	NIMC currentPacketError	received an
	<u></u>	incomplete
		command packet
		and cannot

Error Handling Techniques

In addition to using the error in and error out terminals on each motion VI, check for modal errors by calling the <u>Motion Error Handler</u> VI in the application. The Motion Error Handler VI checks the controller for errors and read and display errors from the Error Stack if they exist. Refer to the example VIs in the LabVIEW\Examples\Motion\NI-Motion directory to see how this VI is used.

How often to check for modal errors depends upon the application, but you can use the following list as a guideline:

- 1. Check for errors at the end of each major subroutine or functional block.
- 2. Check for errors at the end of an initialization procedure. Even better is to also check after each axis initialization. After executing a <u>Find Reference</u> VI, check to make sure the sequence completed successfully.
- 3. Include an error check in every status polling loop. Most applications include a polling loop to display motion status, position, velocity, and so on. This way you are assured of never missing an error.
- 4. During debugging, you can run an independent VI to check for errors. The Error Monitor VI in the Error & Utility palette is provided for this purpose. You can run this VI while another VI is executing motion VIs, and the Error Monitor displays modal errors.

For additional information about errors, refer to the following topics:

Onboard Variables

Fatal Hardware and Communications Errors

Error Message Stack

Modal errors generate an error message containing the command ID, resource ID, and error code that is pushed on the Error Message Stack on the NI motion controller and flagged in the Error Message (Err Msg) bit of the Communications Status Register (CSR). You can return a modal error message to the host by executing the <u>Read Error Message</u> VI.

For your convenience, the <u>Motion Error Handler</u> VI checks the CSR for errors, and if an error is present on the stack, reads the error message from the controller.

The Error Message Stack is organized as a last-in-first-out (LIFO) buffer so that the most recent error is available immediately. Older errors can be read with additional calls to the Motion Error Handler VI and are returned in the inverse order to which they were generated. When the stack is empty, the Error Message (Err Msg) bit in the CSR is reset.

The Error Stack can hold up to 10 errors. When the stack is full, which is an unlikely event, additional error messages are thrown away.

Fatal Hardware and Communications Errors

There are a few errors that, if detected, indicate a severe or fatal error condition. These include but are not limited to Board Failure Error, Watchdog Timeout Error, FPGA Program Error, DSP Initialization Error, I/O Initialization Error, and Ready to Receive Timeout Error.

Refer to Error Codes, for a complete list of error codes and possible causes.

Fatal errors are unlikely, but if they occur, try to clear them by resetting or power cycling. If this procedure does not clear the problem, refer to <u>Technical Support</u> and <u>Professional Services</u>.

Modal and Non-Modal Errors

NI-Motion can detect two types of errors—modal and non-modal. Non-modal errors are errors detected at the time of VI execution. This includes communications failures, parameter value errors, and bad Board ID error.

If a NI-Motion VI encounters a non-modal error, such as those listed above, communication with the controller is impossible, and the command is never sent to or accepted by the controller. The VI reports this error in the error out cluster. When VIs are wired together through the Error In and Error Out terminals, all VIs downstream of a VI causing a non-modal error do not execute. However, the Stop Motion VI always executes, even if another VI encountered an error earlier.

Modal errors, on the other hand, are errors that are not detected at the time of VI execution, because VIs can be buffered in the onboard FIFO. Furthermore, some VIs can be legal at one time and illegal at another, depending on the state or mode of the NI motion controller. This modal error structure also correctly detects errors generated by incorrectly sequenced VIs in onboard programs. Modal errors are placed in the onboard error stack, and you have to explicitly query the controller to check for errors. The Motion Error Handler VI checks the onboard error stack every time you call it.

VI Types and Parameters

In addition to the API functional organization, NI-Motion VIs can be categorized by common format, execution, and parameter characteristics.

Bitmapped versus Per-Resource VIs

Single and Double-Buffered Parameters

Input and Return Vectors

Onboard Variables

Bitmapped versus Per-Resource VIs

There are two basic types of NI-Motion VIs—those that operate on one resource at a time, and those that operate on multiple axes, vector spaces, I/O bits, and so on simultaneously.

Per-resource VIs typically send numeric values to, or read numeric values from, the selected axis or resource. They operate identically on each axis or member in the resource family.

In contrast, VIs that operate on multiple bits send and return bitmaps, where each bit (axis, vector space, I/O bit, and so on) is represented by one bit in the bitmap.

Some VIs set and reset bits in the bitmap using the Must On/Must Off (MOMO) protocol. This tri-state protocol allows you to set/reset one or more bits without affecting the other bits in the bitmap. Refer to any of the MOMO VI descriptions for complete information about this protocol.

Bitmapped VIs act on all bits simultaneously. Because each execution reconfigures all bits in the bitmap, do not use these VIs incrementally.

Input and Return Vectors

Many VIs that load values and virtually all readback VIs support vectoring. Load VIs, for example Load Target Position, take an input vector that specifies the source of the data, either immediate (using the VI input parameters), from a general-purpose onboard variable, or from an indirect variable. Read VIs, for example Read Position, take a return vector that specifies the destination for the returned data, either the host computer, an onboard variable, or an indirect variable.

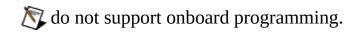
The ability to use variables in motion control VIs is one of the powerful features of the NI-Motion onboard programming environment. You can read data from a resource into a variable, scale or perform some other calculation on the value, and then load the new value as a trajectory or other motion parameter. All <u>Data Operations VIs</u> take data from variables and return the result through a return vector, typically to another variable.

Note Data returned to the host by a return vector of 0xFF is actually left in the Return Data Buffer (RDB). Read data from the RDB using Read Return Data Buffer.

In addition to specifying a variable directly, you can use indirect variables to reference a variable indirectly, much like a pointer in C. Refer to <u>Onboard Variables</u> for more information.

Input and return vectors are very useful when writing onboard programs but have little or no use in programs running on the host computer. For this reason, the default value for input vector is immediate (0xFF).

Note The return vector must be 0xFF on the 7330 and 7390, because they



Refer to <u>Communication between the Host Computer and the NI-Motion</u> <u>Controller</u> for information about the RDB.

Onboard Variables

NI-Motion supports 120 general-purpose variables (0x01 through 0x78) for use in onboard programs. Variables are 32 bits wide and can hold either signed (i32) or unsigned (u32) values. Variables can be referenced in input and return vector parameters. Data Operations VIs use variables exclusively for input operands and the output result.

In general, most VIs have a single data parameter that fits into a single 32-bit variable. If the VI uses only a 16-bit data value, it is in the least significant portion (right-most) of the 32-bit variable. However, some VIs with input or return vectors have more than one data parameter. As a general rule, each parameter, regardless of size, requires it own variable. For these VIs, the vector points to the first variable in a sequential group of variables. Parameters are then associated with variables in sequential order.

A few advanced VIs handle variables differently from the previous description. Refer to <u>Using Inputs and Return Vectors with Onboard Variables</u> for more information about variables and vectors.

Indirect Variables

Indirect variables provide functionality similar to pointers in a programming language such as C. When you use an indirect variable (0x81 through 0xF8) as an input vector or return vector, the NI-Motion software looks at the contents of the indirect variable and uses that value as the input or return vector. Indirect variables 0x81 through 0xF8 correspond to variables 0x01 through 0x78. So when you specify an input vector of 0x81, the NI motion controller uses the contents of variable 0x01 as the input vector. This can be very useful in looping structures in onboard programs, as well as changing the input values to VIs dynamically.

Example

Suppose the values of onboard variables 1 to 10 are as follows:

0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A

0 7 0 0 0 10000 0 0

In this example, reading the indirect variable 0x82 returns the value 1000.

Note You can save the entire set of onboard general-purpose variables to Flash ROM with the <u>Save Default Parameters</u> VI.

Variable Array

Variable arrays are used when reading or writing a buffer of data to or from onboard memory using the <u>Read Buffer</u> or <u>Write Buffer</u> VIs.

In memory, the buffer occupies n + 2 variables, where n is the number of buffer elements read. The first variable contains the number of values read from the buffer. The second variable contains the error code of the read operation. If this value is zero, then no error occurred. The next n elements are the values read from the array. The read buffer accounts for the two additional value. However, when you are writing a buffer using a variable array, the second parameter is the regeneration mode, not an error code.

Example

Suppose the onboard variable looks like the following:

0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 ... 5 0 0 0 0 0 0 0 ...

Further, suppose that there is a buffer containing the following elements:

12345

If you were to read the first five elements of the buffer to onboard memory location 0x81, the onboard memory would look like the following:

0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA 0xB 0xC ...
5 0 0 0 5 0 1 2 3 4 5 0 ...

0x5 contains the number of elements to read. 0x6 displays the errors that occurred. In this case, the location contains a 0, meaning no errors occurred.

0x7–0xB are the five elements that were read.

Reflective Memory VIs

Some of the Advanced Read VIs use reflective memory to return status and data information to the controller. Unlike the <u>trajectory control</u> VIs that also return this information, these specialized VIs read status and data information from the register and send it back in a much higher response time.

These VIs should *not* be used for event checking. Only use the information returned by these VIs for display purposes. This is because the tradeoff for significantly faster response time is that these VIs are not serialized with other operations. VIs that return data information are read from a register on the controller and refreshed every 5 ms. VIs that return status information are refreshed as soon as possible. This configuration makes the data and status information quickly accessible—in about 40 μ s as opposed to > 2 ms with the trajectory control VIs. However, this also means that you can read status information sooner than the VI has time to execute. For example, a call to Start Motion followed immediately by a call to the **move complete** attribute of the Read Execution Status VI may return a True value because the Start Motion VI may not have started before the move complete status information is returned.

Note VIs using reflective memory are not supported by the NI SoftMotion Controller.

The following table lists the VIs using reflective memory, which controller has reflective memory support, and which attributes use reflective memory, if applicable:

Function	7330	7340	7350	7390
Read All Axis Data			Y	Y
Read All Axis Status			Y	Y
Read Axis Data (all attributes)			Y	Y
Read Axis Status (all attributes)			Y	Y
Read Capture Compare Data				
Captured Position			N	N

Position Capture Occurred			Y	Y
Position Compare Occurred			Y	Y
Read Coordinate Data (all attributes) Read Coordinate Status (all attributes) Read Coordinate Position (all attributes) Read Digital I/O Data				Y Y Y
Output Active State	N	N	N	N
Input Active State	N	N	N	N
Output Active	N	N	Y	Y
Input Active	N	N	Y	Y
Configure As Input	N	N	N	
Read Encoder Data (all attributes)		_	Y	Y
Read Motion I/O Data				
Forward Limit Active	N	N	Y	Y
Reverse Limit Active	N	N	Y	Y
Forward Software Limit Active	N	N	Y	Y
Reverse Software Limit Active	N	N	Y	Y
Home Input Active	N	N	Y	Y
Inhibit-In Active	N	N	N	N
In-Position Active	N	N	N	N

Drive Ready Active N N N N N Inhibit-Out Active N N N N N

Single and Double-Buffered Parameters

Almost all NI-Motion parameters are either single-buffered or double-buffered on the controller. Single-buffered parameters take effect immediately upon VI execution and remain in effect until they are overwritten with another call to the VI that loaded or set them. It is not necessary to constantly reload single-buffered parameters each time you deal with an axis, vector space, or resource. The obvious exception to this is action commands like Start Motion, Stop Motion, Find Reference, and so on, which must be called each time.

Most trajectory control parameters are double-buffered. You can load these parameters on the fly without affecting the move in process. They do not take effect until the next Start or Blend Motion VI is executed. Like single-buffered parameters, the controller retains the values so they do not have to be loaded before each move unless you want to change their values.

<u>Load Breakpoint Position</u> and <u>Load Breakpoint Modulus</u> are the only non-trajectory parameters that are also double-buffered. They have no effect until you execute a subsequent <u>Enable Breakpoint Output</u> VI.

Using Input and Return Vectors with Onboard Variables

This table gives additional information about how to use input and return vectors in conjunction with onboard variables. The following table lists VIs with more than one data parameter that require multiple variables when using vectors. The Maximum Variable Number column lists the highest variable number that a vector can point to and still have room for all the data without exceeding the total variable space. This table also contains notes that highlight a few special cases where the number of parameters does not equal the number of variables.

VI Name	Vector Type	Number of Parameters	Number of Variables Required	Maximum Variable Number	Comments
Load All PID Parameters	Input	8	8	0x71	
Load Velocity Filter Parameter	Input	2	2	0x77	
Load Velocity in RPM	Input	1	2 (<u>Note 1</u>)	0x75	National Instruments recommends using Load Velocity for onboard variables.
Load Accel/Decel in RPS/sec	Input	1	2 (<u>Note 1</u>)	0x75	
Load Vector Space Position	Input	3	3	0x76	
					National Instruments

About the NI-Motion Software

NI-Motion software provides a comprehensive API you use to control the NI motion controllers. NI-Motion software combined with the NI motion controllers provide functionality and power for integrated motion systems for use in laboratory, test, and production environments.

For programming ease, NI-Motion software is enhanced by a toolbox of drivers, LabVIEW VI libraries, and Windows dynamic link libraries (DLLs) that implement the entire NI-Motion API.

The NI-Motion software package includes Measurement & Automation Explorer (MAX), which associates physical bus addresses with board IDs, which are used in programs to distinguish between controllers. MAX also verifies that the NI-Motion controller is installed correctly and is communicating with the host computer.

The motion configuration environment under MAX allows you to configure initialization settings that are used to initialize the NI motion controller. MAX also provides an easy interactive environment for testing and troubleshooting the motion system. Refer to the MAX online help for more information.

For application development, the NI-Motion software package includes example programs to help get you up and running quickly.

A separate NI-Motion software package includes a complete motion C library, with examples, for use with C or C++ development environments.

Initialization

The configuration panels in Measurement & Automation Explorer (MAX) allow you to set a large number of NI motion controller attributes such as motor type, limit switch polarity, position mode, and maximum velocity. Most of the attributes are one-time configuration settings that do not change. By setting these attributes one time in MAX, your programming is much simpler, and you can concentrate on the motion-specific part of your application.

When you call the <u>Initialize Controller</u> VI, all the attributes stored in MAX are retrieved and sent to the motion controller. After the controller is initialized, you do not need to initialize each time you run the application. National Instruments recommends you call Initialize Controller one time after restarting the computer or resetting the motion hardware. The motors are temporarily disabled (killed) while this initialization process takes place.

In most cases, you call Initialize Controller with the Settings Name terminal unwired, which causes the current initialization settings in MAX to use. You also can initialize the controller with alternate initialization settings by passing in the name of the alternate setting as specified in MAX.

Click the icons for VI descriptions.



Initialize Controller

Subpalettes

Axis & Resource Configuration VIs Advanced Trajectory VIs

Start & Stop Motion VIsMotion I/O VIsFind Reference VIsBreakpoint VIs

Analog & Digital I/O VIs

High-Speed Capture VIs

Advanced VIs Error & Utility VIs

<u>Trajectory Control VIs</u>
<u>Arc VIs</u>

<u>Advanced Error & Utility VIs</u>

<u>Onboard Programming VIs</u>

Gearing VIs Object Management VIs

Buffered Operations VIs

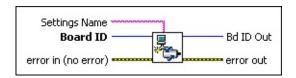
Data Operations VIs

Initialize Controller

Initializes the specified controller according to the settings as configured in

Measurement & Automation Explorer.

Device Compatibility



Details

Settings Name is the name of the configuration settings, such as *Default 7330 Settings*, specified in Measurement & Automation Explorer (MAX). To use the current configuration settings, leave Settings Name unwired or pass NULL for the Settings Name parameter. The current settings are the settings that are configured in MAX.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

Using This VI

This VI allows you to initialize the controller to a known state based upon the configuration settings stored in the Measurement & Automation Explorer database. This VI pulls values from the configuration database and calls dozens of NI-Motion VIs, configuring the controller for you in one simple step.

If the Initialize Controller VI returns an error, you can call the Get Last Error VI to determine the specific NI-Motion VI and resource generating the error. You can then call the <u>Get Error Description</u> VI, which formats a string for you with the VI name, resource ID, and error description.

Note If you are running a NI-Motion startup application on ETS (PharLap) systems, the first VI in your application must be either the Clear N Power Up Status or Initialize Controller VI. Calling any other VI first may result in error –70012 (NIMC_badBoardIDError) because the driver may not have loaded before a NI-Motion VI is called.



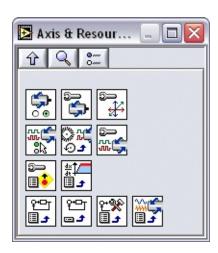
Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Axis & Resource Configuration VIs

These VIs give you access to some of the most powerful features of NI-Motion. They allow you to map encoder, ADC, and DAC resources to various axes, configure axes for servo or stepper control, and combine axes into 2D and 3D vector spaces for advanced motion control applications.

These axis and resource configuration VIs are typically executed one time during system initialization. You can call most of them at any time to reconfigure the motion control system on-the-fly. All of the Axis & Resource Configuration VIs have default values that provide good starting points for many motion control applications.

Click the icons for VI descriptions.



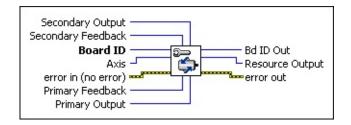
Enable Axes
Configure Axis Resources
Configure Vector Space
Set Stepper Loop Mode
Load Counts/Steps per Revolution
Configure Stepper Output

Configure Move Complete Criteria
Load Velocity Filter Parameter
Load All PID Parameters
Load Single PID Parameter
Load Advanced Control Parameter
Load Commutation Parameter

Configure Axis Resources

Configures an axis by defining its feedback and output resources.

Device Compatibility



Details Remarks

Secondary Output is the number for an optional secondary output resource being mapped to the axis. This is an optional command output. Select None to configure no secondary output resource.
Secondary Feedback is the number for an optional secondary feedback resource being mapped to the axis. If a secondary feedback resource is mapped, it is used for velocity feedback (Kv). Select None to configure no secondary feedback resource. Refer to the <i>Velocity Feedback Gain</i> section of <u>Load Single PID Parameter</u> for more information about Kv.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis for which to load PID parameters. Valid axis numbers are 1 through 15 for all NI motion controllers. On controllers with less than eight axes, configuring non-existent axes has no effect.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs

normally only if no incoming error exists. Otherwise, the VI passes the

Using This VI

The Configure Axis Resources VI defines the feedback and output devices for an axis. You can configure up to two feedback resources and two output resources for each axis. This flexible mapping of resources to axes allows for advanced servo and stepper configurations such as: independent velocity and position feedback devices (dual-loop control), dual DAC outputs with different offsets, and changing feedback on the fly.

The various feedback and output resources on the motion controller have different interface, performance, and functionality characteristics. This VI allows you to define the axis and tailor its performance.

The following table lists valid resources:

Resource Type	7330	7340	7350	7390
Primary Feedback	Encoder/ADC	Encoder/ADC	Encoder/ADC	Encoder
Secondary Feedback	Encoder	Encoder	Encoder/ADC	
Primary Output	Stepper	Stepper/DAC	Stepper/DAC	Stepper
Secondary Output		DAC	DAC	
Tip Refer to Mot	ion Resources	for more inforr	nation about re	source IDs.
Tip For informat	ion about the re	esource IDs, re	fer to <u>Motion R</u>	Resources.

This VI must be called for each axis that is used by an application prior to enabling the axis. The factory default mapping of resources to axes is as follows:

Axis	Primary Feedback	Secondary Feedback	Primary Output	Secondary Output
1	0x21 (Enc 1)	0	0x31 (DAC 1)	0
2	0x22 (Enc 2)	0	0x32 (DAC 2)	0
3	0x23 (Enc 3)	0	0x33 (DAC 3)	0
4	0x24 (Enc 4)	0	0x34 (DAC 4)	0
г	0 ~ 2 Γ (E_{22}, Γ)	0	02F (DACF)	Λ

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- If you are using a 7330, 7340, or 7390 motion controller, you cannot configure an axis when any axes are enabled. You must first disable all axes using the Enable Axes VI.
- To change feedback on the fly (NI 7350 controllers only), call this function when the axis is not moving (move complete).

Note The feedback resources used for switching feedback on the fly must meet the following conditions:

• The feedback device must be enabled.



• The feedback device being switched to must not be used by another axis.

Refer to <u>Switching Feedback on the Fly</u> in the *NI-Motion Help* for detailed information about configuring your system to switch between two feedback devices.

Example 1

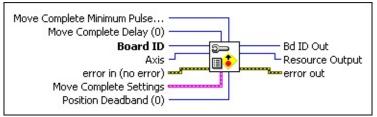
To change axis 3 to use the fourth encoder channel and the first DAC output, call the Configure Axis Resources VI with the following configuration:

Axis	Primary		Secondary Foodback	Primary	Secondary
	Feedback		Feedback	Output	Output
3	0x24 (Enc 4)	0		0x31 (DAC 1) 0	

Configure Move Complete Criteria

Configures the criteria for the Move Complete status to be True.





Details

Move Complete Minimum Pulse (0) is the minimum time in ms that the
move complete status must stay true. This parameter allows you to
enforce a minimum pulse width on the move complete status even if the
axis is started again. This is the amount of time you have to read the move
complete status before it is reset. The range is 0 (default) to 255 ms.
Move Complete Delay (0) is a programmable settling time delay in ms.
You can program it from 0 (default) to 255 ms.
Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Axis is the axis for which to load PID parameters. Valid axis numbers are
1 through 15 for all NI motion controllers. On controllers with less than
eight axes, configuring non-existent axes has no effect.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already
occurred, this VI returns the value of error in in error out . The VI runs
normally only if no incoming error exists. Otherwise, the VI passes the
error in value to error out . The error in cluster contains the following

Using This VI

The Configure Move Complete Criteria VI defines the conditions for reporting a move complete. When a move is complete on an axis, the corresponding bit in the MCS register is set. Refer to the <u>Read Move Complete Status</u> VI for information about reading the MCS register. Also, the <u>Read per Axis Status</u> VI can provide you with move complete status on a specified axis.

This VI allows a great deal of control over when and how a move is considered complete. The Criteria bitmap contains five bits to set the conditions used to determine the Move Complete status. The first two, Profile Complete and Motor Off, are logically OR'd to provide the basis for Move Complete. The Profile Complete bit is always set and cannot be disabled. When the axis trajectory generator completes its profile, this condition is satisfied. If the Motor Off bit is set, any condition that causes the axis to turn its motor off (a kill or following error trip) satisfies this basic requirement for Move Complete. In other words, either Profile Complete OR Motor Off must be True for Move Complete to be True.

The next three criteria, Run/Stop, Delay, and In-Position, are optional conditions that are logically ANDed to further qualify the Move Complete status. If the Run/Stop bit is set, the axis must also be logically stopped for the move to be considered complete. Refer to the <u>Load Run/Stop Threshold</u> VI for information about the Run/Stop status.

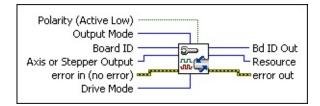
If the Delay bit is set, the axis must wait a user-defined delay after the other criteria are met before the move is considered complete. The user-defined Move Complete Delay parameter is typically used to wait the mechanical settling time so that a move is not considered complete until vibrations in the mechanical system have damped out. It also can be used to compensate for PID pull-in time due to the integral term. This pull-in is typically at velocities below the Run/Stop threshold.

Finally, if the In-Position bit is set, the axis checks its final stopping position versus its target position and only sets the Move Complete status if the absolute value of the difference is less than the in-position deadband.

Configure Stepper Output

Configures the drive mode, output mode, and polarity of a stepper output.

Device Compatibility



Details

resource.

Polarity sets the polarity for the stepper output. Select TRUE (default) for active low. Select FALSE for active high.

When configured as active low, the output is active when there is a low signal on the output pin. Conversely, active high means that the output is active when there is a high signal on the output pin.

Output Mode specifies the mode as Step and Direction or Clockwise/Counterclockwise.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis or Stepper Output is the axis or stepper output to configure. When sent to a stepper axis, this VI configures the mapped stepper output.

error in (no error) describes error conditions that occur before this VI

Alternatively, you can execute this VI directly on the stepper output

Use this VI to configure a stepper output to correctly interface with a stepper driver. NI-Motion supports the two industry standards for stepper control outputs. The most popular mode is **Step and Direction**, where one output produces the step pulses and the other output produces a direction signal.

In **Clockwise and Counterclockwise** (**CW/CCW**) mode, the first output produces pulses when moving forward, or **CW**, while the second output produces pulses when moving reverse, or **CCW**.

In either mode, you can set the active polarity with the polarity bit as active low or active high. For example, in **Step and Direction** mode, the polarity bits determine whether a high direction output is forward or reverse. It also determines the resting states of outputs when they are not pulsing.

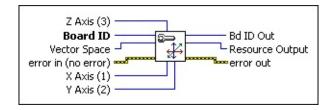
The Configure Stepper Output VI is typically called for each stepper axis prior to using the axis. After the modes and polarity are set, they remain in effect until changed. You can execute this VI at any time.

For the 7350 controller, you can set the stepper output **Drive Mode** to **Open Collector** or **Totem Pole**. On the 7330/40, stepper output is **Open Collector** and cannot be changed. On the 7390, stepper output is isolated so **Drive Mode** does not apply and cannot be changed. For more information about **Open Collector** and **Totem Pole** drive modes, refer to the documentation for your 7350 device.

Configure Vector Space

Defines the axes that are associated with a vector space.

Device Compatibility



Details

Z Axis (3) is the physica The default is 3.	l axis (1 through	15) to act as	the logical	z axis.
The default is 3.				

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Vector Space is the vector space to configure.

Note For NI 7330/40/90 motion controllers, valid vector space numbers are 0x11 (default), 0x12, and 0x13.

Note For the NI 7350 motion controllers, valid vector space numbers are 0x11 (default), 0x12, 0x13, and 0x14.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

Use the Configure Vector Space VI to group axes into a vector space. A vector space defines an x and y (2D) or x, y, and z (3D) coordinate space. You can map any physical axis to the logical x, y, and z axes to coordinate the motion of multiple axes. The total number of vector spaces NI-Motion supports in your motion control system can be calculated using the following formula:

Tip To unconfigure a vector space, set all axes to **None**.

total vector spaces = [number of activated axes/2].

After it is configured, you can use the Vector Space number in all VIs that support vector spaces. Vector spaces are used in 2D and 3D linear interpolation with vector position, vector velocity, vector acceleration and deceleration, and vector operation mode. They are also used in circular, helical and spherical arc moves. You can start, blend, and stop vector spaces just like axes. You can even synchronously start multiple vector spaces for multi-vector space coordination.

Many status and data readback VIs also operate on vector spaces. You can read vector position, vector velocity, vector blend status, and so on, or you can read per-axis values and status for the axes within the vector space.

While vector spaces can be comprised of three axes, it is possible to define two-axis or even one-axis vector spaces. These vector spaces function properly for all VIs that do not require a greater axis count.

You can use other complex motion control VIs with vector spaces, including electronically gearing an independent axis to a master axis contained within a vector space definition.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Example

Enable Axes

Enables the operating axes and defines the PID and trajectory update rate.

Axis Bitmap (Disabled)
Sample PID Rate
Board ID

error in (no error)

But Dout

error out

Device Compatibility

<u>Details</u> <u>Remarks</u>

Axis Bitmap is the bitmap of enabled axes.

EXECUTE: Axis 1—If true, axis is enabled. If false, axis is disabled (default).

Axis 2—If true, axis is enabled. If false, axis is disabled (default).

EXECUTE: Axis 3—If true, axis is enabled. If false, axis is disabled (default).

EXECUTE: Axis 4—If true, axis is enabled. If false, axis is disabled (default).

EXECUTE: Axis 5—If true, axis is enabled. If false, axis is disabled (default).

EXECUTE: Axis 6—If true, axis is enabled. If false, axis is disabled (default).

EXECUTE: Axis 7—If true, axis is enabled. If false, axis is disabled (default).

Axis 8—If true, axis is enabled. If false, axis is disabled (default).

Sample PID Rate is the PID control loop and trajectory generator update rate, for NI 73xx motion controllers. For stepper axes, this parameter also determines how often the step generator is updated. The range for this parameter is 0 to 7, with a default of 3 (250 μ s).

Update Rate = (Sample PID Rate + 1) x 62.5 μ s

Sample PID Rate Update Rate

0 62.5 μs

1 125 μs

The Enable Axes VI is used to enable the specific axes required for the application and set the servo (and stepper) update or sample rate. An axis is enabled for operation by a True in the corresponding location in the bitmap. Only enabled axes are updated and there is a direct correspondence between the number of enabled axes and the fastest update rate allowed.

The fastest update rate is only achievable when all axes are single-feedback servo axes and no extra encoders are enabled.

Caution Update rates that are too fast for the number of axes, stepper outputs, and/or encoders enabled generate an error and the previous setting remains in effect. Refer to Errors and Error Handling for information about errors and error handling.

The Enable Axes VI automatically enables the feedback devices mapped to the enabled axes. It is not necessary to explicitly enable the encoders or ADC channels before enabling the axes. Axes must be killed before they are disabled or the error NIMC_wrongModeError is generated. Refer to the Enable Encoders and Enable ADCs VIs for more information about enabling and disabling these resources when you are using them not mapped to an axis.

Note ADC channel scan rate is affected by the number of channels
enabled. This could limit the effective update rate (for axes with analog feedback). Refer to the Enable ADCs VI for more information.
Caution Illegally configured axes cannot be enabled, and attempting to do so generates an error. For example, an attempt to enable a servo axis that does not at least have its Primary Feedback device mapped generates an error.

You also can set the update rate slower than the maximum. This is useful in many applications to scale the effective range of the PID control loop parameters and/or to improve stability. Refer to the <u>Load Single PID Parameter</u> VI for more information about the PID parameters affected by the update rate.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The ADC channel scan rate is affected by the number of channels enabled. This could limit the effective update rate (for axes with analog feedback). Refer to the Enable ADCs VI for more information.
- This VI is used to enable the specific axes required for the application and set the servo (and stepper) update or sample rate. An axis is enabled for operation by a True in the corresponding location in the bitmap. Only enabled axes are updated and there is a direct correspondence between the number of enabled axes and the fastest update rate allowed.

Number of Enabled Axes (7330/40/90)	Fastest Update Rate	Number of Enabled Axes (7350)	Fastest Update Rate
1	62 µs	1–2	62 μs
2	125 μs	3–4	125 μs
3	188 μs	5–6	188 μs
4	250 μs	7–8	250 μs

• The fastest update rate is achievable only when all axes are single-feedback servo axes and no extra encoders are enabled.

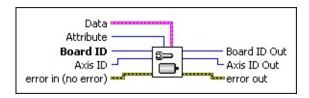
Caution Update rates that are too fast for the number of axes, stepper outputs, and/or encoders enabled generates an error and the previous setting remains in effect. Refer to Errors and Error Handling for information about errors and error handling.

• The Enable Axes VI automatically enables the feedback devices mapped to the enabled axes. It is not necessary to explicitly enable the encoders or

Load Axis Configuration Parameter

Loads data on a per axis basis for an axis configuration object on the motion controller.

Device Compatibility



Details Remarks

Data is a cluster of variables that contains the values for the selected Attribute .
Integer is 32-bit integer data.
Double is 64-bit floating point data.
Boolean is Boolean data.
Attribute is the attribute you want to load:
Enable enables the currently selected axis. Set the Boolean element of the Data cluster to TRUE to enable the axis, and to FALSE to disable the axis.
Board ID is a unique number assigned by Measurement & Automation Explorer used to send and receive commands and data to or from a specific NI motion controller.
Axis ID is the axis you are controlling.

error in (no error) describes error conditions that occur before this VI

Use this VI to load the data on a per axis basis for the Axis Configuration object on the motion controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

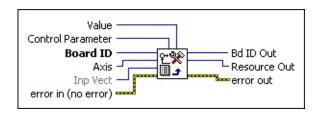
The following list includes considerations you must make when you are using this VI with a 73xx controller:

• Because the number of axes enabled depends on the control loop rate, use the Enable Axes VI to enable/disable an axis for 73xx controllers.

Load Advanced Control Parameter

Sets an advanced control loop parameter for a given axis.

Device Compatibility



Details Remarks

Value is the value to load for the specified parameter.

Control Parameter is one of the following control loop parameters to set:

Static Friction Mode

Static Friction Max Deadzone

Static Friction Min Deadzone

Static Friction Forward Voltage Offset

Static Friction Reverse Voltage Offset

PID Rate Multiplier

Notch Filter Frequency

Notch Filter Bandwidth

Notch Filter Enable

Lowpass Filter Cutoff Frequency

Lowpass Filter Enable

Second Set of PID Parameters

Decel ID:

Use the Load Advanced Control Parameter VI to configure an advanced control loop parameter for a given axis.

Enable **Static Friction Mode** to configure static friction (stiction) compensation on the motion system. Stiction compensation adjusts for increased power needed to overcome static friction when beginning a move from a standstill. Set **Static Friction Mode** to 1 to enable, or 0 to disable, static friction compensation.

The **Max** and **Min Deadzones** are the windows around the target position that determine its sensitivity to external forces that try to move it from its position. Inside the maximum deadzone window, an axis can be moved without response. As soon as the movement causes the axis to exceed the maximum deadzone, the control loop responds by correcting the position back to within the window defined by minimum deadzone. The range is 1 to 32,767.

Forward and **Reverse Offset Voltages** are the voltages necessary to overcome static friction when beginning a move. The voltages can be the same for both directions, but can be different under some circumstances. For example, if an axis moves against gravity in one direction and with gravity in the other direction, the offset voltage necessary to overcome static friction against gravity is higher than that of the direction that moves with gravity.

The **PID Rate Multiplier** specifies a multiplier to slow the PID Rate. The default is 1 (no change), and the range is 1 to 10. For example, with a 250 µs **PID Rate** and a **PID Rate Multiplier** of 4, the effective control loop rate is shown in the following equation:

effective rate = PID Rate \times PID Rate Multiplier = 250 μ s \times 4 = 1 ms

Notch filters remove a narrow band of frequencies around a given point to filter output signals that could cause system instability. Set **Notch Filter Enable** to 1 to enable, or 0 to disable, the filter. Use **Notch Filter Frequency** to specify the frequency of the signal that you want to remove. The range for the frequency is

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx controller:

- **Static Friction** and **PID Rate Multiplier** parameters can be set only on 7340/50 motion controllers.
- **Notch Filter** and **Lowpass Filter** parameters can be set only on 7350 motion controllers.
- **Second Set of PID Parameters** is valid only on 7350 controllers.
 - Only the **Disabled** and **Switch Feedback** options are valid.
 - If the Switch Feedback option is selected, the control loop toggles between sets of PID parameters each time the feedback is changed. For example, if you are using encoder 1, the control loop uses the primary set of PID parameters. If you switch to ADC 1, the control loop uses the secondary set of PID parameters. If you switch back to encoder 1, the control loop switches back to the primary set of PID parameters.

NI SoftMotion Controller Considerations

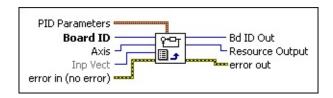
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• If the **Switch Feedback** option is selected, the control loop uses the primary set of PID parameters when you use a primary resource and uses the secondary set of PID parameters when you use an auxiliary resource. For example, if you use encoder 1, the control loop uses the primary set of PID parameters. If you use auxiliary encoder 1, the control loop uses the secondary set of PID parameters.

Load All PID Parameters

Loads all 8 PID control loop parameters for an axis.

Device Compatibility



Details

PID Parameters data structure contains all eight PID parameters in the
following structure:
Kp —Proportional Gain
Ki —Integral Gain
IL—Integration Limit
Kd —Derivative Gain
☐ Td —Derivative Sample Period
Kv —Velocity Feedback Gain
Aff—Acceleration Feedforward Gain
Vff —Velocity Feedforward Gain
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis for which to load PID parameters. Valid axis numbers are
1 through 15 for all NI motion controllers. On controllers with less than eight axes, configuring non-existent axes has no effect.
Inp Vect indicates the source of the data for this VI. Available input

vectors include immediate (0vFF) variable (0v01 through 0v78) or

The Load All PID Parameters VI allows you to set all 8 PID parameters at the same time for a given axis. You can call this VI at any time. However, it is typically used during initialization to configure and tune an axis. NI-Motion also offers a Load Single PID Parameter VI, which you can use to change an individual value on-the-fly without having to worry about the other unchanged values.

Refer to the motion controller documentation for an overview of the PID control loop on the NI motion controller and to the Load Single PID Parameter VI for descriptions on the individual PID parameters and their ranges.

Note If you are doing onboard programming and are using Inp Vect to get the data this VI needs, this VI reads the variables starting at the memory address pointed to by Inp Vect in the following order: Kp, Ki, IL, Kd, Td, Kv, Aff, Vff.

Loading a Second Set of PID Parameters

To load a second set of PID parameters, use the Load All PID Parameters and Load Single PID Parameter VIs to load the second set of PID values. Instead of using regular axis resource IDs, such as axis 1 = 0x1 (1), axis 2 = 0x2 (2), and so on, use the following alternate resource IDs: axis 1 = 0xA1 (161), axis 2 = 0xA10xA2 (162).

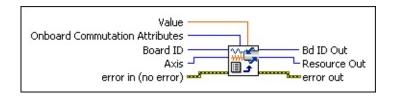
Use the <u>Load Advanced Control Parameter</u> VI to enable the new set of PID parameters based on the condition specified in the **value** parameter.

Refer to the *Remarks* section of the <u>Load Advanced Control Parameter</u> topic for more information about how to change feedback while the axis is enabled.

Load Commutation Parameter

Sets a commutation parameter for a given axis.

Device Compatibility



Details

parameters:

Attribute is one of the following commutation attributes to set: Initialization Type
Find Zero Voltage
Find Zero Time
Direct Set Phase
Electrical Cycle Counts
Hall Sensor Type
Commutation Mode
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis to configure.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs

normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

Use the Load Commutation Parameter VI to configure a commutation parameter for a given axis.



Note Only 7350 controllers support the Load Commutation Parameter VI.

Initialization Type specifies how the controller initializes the commutation phase. Select one of the following values:

Value Description

- 0 **Hall Sensor**—Initializes the phase based on the Hall sensor's input.
- **Shake and Wake**—Initializes the phase by driving the system to the 1 zero degree commutation phase.
- **Direct Set**—Sets the commutation phase directly with the phase angle 2 specified.

Caution Phase initialization is performed when the axes are enabled. If you use **Shake and Wake**, the system can move during the initialization process.

Find Zero Voltage specifies the voltage to use with **Shake and Wake** initialization. The range is 0 to 10. **Find Zero Time** specifies the duration of **Shake and Wake** initialization in milliseconds. The range is 0 to 10,000 ms.

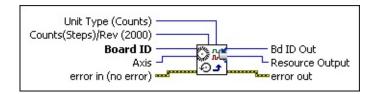
Direct Set Phase specifies phase angle to set with **Direct Set** initialization. For **Direct Set**, the controller sets the current position as the specified phase angle. National Instruments recommends this initialization method only for custom systems with a known initial phase angle. The range for the phase angle is 0 to 359°.

Electrical Cycle Counts specifies the encoder counts for the commutation cycle. This information is usually listed in the motor or stage documentation. Some motor manufacturers present this information as the number of poles. In such cases, convert number of poles to counts per electrical cycle with the following equation. The range is 2 to 268 435,000 counts

Load Counts/Steps per Revolution

Loads the quadrature counts or steps per revolution for an axis.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis is the axis to control.

Counts(Steps)/Rev is either the quadrature counts per revolution for the encoder mapped to the axis or the number of steps (full, half, or microstep) per motor revolution. The default is 2,000.

Unit Type is the type of information, counts or steps, to load. The default is counts.

Valid values are:

(1)

Counts—Use to define counts/revolution for a quadrature encoder.

Steps—Use to define steps/revolution for a stepper motor.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs permally only if no incoming error exists. Otherwise, the VI passes the

Use the Load Counts/Steps per Revolution VI to load any feedback value (counts or volts) per unit of measure or to load the number of steps per motor revolution. For encoders, this is typically in units of quadrature counts per motor revolution, but can be counts per inch, per cm, or per any unit of measure. For analog feedback, it can be in units of scaled voltage. Steps can be full steps, half steps, or microsteps depending upon how you have the external stepper driver and motor configured.

This parameter must be correctly loaded before you call the <u>Load Velocity in RPM</u>, <u>Load Accel/Decel in RPS/sec</u>, <u>Load Velocity Threshold in RPM</u>, <u>Read Velocity in RPM</u>, <u>Load Move Constraints</u>, or Find Reference VIs.

The Find Reference VI searches for the encoder index for one revolution as defined by this VI. Therefore, another useful unit of measure is counts per index period. Linear encoders often have indexes every inch or every centimeter.

Closed-loop stepper functionality relies on the ratio of counts to steps and not on the absolute values of counts or steps per revolution. For closed-loop operation, any unit of measure (UOM) that allows you to enter both counts per UOM and steps per UOM that are within their valid ranges work. Notice that this VI must be called twice: one time to set up the stepper motor steps/rev, and again to set up the quadrature encoder counts/rev.

Caution For closed-loop stepper controllers, steps per revolution/counts per revolution must be in the range of 1/32,767 < steps/counts < 32,767.

The Load Counts/Steps per Revolution VI simply loads a scale factor that affects subsequently loaded and read back values of velocity and acceleration.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx controller:

- Counts(Steps)/Rev is interpreted as number of encoder counts per revolution for encoder feedback or number of ADC codes (LSBs) per revolution for analog feedback.
- The range for the **Counts(Steps)/Rev** parameter is 2 to $2^{28} 1$ with a default value of 2,000.

NI SoftMotion Controller Considerations

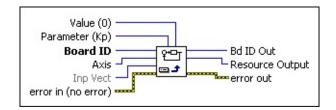
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The range for the **Counts(Steps)/Rev** parameter is 1 to $2^{31} 1$ with a default value of 1.
 - **Note** This parameter must be correctly loaded before you load target positions or write contour position information to the controller or camming table.
- This VI determines the unit used when loading velocity. You must set the Counts(Steps)/Rev value to 1 to use this VI to load velocity values in counts/sec and load acceleration values in counts/sec². If you set Counts(Steps)/Rev to any value other than 1, you must use the Load Move Constraints VI to load velocity and acceleration in counts.

Load Single PID Parameter

Loads a single PID control loop parameter for an axis.

Device Compatibility



a specific NI motion controller.

Details

Value (0) is the value to load for the selected PID parameter.

	PID Parameter	Abbreviation	Data Range	Default
	Proportional Gain	Kp	0 to 32,767	100
	Integral Gain	Ki	0 to 32,767	0
	Integration Limit	Ilim	0 to 32,767	1,000
	Derivative Gain	Kd	0 to 32,767	1,000
	Derivative Sample Period	Td	0 to 63	2
	Velocity Feedback Gain	Kv	0 to 32,767	0
	Acceleration Feedforward Gain	Aff	0 to 32,767	0
	Velocity Feedforward Gain	Vff	0 to 32,767	0
1	Parameter (Kp) is the selector for the PID parameter to load.			
	Board ID is a unique number as	signed by Meas	surement & A	utomatio

Explorer (MAX) used to send and receive commands and data to or from

Axis is the axis for which to load PID parameters. Valid axis numbers are 1 through 15 for all NI motion controllers. On controllers with less than

eight axes, configuring non-existent axes has no effect.

This VI expects to be given the parameters in the following order: Axis, Parameter, Value.

Refer to the motion controller documentation for an overview of the enhanced PID control loop on the NI motion controller. The Load Single PID Parameter VI allows you to change an individual PID value on-the-fly without having to worry about the other unchanged PID values.

Example

To load a Kp of 1,000 to an axis 2, call the Load Single PID Parameter VI with the following parameters:

```
Axis = 2
Parameter = Kp
Value = 1,000
Inp Vect = 0xFF (Immediate)
```

Note If you are doing onboard programming and are using Inp Vect to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by Inp Vect in the following order: Parameter, Value.

Loading a Second Set of PID Parameters

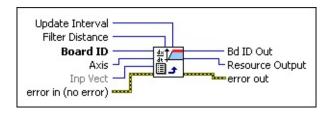
To load a second set of PID parameters, use the Load All PID Parameters and Load Single PID Parameter VIs to load the second set of PID values. Instead of using regular axis resource IDs, such as axis 1 = 0x1 (1), axis 2 = 0x2 (2), and so on, use the following alternate resource IDs: axis 1 = 0xA1 (161), axis 2 = 0xA2 (162).

Use the <u>Load Advanced Control Parameter</u> VI to enable the new set of PID parameters based on the condition specified in the **value** parameter.

Load Velocity Filter Parameter

Sets the velocity Filter Time Constant for the digital single-pole lowpass filter used to average the instantaneous axis velocity.

Device Compatibility



Vectors for more detailed information.

Details Remarks

Update Interval is the maximum velocity update period in ms. The range for this parameter is 1 to 2,500 ms.
Filter Distance is the number of position changes in counts/steps necessary for a velocity to be reported. The range for this parameter is from 0 to 1,000 counts/steps.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis to configure.
Inp Vect contains the distance and interval parameters that are the source of the data for this VI.
Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return

error in (no error) describes error conditions that occur before this VI

This VI configures the internal filter for velocity reporting. The reported velocity can oscillate due to feedback quantization error, noise, and jittery systems, rendering the data useless. You can apply a filter over multiple data samples to calculate an average velocity that reflects the actual system behavior. This filter is represented by **Filter Distance**. The larger the **Filter Distance**, the more position changes accumulate to perform velocity calculation. This is usually better for a noisy system.

A large **Filter Distance** can introduce a longer delay in velocity reporting as the position changes accumulate. To limit this delay and sluggish data reading, specify a minimum update interval using the **Update Interval** parameter. The axis velocity is updated within this update interval if the **Filter Distance** requirement is not met.

Note Velocity quantization noise is a measurement-only phenomenon and does not affect the ability of the motion controller to accurately control velocity and position during a move.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• For the **Inp Vect** parameter, the NI SoftMotion Controller supports only the immediate return vector (0XFF).

Set Stepper Loop Mode

Sets a stepper axis to operate in either open-loop, closed-loop, or P-Command mode.

Device Compatibility



Details

0	Loop Mode (Open) sets the mode for the stepper axis.
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis is the axis for which to load PID parameters. Valid axis numbers are 1 through 15 for all NI motion controllers. On controllers with less than eight axes, configuring non-existent axes has no effect.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:

status is FALSE, **code** is zero or a warning code. **code** is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

status is TRUE if an error occurred before this VI was called, or

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

You can operate stepper axes in both open and closed-loop modes. In open-loop mode, the stepper axis controls the trajectory profile and generates steps but has no feedback from the motor or actuator to determine if the profile is followed correctly.

In closed-loop mode, the feedback position is constantly compared to the number of steps generated to see if the stepper motor is moving correctly. When the trajectory profile is complete, missing steps (if any) are made up with a pull-in move. If, at any time during the move, the difference between the instantaneous commanded position and the feedback position exceeds the programmed following error limit, the axis is killed and motion stops.

You use P-Command mode for systems in which a servo drive receives step/dir (CW/CCW) from the motion controller. The drive closes the PID loop, and the motion controller generates the trajectory. Similar to closed-loop mode, in P-Command mode, the axis is killed when the programmed following error limit is exceeded.

Caution For proper closed-loop and P-Command operation, the correct values for steps/rev and counts/rev must be loaded with the <u>Load</u>
 <u>Counts/Steps per Revolution</u> VI. Incorrect counts to steps ratio can result in failure to reach the target position and erroneous closed-loop stepper operation.

To operate in closed-loop or P-Command mode, a stepper axis must have a primary feedback resource, such as an encoder, mapped to it prior to enabling the axis. Refer to the <u>Configure Axis Resources</u> VI for more information about feedback resources. You can operate an axis with a primary feedback resource in open-loop, closed-loop, or P-Command mode and you can switch the mode at any time. You can still read the position of the mapped feedback resource even when the axis is in open-loop mode.

Trajectory Control VIs

This VI group contains detailed descriptions of VIs used to set up and control motion trajectories on the NI motion controller. It includes VIs to load double-buffered trajectory parameters, read back instantaneous velocity, position and trajectory status, as well as VIs to configure blending, gearing, and other advanced trajectory features.

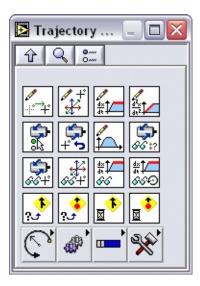
Double buffered parameters for axes and vector spaces include acceleration, deceleration, velocity, s-curve, operation mode, target position, and circular, helical, and spherical arc parameters. You can send these parameters to the controller at any time, but they do not take effect until you execute the next Start Motion or Blend Motion VI. This double buffering allows you to set up moves ahead of time, synchronizing them with a single Start Motion or Blend Motion call.

Other trajectory VIs allow you to configure the operation of trajectory generators and set status thresholds. These parameters include following error, blend factor, gear master, ratio and enable, position modulus, velocity threshold, torque limit, torque offset, and software limit positions. Unlike double-buffered parameters, if you change these parameters on the fly, they take effect immediately. Also in this category are VIs to reset position to zero or another specified value and to force a velocity override.

During a move, you can read the instantaneous values of position, velocity, following error, and DAC output (torque). There are also VIs to read the following trajectory status: move complete, profile complete, blend complete, motor off, following error trip, velocity threshold, and DAC limit status. These trajectory values and status are used for move sequencing, system coordination, and overall monitoring purposes.

Finally, NI-Motion offers a set of VIs to acquire time-sampled position and velocity data into a large onboard buffer and then later read it out for analysis and display. These VIs implement a digital oscilloscope that is useful during system set up, PID tuning, and general motion with data acquisition synchronization.

Click the icons for VI descriptions.



Load Target Position

Load Vector Space Position

Load Velocity

Load Acceleration/Deceleration

Set Operation Mode

Reset Position

Load Move Constraint

Read per Axis Status

Read Position

Read Vector Space Position

Read Velocity

Read Velocity in RPM

Check Blend Complete Status

Check Move Complete Status

Wait for Blend Complete

Wait for Move Complete

Subpalettes

Arc VIs

Gearing and Camming VIs

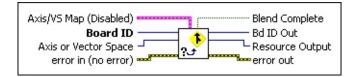
Buffered Operations VIs

Advanced Trajectory VIs

Check Blend Complete Status

Checks the blend complete status for an axis, vector space, group of axes, or group of vector spaces.

Device Compatibility



<u>Details</u>

Axis/VS Map is the bitmap of axes or vector spaces to check. It is only required when multiple axes or vector spaces are selected with the Axis or Vector Space parameter. Otherwise, this parameter is ignored.

When checking multiple axes (Axis/VS Map = Axis Control):

- **Axis 1**—If true, blend must be complete on the specified axis. If false, blend can be either complete or not complete on the specified axis (do not care).
- **Axis 2**—If true, blend must be complete on the specified axis. If false, blend can be either complete or not complete on the specified axis (do not care).
- **Axis** 3—If true, blend must be complete on the specified axis. If false, blend can be either complete or not complete on the specified axis (do not care).
- **Axis 4**—If true, blend must be complete on the specified axis. If false, blend can be either complete or not complete on the specified axis (do not care).

Axis 5—If true blend must be complete on the specified axis. If

This VI extends the functionality of the <u>Read Blend Status</u> VI, such that you can define a combination of axes or vector spaces to check as a group. It also can check the blend complete status for a single axis or vector space, much the same as Read Blend Status. Instead of decoding the output of the Read Blend Status VI yourself, this VI does that for you by comparing the axes or vector spaces specified in the **Axis or Vector Space** and **Axis/VS Map** input parameters with the blend complete status for the appropriate axes or vector spaces. The output is a single true/false value indicating whether or not the specified blend or blends are complete.

This VI does not check for following error status or axis off status. Checking for faults during moves is recommended. To do this, use <u>Read per Axis Status</u>, which returns following error and axis off status in addition to the move complete status. Refer to the <u>NI-Motion Help</u> for more information and example code.

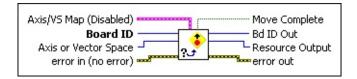
For more information about blend complete status, refer to the <u>Read Blend</u> Status VI.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Check Move Complete Status

Checks the move complete status for an axis, vector space, group of axes, or group of vector spaces.

Device Compatibility



Details

Axis/VS Map is the bitmap of axes or vector spaces to check. It is only required when multiple axes or vector spaces are selected with the Axis or Vector Space parameter. Otherwise, this parameter is ignored.

When checking multiple axes (Axis/VS Map = Axis Control):

- **Axis 1**—If true, move must be complete on the specified axis. If false, move can be either complete or not complete on the specified axis (do not care).
- **Axis 2**—If true, move must be complete on the specified axis. If false, move can be either complete or not complete on the specified axis (do not care).
- **Axis 3**—If true, move must be complete on the specified axis. If false, move can be either complete or not complete on the specified axis (do not care).
- **Axis 4**—If true, move must be complete on the specified axis. If false, move can be either complete or not complete on the specified axis (do not care).

Axis 5—If true, move must be complete on the specified axis. If

This VI extends the functionality of the <u>Read Trajectory Status</u> VI, such that you can define a combination of axes or vector spaces to check as a group. It also can check the move complete status for a single axis or vector space, much the same as Read Blend Status. Instead of decoding the output of the Read Trajectory Status VI yourself, this VI does that for you by comparing the axes or vector spaces specified in the **Axis or Vector Space** and **Axis or VS Map** input parameters with the move complete status for the appropriate axes or vector spaces. The output is a single true/false value indicating whether or not the specified move or moves are complete.

This VI does not check for following error status or axis off status. Checking for faults during moves is recommended. To do this, use <u>Read per Axis Status</u>, which returns following error and axis off status in addition to the move complete status. Refer to the <u>NI-Motion Help</u> for more information and example code.

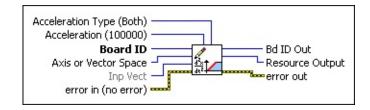
For more information about move complete status, refer to the <u>Read Trajectory</u> <u>Status</u> and <u>Configure Move Complete Criteria</u> VIs.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Load Acceleration/Deceleration

Loads the maximum acceleration and/or deceleration value for an axis or vector space.

Device Compatibility



<u>Details</u> <u>Remarks</u>

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis or Vector Space is the axis or vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE,

The Load Acceleration VI specifies the maximum rate of acceleration and/or deceleration for individual axes or vector spaces. When executed on a vector space, the value controls the vector acceleration (deceleration) along the vector move path.

You can use this VI to load separate limits for acceleration and deceleration or to set them both to the same value with one call. These parameters are double-buffered so you can load them on the fly without affecting the move in process, and they take effect on the next <u>Start</u> or <u>Blend Motion</u> call. After they are loaded, these parameters remain in effect for all subsequent motion profiles until re-loaded by this VI. You do not need to load acceleration before each move unless you want to change the acceleration and/or deceleration.

Acceleration defines how quickly the axis or axes come up to speed and is typically limited to avoid excessive stress on the motor, mechanical system, and/or load. A separate, slower deceleration is useful in applications where coming to a gentle stop is important.

Note Use <u>Load Move Constraints</u> to load an acceleration of less than 1 count/s².

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

 Acceleration and deceleration values are converted to an internal 16.16 fixed-point format in units of counts/s² before being used by the trajectory generator. You can calculate the minimum acceleration increment with the following formula:

minimum acceleration/deceleration = Amin $\times (\frac{1}{T_s})^2$

where: *Amin* is 1/65,536 counts/sample² or steps/sample², *Ts* is the sample period in seconds per sample.

• For a typical servo axis with 2,000 counts per revolution operating at the 250 ms update rate, calculate the minimum acceleration/deceleration increment using the following equation:

$$\left(\frac{1}{65,536}\right) \times \left(\frac{4,000^2}{2,000}\right) = 0.122070 \text{ counts/s}$$

• You can calculate the maximum acceleration/deceleration using the following equation:

maximum acceleration/deceleration = Amax $\times (\frac{1}{T_5})^2$

where: *Amax* is 32 counts/sample, *Ts* is the sample period in seconds per sample, and is constrained according to the following equations:

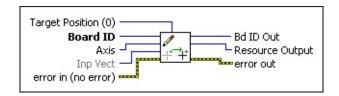
acceleration ≤ 256 × deceleration deceleration ≤ 65536 × acceleration

• If you are doing onboard programming and are using **Inp Vect** to get the

Load Target Position

Loads the target position for the next axis move.

Device Compatibility



Details

Target Position (0) is the target position for the next axis move in counts (servo axes) or steps (stepper axes). Target positions can be anywhere within the 32-bit position range, $-(2^{31})$ to $+((2^{31})-1)$. The default value for target position is zero (0).

Caution Any single move is limited to $\pm (2^{31}-1)$ counts or steps. An error is generated if you exceed this limit by loading a target position too far from the current Axis position.

Board ID is a unique number assigned by Measurement & AutomationExplorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis is the axis to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs permally only if no incoming error exists. Otherwise, the VI passes the

The Load Target Position VI loads a target position to the specified axis. Target positions can be for single axis moves, multi-axis coordinated moves, or vector space moves. Position values indicate the end location and direction of motion (target position).



Note Refer to the <u>Load Vector Space Position</u> VI for an easy way to load up to three target positions for a vector space in one call.

Target position is double-buffered so you can load it on the fly without affecting the move in process, and it takes effect on the next Start or Blend Motion VI. When the target position is loaded, it is interpreted as either an absolute target position, a relative target position, a target position relative to the last captured position or with the effect of a position modulus, depending on the mode set with the <u>Set Operation Mode</u> VI.

This means that if you set the operation mode to "relative to capture," you must load the target position after the capture has occurred because the position is evaluated on load.

After you execute the start or blend, the axis or axes follow the programmed trajectory and end up at the absolute, relative, or modulo target position.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Load Vector Space Position

Loads the axis target positions for the next vector space move.

Z Position (0)

Board ID

Vector Space
Inp Vect
error in (no error)
X Position (0)
Y Position (0)

Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Vector Space is the vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRIJE

The Load Vector Space Position VI loads up to three axis target positions for the vector space specified. This VI is identical to calling the Load Target <u>Position</u> VI up to three times, one time per each axis in the vector space. Position values indicate the end location and direction of motion (target position).

Target positions are double-buffered so you can load them on the fly without affecting the move in process, and they take effect on the next Start or Blend Motion VI. When the target positions are loaded, they are interpreted as either absolute target positions, relative target positions, target positions relative to the last captured positions or with the effect of a position modulus, depending on the mode set with the **Set Operation Mode** VI.

This means that if you set the operation mode to "relative to capture," you must load the target position after the capture has occurred because the position is evaluated on load.

After you execute the start or blend, the axes in the vector space follow the programmed trajectory and end up at the absolute, relative, or modulo target positions.



Note If the vector space contains less than three axes, the extra target position values are ignored.

Note If you are doing onboard programming and are using Inp Vect to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by Inp Vect in the following order: X Position, Y Position, Z Position.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Load Velocity

Loads the maximum velocity for an axis or vector space.

Velocity (10000)

Board ID

Axis or Vector Space
Inp Vect
error in (no error)

Bd ID Out
Resource Output
error out

Device Compatibility

Details Remarks

parameters:

132	Velocity is the target or maximum velocity in counts/s (servo axes) or steps/s (stepper axes). The default is 10000.
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis or Vector Space is the axis or vector space to control.
	Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

The Load Velocity VI specifies the maximum trajectory velocity for individual axes or vector spaces. When executed on a vector space, the value controls the vector velocity along the vector move path. For velocity control applications, the sign of the loaded velocity specifies the move direction.

Velocity is a double-buffered parameter so you can load it on the fly without affecting the move in process, and it takes effect on the next Start or Blend Motion call. After it is loaded, this parameter remains in effect for all subsequent motion profiles until re-loaded by this VI. You do not need to load velocity before each move unless you want to change the velocity.



Note Use <u>Load Move Constraints</u> to load a velocity of less than 1 count/sec.

Note The velocity loaded with this VI is the maximum move velocity. Actual velocity attainable is determined by many factors including PID tuning, length of move, acceleration and deceleration values, and physical constraints of the amplifier/motor/mechanical system.



Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

• Velocity values in counts/s or steps/s are also converted to the internal 16.16 fixed-point format in units of counts or steps per sample (update) period before being used by the trajectory generator. Although the motion controller can control velocity to 1/65,536 of a count or step per sample, it is impossible to load a value that small with the Load Velocity VI, as shown in the following formula:

Velocity in counts or step/s = Vmin × $(\frac{1}{T_s})$

where:

Vmin is 1/65,536 counts/sample or steps/sample, and *Ts* is the sample period in seconds per sample.

Even at the fastest update rate,

Ts =
$$625 \times 10^{-6}$$

 $\left(\frac{1}{65,536}\right) \times 16,000 = 0.244$ counts or steps/s

• You can calculate the maximum velocity with the following equation:

maximum velocity = Vmax

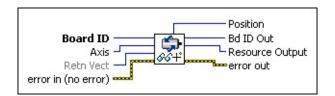
where:

Vmax = 20 MHz for servos,

8 MHz for steppers on a 7350 controller,

Read Position

Reads the position of an axis.



Device Compatibility

Details

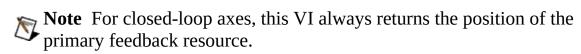
parameters:

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis to read.
Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

The Read Position VI returns the instantaneous position of the specified axis. For servo axes, it returns the primary feedback position in counts. For open-loop stepper axes, it returns the number of steps generated. For closed-loop stepper axes, it converts the primary feedback position from counts to steps and then returns the value in steps. Closed-loop stepper axes require you to correctly load values of steps per revolution and counts per revolution to function correctly.



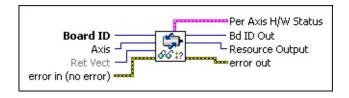
Tip Refer to the <u>Read Vector Space Position</u> VI for an easy way to read up to three axis positions for a vector space in one call.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Read per Axis Status

Reads the motion status on a per-axis basis.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis is the axis to read.

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

The Read Per Axis Status VI returns the trajectory and motion I/O status for the specified axis. It also returns the success or failure status of the most recent <u>Find Reference</u> sequence.



Note You also can read individual item status in a multi-axis format with NI-Motion VIs like <u>Read Limit Status</u>, <u>Read Trajectory Status</u>, and so on.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

• In the **Per Axis H/W Status** bitmap, D7 Communication Watchdog (WD) is not supported by 73*xx* controllers.

NI SoftMotion Controller Considerations

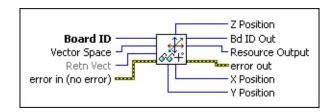
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- When you are using the NI SoftMotion Controller with a device that supports multiple high-speed captures and breakpoints, the status for primary and secondary inputs is concatenated by a logical OR operation to populate **Per Axis H/W Status**.
- In the Per Axis H/W Status bitmap, Watchdog is used for a communication status watchdog on the NI SoftMotion Controller.

Read Vector Space Position

Reads the position of all axes in a vector space.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Vector Space is the vector space to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

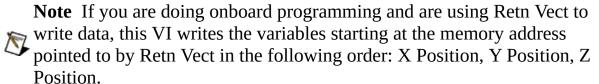
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

The Read Vector Space Position VI returns the instantaneous positions of the axes in the specified vector space. For servo axes, it returns the primary feedback position in counts. For open-loop stepper axes, it returns the number of steps generated. For closed-loop stepper axes, it converts the primary feedback position from counts to steps and then returns the value in steps. Closed-loop stepper axes require correctly loaded values of steps per revolution and counts per revolution to function correctly.



Note For closed-loop axes, this VI always returns the position of the primary feedback resource.



Read Velocity

Reads the filtered velocity of an axis or vector space.

Filtered Velocity Board ID -Bd ID Out Axis or Vector Space -Resource Output Retn Vect error in (no error) ----

Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Axis or Vactor Space is the axis or vactor space to read

Axis or Vector Space is the axis or vector space to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code. Use the error handler VIs to look up the meaning of

This VI returns the axis or vector space filtered velocity. To reduce the effects of noise and jitter on the filter calculation, the data this VI returns is averaged based on parameters set in the Load Velocity Filter Parameter VI.

For vector spaces, this VI returns vector velocity, the root-mean-square of the filtered velocities of the individual axes that make up the vector space.



Note You also can read velocity in RPM by calling the Read Velocity in RPM VI.

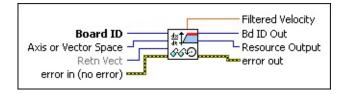


Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Read Velocity in RPM

Reads the filtered velocity of an axis or vector space in RPM.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis or Vector Space is the axis or vector space to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

This VI returns the axis or vector space filtered velocity in RPM. The filtered velocity is measured based on time elapsed between position changes. To avoid quantization error and jitter in the mechanical system, a filtering algorithm in <u>Load Velocity Filter Parameter</u> is performed with parameters configured.

For vector spaces, this VI returns vector velocity, the root-mean-square of the filtered velocities of the individual axes that make up the vector space.



Note This VI requires previously loaded values of either counts per revolution (for servo axes) or steps per revolution (for stepper axes) to operate correctly.



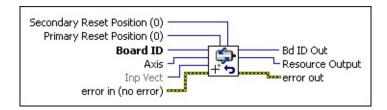
Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Reset Position

Resets the axis position to the specified position, taking following error into account.

Note If you are using this VI with an NI 7350 or NI 7390 device that you are reading positions on, it might take up to 5 ms for the application to display the most recently read position after you call this VI.

Device Compatibility



Details Remarks

I32

Primary Reset Position (0) is the reset value for the axis and its associated primary feedback resource, taking following error into account. You can reset position to any value in the total position range of $-(2^{31})$ to $+(2^{31}-1)$. The value is in steps for stepper systems and counts for servo systems.

Secondary Reset Position (0) is the reset value for the optional secondary feedback resource. You can reset position to any value in the total position range of $-(2^{31})$ to $+(2^{31}-1)$.

Note For stepper closed-loop configurations, where the encoder counts per revolution is greater than the steps per revolution, the range of the position parameters is reduced by the ratio of counts/rev to steps/rev. For example, if the ratio is 4,000 counts per revolution to 2,000 steps per revolution (2:1 ratio), then the range of position

This function resets the axis position and the associated primary feedback position, taking following error into account, and resets the optional secondary feedback position. This means that if you reset the position to 0 and have a following error of 15, the feedback position is −15.

You can reset position to zero or any value in the 32-bit position range. You can reset the secondary feedback to the same value as the axis and primary feedback or you can reset it to a different value. If a secondary feedback resource is not in use, the corresponding reset value is ignored. If you use the same feedback device for both primary and secondary feedback, you must reset the positions to be the same value.



Note Non-zero reset values are useful for defining a position reference offset.

Position can be reset at any time. However, it is recommended that you reset position only while the axis is stopped. An axis reset while the axis is moving does not have a repeatable reference position.



Note For axes in p-command mode, ensure that the axis is not moving N before calling Reset Position. Calling Reset Position when the axis is moving introduces position error into the system.

Typically, you execute the Reset Position VI one time after the Find Home and Find Index sequences of the **Find Reference** VI have completed successfully to set the zero position for the system and not again until the next power-up.

An ADC channel used as a primary feedback resources is reset by storing an offset value when this VI is executed. In this way, its zero reference is not lost and you can still read the actual ADC value with the Read ADCs VI.

Note If you are doing onboard programming and are using **Inp Vect** to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by Inp Vest in the following orders Drim

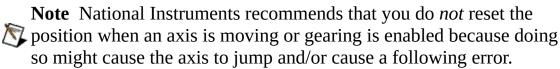
Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

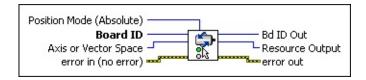
- The secondary reset position is ignored.
- When you reset a slave position, gearing is not disabled.



Set Operation Mode

Sets the operation mode for an axis or vector space.

Device Compatibility



Details Remarks

Position Mode (Absolute) selects the type of position or velocity mode for an axis or vector space.

Value Definition

- 0 Absolute Position
- 1 Relative Position
- 2 Velocity
 - 3 Relative to Capture
 - 4 Modulus Position
 - 5 Absolute Contouring
 - 6 Relative Contouring

These modes are described in the <u>Using This VI</u> section.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis or Vector Space is the axis or vector space to configure.

The Set Operation Mode VI is used both during initialization and during normal motion control operation to configure the mode of operation for all trajectory commands to the axis or vector space specified.

Position modes are applied to all axes in a coordinate (vector) space. If you later want to operate an axis independently in a different mode from the other axes, call Set Operation Mode again on that axis.



Note All axes in a coordinate (vector) space must have the same operation Mode. If the operation modes are different on each Axis when a Start Motion or Blend Motion VI is executed, an error is generated.

The operation mode must be set or changed before any other trajectory parameters are loaded for the next move. The operation mode affects how the target position and velocity values are interpreted. Trajectory parameters loaded after a mode change are interpreted in the newly selected mode. Trajectory parameters loaded any time before a mode change do not reflect the new mode.



Note Changing operation mode after the trajectory parameters are loaded can result in improper operation.

The five operation modes are described in the following sections:

Absolute Position

In absolute position mode, target positions are interpreted with respect to an origin, reference, or zero position. The origin is typically set at a home switch, end of travel limit switch, or encoder index position. An absolute position move uses the preprogrammed values of acceleration, deceleration, s-curve, and velocity to complete a trajectory profile with an ending position equal to the loaded absolute target position.

The length of an absolute move depends upon the loaded position and the current position when the move is started. If the target position is the same as

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- NI 7330 and NI 7390 controllers do not support absolute or relative contouring.
- NI 73xx controllers do not support the **Velocity** option of the **Position Mode** parameter for vector spaces.

NI SoftMotion Controller Considerations

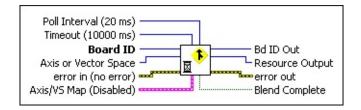
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• The NI SoftMotion Controller does not support the **Modulus Position** option for **Position Mode**.

Wait for Blend Complete

Waits up to the specified period of time for a blend to be completed on an axis, vector space, group of axes, or group of vector spaces.

Device Compatibility



Details

Poll Interval (20 ms) is the amount of time, in milliseconds, to wait between successive queries to the controller to determine if the blend is complete.
Timeout (10,000 ms) is the amount of time, in milliseconds, to wait for the blend to become complete.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis or Vector Space can select an axis (1 through 15), vector space (0x11 through 0x14), multiple axes (Axis Control), or multiple vector spaces (Vector Space Control). When simultaneously acting on multiple axes or vector spaces, the Axis/VS Map parameter indicates which axes or vector spaces are involved.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs

This utility VI is built on top of the <u>Check Blend Complete Status</u> VI, and is provided for your programming convenience. This VI compares the axes or vector spaces specified in the **Axis or Vector Space** and **Axis/VS Map** input parameters with the blend complete status for the appropriate axes or vector spaces. It does this repeatedly, with the **Poll Interval** time determining the frequency that the controller is queried. When the blend is complete, this VI returns with Blend Complete set to True, otherwise it waits for the amount of time specified by the **Timeout** parameter. If the VI waits longer than the specified timeout, it returns NIMC_eventTimeoutError and **Blend Complete** is set to false.

The output is a single true/false value indicating whether or not the specified blend or blends are complete.

This VI does not check for following error status or axis off status. Checking for faults during moves is recommended. To do this, use Read per Axis Status, which returns following error and axis off status in addition to the move complete status. Refer to the NI-Motion Help for more information and example code.

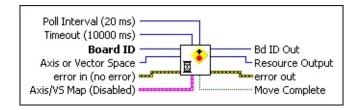
For more information about blend complete status, refer to the <u>Read Blend</u> Status VI.

Note to LabVIEW RT users If this VI is used with LabVIEW RT in a time-critical priority thread, the entire thread is put to sleep until the next multiple of Poll Interval. For more information refer to the LabVIEW RT documentation

Wait for Move Complete

Waits up to the specified period of time for a move to be completed on an axis, vector space, group of axes, or group of vector spaces.

Device Compatibility



Details

Poll Interval (20 ms) is the amount of time, in milliseconds, to wait between successive queries to the controller to determine if the blend is complete.
Timeout (10,000 ms) is the amount of time, in milliseconds, to wait for the blend to become complete.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis or Vector Space can select an axis (1 through 15), vector space (0x11 through 0x14), multiple axes (Axis Control), or multiple vector spaces (Vector Space Control). When simultaneously acting on multiple axes or vector spaces, the Axis/VS Map parameter indicates which axes or vector spaces are involved.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs

This utility VI is built on top of the <u>Check Move Complete Status</u> VI, and is provided for your programming convenience. This VI compares the axes or vector spaces specified in the **Axis or Vector Space** and **Axis/VS Map** input parameters with the move complete status for the appropriate axes or vector spaces. It does this repeatedly, with the **Poll Interval** time determining the frequency that the controller is queried. When the move is complete, this VI returns with **Move Complete** set to True, otherwise it waits for the amount of time specified by the **Timeout** parameter. If the VI waits longer than the specified timeout, it returns NIMC_eventTimeoutError and **Move Complete** is set to false.

The output is a single true/false value indicating whether or not the specified move or moves are complete.

This VI does not check for following error status or axis off status. Checking for faults during moves is recommended. To do this, use Read per Axis Status, which returns following error and axis off status in addition to the move complete status. Refer to the NI-Motion Help for more information and example code.

For more information about move complete status, refer to the <u>Read Trajectory</u> Status and Configure Move Complete Criteria VIs.

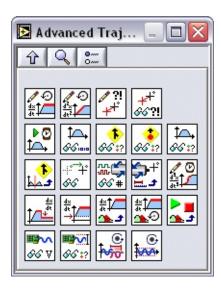
Note to LabVIEW RT users If this VI is used with LabVIEW RT in a time-critical priority thread, the entire thread is put to sleep until the next multiple of Poll Interval. For more information refer to the LabVIEW RT documentation

Advanced Trajectory VIs

Advanced Trajectory VIs are useful in special applications and showcase some of NI-Motion's power and flexibility.

You can use Advanced Trajectory VIs to acquire time-sampled position and velocity data into a large onboard buffer and then later read it out for analysis and display. These VIs implement a **digital oscilloscope** that is useful during system set up, PID tuning, and general motion with data acquisition synchronization.

Click the icons for VI descriptions.



Load Velocity in RPM

Load Accel/Decel in RPS/sec

Load Following Error

Read Following Error

Acquire Trajectory Data

Read Trajectory Data

Read Blend Status

Read Move Complete Status

Read Trajectory Status

Load Position Modulus

Load S-Curve Time

Load Velocity Override

Load Base Velocity

Load Velocity Threshold

Load Velocity Threshold in RPM

Load Run/Stop Threshold

Read DAC

Read DAC Limit Status

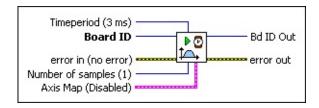
Load Blend Factor
Read Target Position
Read Steps Generated

Load Torque Offset Load Torque Limit

Acquire Trajectory Data

Acquires time-sampled position and velocity data on multiple axes.

Device Compatibility



Timeperiod (3 ms) is the time period between samples in ms. The range is from 3 (default) to 65,535 ms.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

The Acquire Trajectory Data VI initiates the automatic acquisition of position and velocity data for the selected axes. The data is held in an onboard first-infirst-out (FIFO) buffer until later read back with the <u>Read Trajectory Data</u> VI. You can select which axes to acquire data for and program the time period between samples.

The Acquire and Read Trajectory Data VIs are used to acquire and read back time-sampled position and velocity data for analysis and display. These VIs implement a digital oscilloscope that is useful during system set up, PID tuning, and general motion with data acquisition synchronization.

After it is started, this data acquisition operates autonomously in the background as a separate task. Motion control operates normally and you can execute other motion VIs simultaneously. Depending upon the programmed time period and the total number of samples, this acquisition task can run anywhere from a few milliseconds to tens of hours.

Caution Wait an appropriate amount of time before attempting to read back the trajectory data.

Because host communication, event handling, arc point generation, contour point dispensing, buffered breakpoints, and buffered high-speed capture are a higher priority on the controller than acquiring trajectory data, you may notice irregular time spacing of sampled data when host communications or event notification is heavy. To minimize the chance of irregular data, limit host communication to a minimum while acquiring trajectory data. If you still notice irregular data, increase the time period.

Example

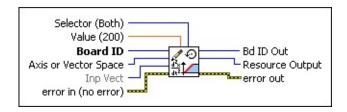
To acquire 100 samples of data on axes 1 and 2 at 10 ms/sample, call the Acquire Trajectory Data VI with the following parameters:

Axis Map corresponds to the following

Load Accel/Decel in RPS/sec

Loads the maximum acceleration and/or deceleration value for an axis or vector space in RPS/s.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

The Load Accel/Decel in RPS/sec VI specifies the maximum rate of acceleration and/or deceleration for individual axes or vector spaces in revolutions/s². When executed on a vector space, the value controls the vector acceleration (deceleration) along the vector move path. This VI requires previously loaded values of either counts per revolution (for stepper axes) or steps per revolution (for servo axes) to operate correctly.

You can use this VI to load separate limits for acceleration and deceleration or to set them both to the same value with one call. These parameters are double-buffered so you can load them on the fly without affecting the move in process, and they take effect on the next <u>Start</u> or <u>Blend Motion</u> call. After they are loaded, these parameters remain in effect for all subsequent motion profiles until re-loaded by this VI. You do not need to load acceleration before each move unless you want to change the acceleration and/or deceleration value.

Acceleration defines how quickly the axis or axes come up to speed and is typically limited to avoid excessive stress on the motor, mechanical system, and/or load. A separate, slower deceleration is useful in applications where gently coming to a stop is most important.

Tip Use <u>Load Move Constraints</u> instead of this VI for more move options.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- RPS/s values stored in onboard variables are in double-precision IEEE format (f64). For information about the number of variables required to hold an RPS/s value, refer to <u>Using Inputs and Return Vectors with</u> Onboard Variables.
- NI-Motion firmware uses a 16.16 fixed-point format internally to represent velocity in counts/sample period or steps/sample period and acceleration/deceleration in counts/sample period² or steps/sample period². If you change counts or steps per revolution or change the update rate using <u>Enable Axes</u>, you must reload the velocity and acceleration/deceleration.
- You also can load acceleration and deceleration in counts/s² or steps/s² by calling the <u>Load Acceleration/Deceleration</u> VI.
- If you are doing onboard programming and are using inputVector to get
 the data this VI needs, notice that this VI reads the variables starting at the
 memory address pointed to by inputVector in the following order:
 accelerationType, RPSPS.
- Acceleration and deceleration values in RPS/s are converted to an internal 16.16 fixed-point format in units of counts/sample² or steps/sample² before being used by the trajectory generator. You can calculate the minimum acceleration increment in RPS/s with the following formula:

RPS/s = Amin
$$\times (\frac{1}{T_s})^2 \times \frac{1}{R}$$

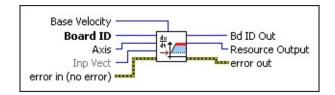
where:

Amin is 1/65.536 counts/sample² or steps/sample².

Load Base Velocity

Sets the base velocity used by the trajectory control loop for the axis specified.

Device Compatibility



Details

Base Velocity is the base velocity for the stepper axis in steps/second and is a 16-bit data word in the range of 0 through 65,535. The default value is 0.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis is the axis to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

Base velocity is the minimum step rate used by the trajectory generator during acceleration and deceleration. Base velocity is useful when the system uses full-step or half-step mode. Base velocity is not necessary if the system uses microstepping.

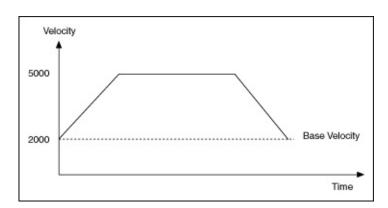
If the target velocity loaded with the Load Velocity VI is lower than the base velocity, the base velocity is reduced to equal the loaded target velocity.



Note This VI is valid only on axes configured as steppers, so you must Configure an axis as a stepper using the Configure Axis Resources VI before executing this VI.

Example

If the base velocity loaded is 2,000 steps/s and the loaded velocity is 5000, the axis starts at the base velocity and accelerates to the loaded velocity. The axis then decelerates to the base velocity and stops, as shown in the following figure.



Load Blend Factor

Loads the blend factor for an axis or vector space.

Blend Factor (-1)

Board ID

Axis or Vector Space
Inp Vect
error in (no error)

Bd ID Out
Resource Output
error out

Device Compatibility

Details

Blend Factor is the blend factor mode and/or dwell time. –1 specifies normal blending (default), 0 specifies a start after the previous move is fully stopped, and values > 0 specify additional dwell time in milliseconds.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis or Vector Space is the axis or vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

The Load Blend Factor VI controls how the <u>Blend Motion</u> VI operates. Blending automatically starts a pending move on an axis or vector space when the move in process completes. Exactly when the pending move starts is determined by the loaded blend factor.

A blend factor of -1 causes the pending move to start when the existing move finishes its constant velocity segment and starts to decelerate, as shown in the following figure. This blends the two moves together at the optimum blend point.

If the two moves are at the same velocity, in the same direction, and have matching acceleration and deceleration, they superimpose perfectly without a dip or increase in axis velocity. Vector velocity dips based on the amount of deflection from a straight line.

For a vector move, if all of the axes continue in the same direction, the vector velocity remains constant. But, if one of the axes changes direction, the vector velocity does not remain constant during the transition phase.

A blend factor of zero (0) causes the pending move to start when the existing move fully completes its profile.

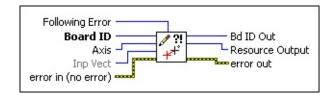
Positive blend factors allow for a dwell at the end of the first move before the automatic start of the pending move. The blend factor dwell is programmed in milliseconds.

The maximum value of the positive blend factor depends upon the Sample PID Rate that you set in the <u>Enable Axes</u> VI, because the DSP delays the trajectory generators based on PID sample periods. The formula used to determine the maximum positive blend factor is as follows:

Load Following Error

Loads the following error trip point.

Device Compatibility



Details Remarks

parameters:

Following Error is the following error trip point in counts. If the following error exceeds this value, the axis is automatically killed. The range is 0 to 32,767 with a default of 32,767 counts. Loading zero (0) is a special case that disables the following error trip functionality. **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller. **Axis** is the axis to control. **Inp Vect** indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information. **error in (no error)** describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

The Load Following Error VI sets the maximum allowable following error. Following error is the difference between the instantaneous commanded trajectory position and the feedback position. If the absolute value of this difference exceeds the trip point, an internal kill stop is issued and the axis is disabled.

If this axis is part of a coordinate (vector) space, all other axes are commanded to decelerate to a stop when one axis trips on following error.

This VI is a safety feature used to protect the motion hardware and associated system components from damage when the position error gets excessive due to friction, binding, or a completely stalled motor. It also protects the system in case you load unobtainable values for velocity and/or acceleration.

This feature is available on all servo and closed-loop stepper axes. It has no effect on stepper axes running in open-loop mode. You can completely disable the following error feature by loading a zero (0) value.

Caution Following error must not be disabled unless the application absolutely requires operating with greater than 32,767 counts of error.

You can monitor following error status with the <u>Read Trajectory Status</u> or <u>Read per Axis Status</u> VIs. A following error trip always sets the Motor Off status. In servo systems, you can further diagnose the cause of the trip by checking the torque limit status with the <u>Read DAC Limit Status</u> VI.

If an axis trips on following error, that axis, or any axes associated through a coordinate space, are prohibited from starting from an onboard program until a start from the host, pause of the onboard program, or run of an onboard program.

In general, a following error trip is considered normal operation and does not generate an error. There are a few cases where an unexpected following error trip generates a modal error: during <u>Find Reference</u>, and while executing a

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

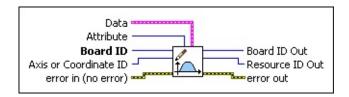
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The NI SoftMotion Controller does not support loading zero (0) in **Following Error** to disable the following error.
- For the **Inp Vect** parameter, the NI SoftMotion Controller supports only the immediate return vector (0XFF).

Load Move Constraints

Loads move constraints in user units.

Device Compatibility



Details Remarks

Data is a cluster	of variables t	that contains	the values	for the	selected
Attribute.					

Integer is 32-bit integer data.

Double is 64-bit floating point data.

Boolean is Boolean data.

Attribute is the attribute you want to load:

- **Velocity Constraint**—Set the **Double** element of the **Data** cluster to the appropriate velocity limit in user units/second.
- **Acceleration Constraint**—Set the **Double** element of the **Data** cluster to the appropriate acceleration limit in user units/second².
- **Deceleration Constraint**—Set the **Double** element of the **Data** cluster to the appropriate deceleration limit in user units/second².
- **Acceleration Jerk**—Set the **Double** element of the **Data** cluster to the appropriate acceleration jerk limit in user units/second³.
- **Deceleration Jerk**—Set the **Double** element of the **Data** cluster to the appropriate deceleration jerk limit in user units/second³.

Board ID is a unique number assigned by Measurement & Automation Explorer used to send and receive commands and data to or from a

This VI loads move constraints in user units using the **Double** parameter of the **Data** cluster. Use <u>Load Counts/Steps per Revolution</u> to set the user unit ratio.

Example 1

If you load 2,000 counts/rev using the <u>Load Counts/Steps per Revolution</u> VI, and you call Load Move Constraints with **Velocity Constraint** set to 10.5, the velocity loaded is 10.5 revolutions/second or 21,000 counts/second.

Example 2

If you set **Counts(Steps)/Rev** to 1 in the <u>Load Counts/Steps per Revolution</u> VI, and you call Load Move Constraints with **Velocity Constraint** set to 1.3, the velocity loaded is 1.3 counts/second.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

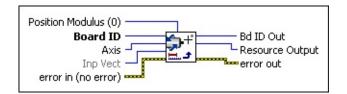
The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- 73xx controllers do not support the **Acceleration Jerk** and **Deceleration Jerk** attributes.
- Use <u>Load Velocity</u> and <u>Load Acceleration/Deceleration</u> for faster performance.

Load Position Modulus

Loads the position modulus for an axis.

Device Compatibility



Details

Position Modulus (0) is the position modulus value in counts (servo axes)
or steps (stepper axes). The modulus range is from 0 (default) to $(2^{31}) - 1$.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis to control.
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:
parameters.

status is TRUE if an error occurred before this VI was called, or

code is a number identifying an error or warning. If status is TDIJE

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

status is FALSE, **code** is zero or a warning code.

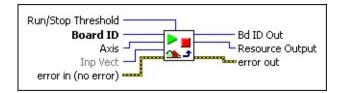
The Load Position Modulus VI sets the modulus used when the axis is operating in Modulus Position mode. It has no effect when the axis is operating in other modes. When a target position is loaded, it is interpreted within the boundaries of a modulus range.

Refer to the <u>Set Operation Mode</u> VI for a complete description of the Modulus Position mode.

Load Run/Stop Threshold

Sets the velocity Run/Stop Threshold, which affects the run/stop status returned by Read Trajectory Status.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Run/Stop Threshold is the Run/Stop threshold velocity in counts/sample period for servo axes or steps/sample period for open-loop and closed-loop stepper axes. The range for this parameter is 1 (default) to 32,767 counts or steps/sample period. This parameter sets the number of counts/steps an axis must move to be considered running.
Axis is the axis to configure.
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs

normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

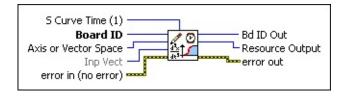
This VI sets the velocity threshold above which an axis is considered running. The **Run/Stop Threshold** parameter has time units of update sample periods and is affected by the update rate set in the <u>Enable Axes</u> VI.

If the velocity of an axis is above the step/count per sample period specified in this VI, the run/stop status from Read Trajectory Status returns TRUE. If it is below the threshold, the run/stop status returns FALSE.

Load S-Curve Time

Loads the s-curve time for an axis or vector space.

Device Compatibility



Details Remarks

acceleration profile is smoothed as it transitions from zero to the programmed value and back to zero. The s-curve range is from 1 to 32,767 with a default of 1 sample period.
 Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
 Axis or Vector Space is the axis or vector space to control.
 Inp Vect indicates the source of the data for this VI. Available input

S Curve Time (1) is the time in update sample periods over which the

vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

The Load S-Curve Time VI smoothes the acceleration and deceleration portions of a motion profile, resulting in less abrupt transitions from start motion to acceleration, acceleration to constant velocity, constant velocity to deceleration, and deceleration to stop. Using s-curve acceleration limits the jerk in a motion control system.

Officially, jerk is defined as the derivative of acceleration (change of acceleration per unit time) and is measured in units of counts (steps)/s³. This VI, however, allows you to load s-curve time in update sample periods rather than have to deal with the obscure units of jerk.

With the default s-curve time of one (1) sample period, there is almost no effect on the motion profile, and the standard trapezoidal trajectory is executed. As scurve time increases, the smoothing effect on the acceleration and deceleration portions of the motion profile increase, as shown in the following figure. Large values of s-curve time can override the programmed values of acceleration and deceleration by sufficiently smoothing the profile such that the acceleration and deceleration slopes are never reached.

Note A large s-curve value makes the time to accelerate to the commanded velocity longer. Therefore, the time it takes to reach the target position is also longer. If the s-curve value is too large, the value is ignored and full acceleration is used. This happens when the acceleration and deceleration become too long to perform smoothly.

You can load s-curve time to individual axes or to a vector space for smoothing all axes in the vector space. When sent to a vector space, the s-curve time is broadcast to all axes in the vector space to change the per-axis s-curve times. If you later want to operate an axis independently with a different s-curve time, you must execute the Load S-Curve Time VI again for that axis.

Note All axes in a vector space must have the same s-curve time for best

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- When you use this VI with the NI SoftMotion Controller, **S Curve Time** (1) is the smoothing time in the host loop period.
- National Instruments suggests you use the <u>Load Move Constraints</u> VI with the NI SoftMotion Controller instead of using this VI.

Load Torque Limit

Loads primary and secondary DAC torque limits for an axis.

Secondary (-) Limit
Secondary (+) Limit
Board ID
Axis
Inp Vect
error in (no error)
Primary (+) Limit
Primary (-) Limit

Device Compatibility

Details

Secondary (–) Limit is the optional secondary DAC negative torque (or velocity) limit. The range is -32,768 to +32,767 (-10 V to +10 V) with a default value of -32,768 (-10 V).

Note The positive limit cannot be less than the negative limit.

Secondary (+) Limit is the optional secondary DAC positive torque (or velocity) limit. The range is –32,768 to + 32,767 (–10 V to +10 V) with a default value of 32,767 (+10 V).

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis is the axis to control.

Inp Vect contains the primary positive, primary negative, secondary positive, and secondary negative parameters that are the source of data for this VI.

Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return

The Load Torque Limit VI allows you to limit the output range of the DAC output(s) on the selected servo axis. This VI has no effect on stepper axes or independent DAC outputs that are not mapped to an axis.

By limiting the output range of a DAC, it is possible to control the maximum torque (when connected to a torque block servo amplifier) or velocity (when connected to a velocity block servo amplifier). This VI is also helpful when interfacing to amplifiers that do not support the standard ±10 V command range.

Primary and secondary DACs can have different limits, and the positive and negative limits can be both positive or both negative to limit the DAC output to a unipolar range. The only restriction is that a positive DAC limit cannot be less than the negative DAC limit.

You also can set a torque offset on the primary and secondary DAC outputs. Refer to the <u>Load Torque Offset</u> VI for more information.

Example

Calling the Load Torque Limit VI with the following parameters limits the output ranges of the primary and secondary DACs mapped to the axis, as shown in the following figure:

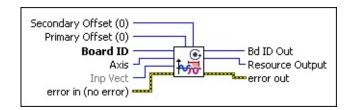
Primary Positive Limit = 16,383 Primary Negative Limit = -16,384 Secondary Positive Limit = 8,191 Secondary Negative Limit = -32,768

The result of this VI call is to limit the primary DAC to only half its range in either direction, or ±5 V. The secondary DAC can only travel over a quarter of its positive range but has its full negative range.

Load Torque Offset

Loads primary and secondary DAC torque offsets for an axis.

Device Compatibility



Details

Secondary Offset (0) is the secondary DAC torque (or velocity) offset. The offset range is -32,768 to +32,767 (-10 V to +10 V) with a default value of 0 (0 V).

Note The offset value must be within the range limits set by the <u>Load</u> <u>Torque Limit</u> VI.

Primary Offset (0) is the primary DAC torque (or velocity) offset. The offset range is -32,768 to +32,767 (-10 V to +10 V) with a default value of 0 (0 V).

- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
- **Axis** is the axis to control.

Inp Vect contains the primary and secondary parameters that are the source of data for this VI.

Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

The Load Torque Offset VI loads offset values for the DACs mapped to the selected servo axis. This VI has no effect on stepper axes or independent DAC outputs that are not mapped to an axis. When a DAC is connected to a velocity block servo amplifier, the torque offset functions as a velocity offset.

A torque (or velocity) offset shifts the DAC output(s) by the programmed offset value without requiring any action from the PID loop. In a servo system, this can be used to overcome amplifier input offsets, system imbalances, or the effects of outside forces such as gravity. Different torque offsets can be loaded for the primary and secondary DAC.

Note When an Axis is killed, its DAC outputs are zeroed regardless of the torque offset loaded.

DAC offsets can be used in conjunction with DAC range limits to interface to servo amplifiers with unipolar input ranges (for example, 0 to 5 V or 0 to 10 V).

Example

Calling the Load Torque Offset VI with Primary Offset = 4,096 and Secondary Offset = 0 shifts the output ranges of the primary DAC mapped to the axis as shown in the following figure:

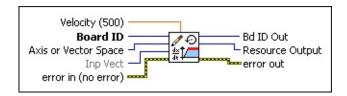
The result of this VI call is to limit the primary DAC to a range of -8.75 V to +10 V with an offset or null value of +1.25 V. This is because even when the PID loop is commanding full negative torque, the torque offset is added and the resulting output is -8.75 V. In the positive direction, the DAC cannot go above +10 V no matter what the offset is.

The VI call leaves the secondary DAC offset at its default value of zero (0). This example assumes the full torque range is available and not limited by the

Load Velocity in RPM

Loads velocity for an axis or vector space in RPM.

Device Compatibility



Details Remarks

Velocity (500) is the double precision floating point velocity value in RPM. The Velocity range depends upon the motor counts or steps per revolution and the trajectory update rate.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis or Vector Space is the axis or vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

The Load Velocity in RPM VI specifies the maximum trajectory velocity for individual axes or vector spaces. When executed on a vector space, the value controls the vector velocity along the vector move path. For velocity control applications, the sign of the loaded velocity specifies the move direction. This VI requires previously loaded values of either counts per revolution (for servo axes) or steps per revolution (for stepper axes) to operate correctly.

RPM is double-buffered, so you can load it on the fly without affecting the move in process, and it takes effect on the next <u>Start</u> or <u>Blend Motion</u> call. After it is loaded, this parameter remains in effect for all subsequent motion profiles until re-loaded by this VI. You do not need to load velocity before each move unless you want to change the velocity.

Note The velocity loaded with this VI is the maximum move velocity. Actual velocity attainable is determined by many factors including PID tuning, length of move, acceleration and deceleration values, and physical constraints of the amplifier/motor/mechanical system.

You also can load velocity in counts/s or steps/s by calling the <u>Load Velocity</u> VI.

Tip Use <u>Load Move Constraints</u> instead of this VI for more move options.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

 For 73xx motion controllers, RPM values stored in onboard variables are in double-precision IEEE format (f64). For information about the number of variables required to hold an RPM value, refer to <u>Using Inputs and</u> <u>Return Vectors with Onboard Variables</u>.

Note NI-Motion firmware uses a 16.16 fixed-point format internally to represent velocity in counts/sample period or steps/sample period and acceleration/deceleration in counts/sample period² or steps/sample period². If you change counts or steps per revolution or change the update rate using Enable Axes, you must reload the velocity and acceleration/deceleration.

• Velocity values in RPM are converted to an internal 16.16 fixed-point format in units of counts (steps) per sample period (update period) before being used by the trajectory generator. NI-Motion can control velocity to 1/65,536 of a count or step per sample. You can calculate this minimum velocity increment in RPM with the following formula:

minimum RPM = Vmin
$$\times (\frac{1}{T_5}) \times 60 \times (\frac{1}{R})$$

where:

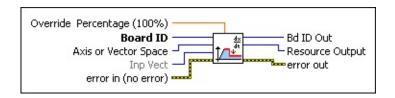
Vmin is 1/65,536 count/sample or step/sample, *Ts* is the sample period in seconds per sample, 60 is the number of seconds in a minute, and *R* is the counts/steps per revolution.

For a typical servo axis with 2,000 counts per revolution operating at the

Load Velocity Override

Loads an instantaneous velocity override for an axis or vector space.

Device Compatibility



Details Remarks

Override Percentage (100%) is a single precision floating point value from 0 to 150%. This value directly scales the programmed velocity. The default value is 100% (no effect).

Board ID is a unique number assigned by Measurement & Automation

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis or Vector Space is the axis or vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

The Load Velocity Override VI scales the operating velocity on an axis or vector space from 0 to 150%. Velocity override is not double-buffered. The VI takes effect immediately and does not require a <u>Start</u> or <u>Blend Motion</u> VI execution to change the operating velocity. All velocity changes use the loaded values of acceleration, deceleration, and s-curve to smoothly transition the velocity to its new value.

Velocity override scales velocity in linear interpolation as well as in circular, helical, and spherical arcs.



Note Velocity override is not valid for contoured moves.

You can load velocity override to individual axes or to a vector space for coordinated velocity scaling. When sent to a vector space, the velocity override is broadcast to all axes in the vector space to change the per-axis overrides. If you later want to operate an axis independently with a different velocity override, you must execute the Load Velocity Override VI again for that axis. Note that this also works with gearing. In addition to following the position of the master axis, the slave follows the velocity of the master axis.

After it is loaded, velocity override remains in effect until changed by another call to this VI. All subsequent moves are at velocities scaled by the most recent override percentage. At power-up reset, velocity override is always reset to 100%.

Velocity override is commonly used in machine tool and other applications to reduce the speed of a programmed motion sequence and can be used to implement a feed hold by setting the value to zero (0). You can directly use a scaled value from an analog input as the velocity override value.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- For 73xx controllers, the resolution of this VI is approximately 0.4%.
- The conversion from floating-point to fixed-point is performed on the host computer, not on the motion controller. To load velocity override from an onboard variable, you must use the integer representation of 0 to 384.
- All axes in a vector space must have the same velocity override. If axes have different velocity overrides, the vector move cannot function as expected. This mode is legal and does not generate an error, but it causes the axes to finish their moves at different times.

NI SoftMotion Controller Considerations

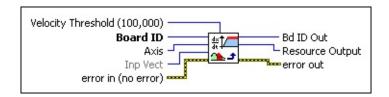
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- If velocity override is loaded on an axis, it is applied only during single-axis moves.
- If the velocity override is loaded on a vector space, it does not overwrite the values for the axes that are in the vector space. In this case, the velocity override is applied only during vector space moves.
- The range for the **Override Percentage** is the full 64-bit range, and is not limited to 150% for the NI SoftMotion Controller.

Load Velocity Threshold

Loads a velocity threshold for an axis.

Device Compatibility



Details Remarks

parameters:

Velocity Threshold (100,000) is the velocity threshold in counts/s (servo axes) or steps/s (stepper axes). For servo axes, the threshold range is 1 to 16,000,000 counts/s. For stepper axes, it is 1 to 1,500,000 steps/s. The factory default value for threshold is the maximum, so the feature is effectively disabled until a threshold is loaded. **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller. **Axis** is the axis to control. **Inp Vect** indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information. **error in (no error)** describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the

status is TRUE if an error occurred before this VI was called, or

error in value to **error out**. The **error in** cluster contains the following

The Load Velocity Threshold VI establishes a velocity threshold for the specified axis, which can then be monitored with the <u>Read Trajectory Status</u> VI. The velocity threshold status is True when the absolute value of filtered axis velocity is above the threshold and False when the velocity drops below the threshold.

Velocity threshold is a status and does not have to be enabled or disabled. Loading a maximum value effectively disables the feature because the status will always be off. Increasing the velocity filter time constant with the Configure Velocity Filter VI reduces quantization noise in the threshold status but at the expense of increasing threshold status latency.

Velocity threshold is typically used to monitor the acceleration and deceleration trajectory periods to see when or if an axis is up to speed. You can then change PID tuning or other parameters as a function of velocity.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

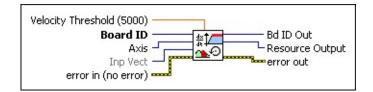
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• Use <u>Load Counts/Steps per Revolution</u> to set **Counts(Steps)/Rev** value to 1. Otherwise, NI-Motion returns an error.

Load Velocity Threshold in RPM

Loads a velocity threshold for an axis in RPM.

Device Compatibility



Details

parameters:

Velocity Threshold (5000) is the double-precision floating point velocity threshold value in RPM. The RPM range depends upon the motor counts or steps per revolution and the trajectory update rate, and is always a positive number.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis to control.
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following

status is TRUE if an error occurred before this VI was called, or

The Load Velocity Threshold VI establishes a velocity threshold in RPM for the specified axis which can then be monitored with the <u>Read Trajectory Status</u> VI. The velocity threshold status is True when the absolute value of filtered axis velocity is above the threshold and False when the velocity drops below the threshold.

Velocity threshold is a status and does not have to be enabled or disabled. Loading a maximum value effectively disables the feature because the status will always be off. Increasing the velocity filter time constant with the <u>Load Velocity Filter Parameter</u> VI reduces quantization noise in the threshold status but at the expense of increasing threshold status latency.

Velocity threshold is typically used to monitor the acceleration and deceleration trajectory periods to see when or if an axis is up to speed. You can then change PID tuning or other parameters as a function of velocity.

Read Blend Status

Reads the Blend Complete status for all axes or vector spaces.

Blend Status

Board ID

Axis or Vector Space

Ret Vect

error in (no error)

Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Axis or Vector Space selects axis, axis control, vector space, or vector space control for move complete, or axis or axis control for all other types.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return</u> Vectors for more detailed information

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

Blending smoothly combines two move segments on an axis, axes, or vector space(s). When continuously blending move segments into each other, it is necessary to wait until the blend is complete between the previous two moves before you load the trajectory parameters for the next move to blend.

The status information returned by the Read Blend Status VI indicates that the previous blend is complete and the axis, axes, or vector space(s) are ready to receive the next blend move trajectory data.

Note Attempting to execute a <u>Blend Motion</u> VI before the previous blend is complete on the axes involved generates a modal error. Refer to <u>Errors</u> and <u>Error Handling</u> for information about errors and error handling.

Example

While blending a 2D vector straight-line move, you call the Read Blend Status VI with Axis or Vector Space = Vector Space Control to select vector space status. If the blend on vector space 1 is still pending, this VI returns a Blend Status, which corresponds to the following bitmap.

Axis/VS 1 Axis/VS 2 Axis/VS 3 Axis/VS 4 Axis 5 Axis 6 Axis 7 Axis 8 False True False False False False False

The blend is complete (1) on vector spaces 2 and 3 (or they do not exist), but the blend is still pending (0) on vector space 1.

For your programming convenience, two utility VIs—<u>Check Blend Complete</u>
<u>Status</u> and <u>Wait for Blend Complete</u>—are provided, which allow you to specify an axis, vector space, group of axes, or group of vector spaces, and find out if a blend is complete, or wait until a blend is complete. These VIs return a simple true/false value indicating whether or not a blend is complete.

Read DAC

Reads the commanded DAC output value for an axis.

Board ID

Axis or DAC

Retn Vect
error in (no error)

Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis or DAC is the axis or DAC to read.
Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8) and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code. Use the error handler VIs to look up the meaning of

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

status is FALSE, **code** is zero or a warning code.

The Read DAC VI returns the value of the specified DAC output. When sent to an axis, this VI returns the value of the primary DAC mapped to that axis. The signed 16-bit value returned corresponds to the ± 10 V full scale range of the DAC.

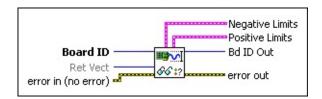
This VI is used to monitor the output command from the PID loop. When the DAC output is connected to a torque block servo amplifier, you can use this value to calculate motor torque or to monitor the acceleration and deceleration portions of a trajectory to see how close the control loop is to saturating at its maximum torque limits.

When the DAC output is connected to a velocity block servo amplifier, the DAC value read is a direct representation of the instantaneous commanded velocity.

Read DAC Limit Status

Reads the status of the DAC limits.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

The Read DAC Limit Status VI returns the positive and negative torque limit status of the DAC output. Independent DACs that are not mapped to axes do not have torque limits, so those DACs will always return False.

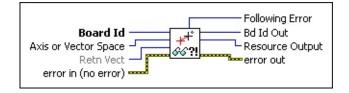
A DAC torque limit status is True (1) when the DAC output is saturated at the corresponding limit. This information tells you that the motor is operating at its maximum torque, probably due to an excessively high value of acceleration or deceleration. It also can indicate excessive friction on the axis, a completely stalled motor, or some other system fault.

When an axis is active (not in the killed, motor off state), this VI returns the instantaneous state of the torque limit circuits. If the axis trips out on following error (a typical occurrence when operating at the torque limits), the DAC limit status is latched so you can tell which limit, positive or negative, caused the following error trip. The status remains latched until the axis is activated again by a <u>Start</u>, <u>Stop</u>, or <u>Blend Motion</u> VI.

Read Following Error

Reads the instantaneous following error for an axis or vector space.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis or Vector Space is the axis or vector space to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

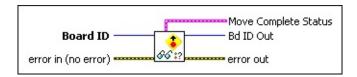
code is a number identifying an error or warning. If **status** is TRUE,

The Read Following Error VI returns the instantaneous following error for the axis or vector space specified. For vector spaces, following error is the root-mean-square of the following errors for the individual axes that make up the vector space.

Read Move Complete Status

Reads the move complete status (MCS) register.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Move Complete Status is a bitmap of MCS for up to eight axes. The

This section includes more detailed information about using this VI.



Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- This VI performs a direct read of the Move Complete Status (MCS) register on the 73xx controller. Because a register read is virtually instantaneous and does not affect communication processing or other NI-Motion operations, you can call this VI repeatedly to get the most up-to-date status for the axes.
- Move Complete Status is configurable individually for each axis with the Configure Move Complete Criteria VI. The criteria for considering motion to be complete include Profile Complete, Run/Stop, In-Position, Settling time delay, and so on.

Note Reading the MCS register immediately after calling the <u>Start</u> Motion VI might not return the status you expected. The Start Motion can still be buffered in the communications FIFO when the instantaneous read of the MCS occurs.

• This VI also returns the state of the User Status bits. You can set and reset these three bits during onboard program execution as general-purpose flags to the host computer. Refer to the <u>Set User Status MOMO</u> VI for more information.

Note When the 73xx motion controller is in the Power-Up state, the MCS register contains a power-up code that describes why the controller is in the Power-Up state. For a list of these power-up codes, refer to the <u>Clear Power Up Status</u> VI.

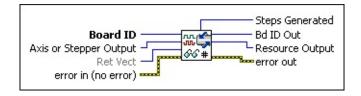
NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using

Read Steps Generated

Reads the number of steps generated by a stepper output.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis or Stepper Output is the axis or stepper output to read.

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

The Read Steps Generated VI returns a number based on steps generated by a stepper axis or stepper output resource. For open-loop stepper axes, this VI returns exactly the same value as the <u>Read Position</u> VI.

For closed-loop stepper axes, this VI returns the actual number of steps generated while the <u>Read Position</u> VI returns the feedback position converted from counts to steps. The number of steps generated includes extra steps added during any pull-in move required to reach the target position.

To reset the value returned by this VI, use <u>Reset Position</u>. Reset Position sets the **steps** parameter to the reset position value.

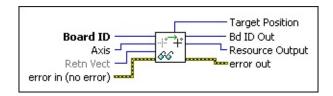
Closed-Loop Stepper Example

Suppose you execute a move from position 0 to position 100, and 10 additional steps are generated for the pull-in move. Read Position returns 100, but Read Steps Generated returns 110. If you execute another move to position 50, Read Position returns 50, and Read Steps Generated returns 60.

Read Target Position

Reads the destination position of the current motion trajectory.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis is the axis to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

The Read Target Position VI returns the destination position of the motion trajectory currently in process. If the axis is stopped, it returns the target position of the last trajectory completed.

This VI differs from the <u>Read Position</u> VI in that it returns the commanded target (destination) position rather than the actual feedback position.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

 Loading a new target position with <u>Load Target Position</u> does not affect the value read until a start motion occurs. When calling <u>Blend Motion</u>, the value read does not reflect the last target position loaded until after the blend occurs.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

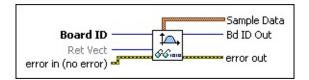
- The NI SoftMotion Controller returns the target position loaded regardless of whether the <u>Start Motion</u> VI has been called.
- The NI SoftMotion Controller returns the target position of individual axes, but does not return the target position loaded by the <u>Load Vector</u> <u>Space Position</u> VI, even if the axis is configured as part of the vector space.

The NI SoftMotion Controller does not support reading the target position of a vector space.

Read Trajectory Data

Reads a sample of acquired data from the samples buffer.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Ret Vect contains position and velocity. Position is returned in onboard variable n in counts (steps), and velocity is returned in onboard variable n+1 in counts/s (steps/s).

Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code

The Read Trajectory Data VI is used to read back a **single sample** of acquired data from the onboard sample buffer. The number of samples, the time between samples and the size of each sample is set when you execute the <u>Acquire</u> <u>Trajectory Data</u> VI. The sample buffer operates first-in-first-out (FIFO), so multiple calls to this VI return samples in their correct time sequence.

While it is possible to read the sample buffer while samples are still being acquired, you must wait enough time between calls to the Read Trajectory Data VI to avoid emptying the buffer.



Note Attempting to read an empty sample buffer generates an error. Refer to Errors and Error Handling for information about errors and error handling.

The Acquire and Read Trajectory Data VIs are used to acquire and read back time-sampled position and velocity data for analysis and display. These VIs implement a digital oscilloscope that is useful during system set up, PID tuning, and general motion with data acquisition synchronization.

Example

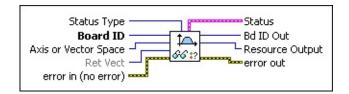
The <u>Acquire Trajectory Data</u> VI is executed with axes 1, 2, and 4 selected. Each call to the Read Trajectory Data VI returns one sample with an array size of six and the following data in the array:

```
Sample Data[] = { Axis 1 position, Axis 1 velocity,
Axis 2 position, Axis 2 velocity
Axis 4 position, Axis 4 velocity }
```

Read Trajectory Status

Reads the selected motion trajectory status of all axes or vector spaces.

Device Compatibility



Details

Status Type is the selector for the type of trajectory status to read.

Value Definition

- 0 Run/Stop
 - 1 Motor Off
 - 2 Velocity Threshold
 - 3 Move Complete
- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
- **Axis or Vector Space** selects axis, axis control, vector space, or vector space control for move complete, or axis or axis control for all other types.
- **Ret Vect** indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 cm)
 - through 0xF8), and do not return data (0). Refer to <u>Input and Return</u> <u>Vectors</u> for more detailed information.
 - error in (no error) describes error conditions that occur before this VI

The Read Trajectory Status VI returns a multi-axis status bitmap of the status type selected. You can select one of the following three trajectory status types:

Run/Stop Status

Run/Stop status is updated based on filtered velocity and run/stop criteria set by the <u>Load Run/Stop Threshold</u> VI. The status is set when the axis is travelling fast enough to be considered running. The axis is considered to be running when the change in position per sample period exceeds the Run/Stop threshold set with the Load Run/Stop Threshold VI.

Motor Off Status

A motor can be Off for two reasons. Either a kill stop was executed or the following error trip point was exceeded. A Motor Off condition also means that an enabled inhibit output is active. Refer to the <u>Configure Inhibit Outputs</u> VI for more information.

Velocity Threshold Status

The Velocity Threshold status indicates whether the axis velocity is above (True) or below (False) the programmed velocity threshold. Refer to the <u>Load Velocity Threshold</u> VI for information about setting and using a velocity threshold.

Move Complete

The Move Complete status indicates whether an axis or vector space is in the move complete state, which is the default when an axis or vector space is idle. While a move is in progress, the move complete status is false. For a move to be complete on a vector space, the move complete status must be true on all axes in the vector space.

During a vector space move, if one axis in a vector space trips out on a

Arc VIs

You can use Arc VIs to load parameters for circularly interpolated moves. It includes 2D circular arcs, 3D helical arcs, and full 3D spherical arcs VIs.

Circular interpolation is an advanced feature of NI-Motion and is primarily used in continuous path applications such as machining, pattern cutting, liquid dispensing, robotics, and so on. For maximum smoothness and accuracy, the NI-Motion DSP implements arcs through a cubic spline algorithm.

Arc VIs are always sent to a vector space. Velocity and acceleration parameters loaded by executing those VIs on the vector space are used as the vector velocity and vector acceleration for all subsequent arc moves. All arc parameters are double-buffered and take effect upon the next Start Motion or Blend Motion VI execution.

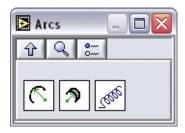


Note Arc radius determines the practical range for vector acceleration and N velocity. Unrealizable vector values generate an error and the start or blend does not execute.

You can blend arc moves into linearly interpolated moves and vice versa. You also can load all axes in the vector space in the same blend factor with the Load Blend Factor VI.

Arc moves are defined relative to their starting position and as such, are inherently operated in relative position mode. This approach guarantees that the axes are already on the circle in the x'y' plane, and avoids any impossible situations where the end point of the last move and the beginning of the arc move are not coincident. The mode selected with the <u>Set Operation Mode</u> VI has no effect on the arc move. It can, however, affect the linearly interpolated vector move you might be blending into or from.

Click the icons for VI descriptions.



Load Circular Arc

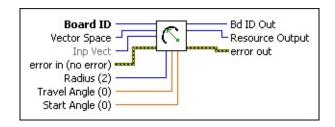
Load Spherical Arc

Load Helical Arc

Load Circular Arc

Loads parameters for making a circular arc move in a 2D or 3D vector space.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Vector Space is the vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

Note The conversion from floating-point to fixed-point is performed on the host computer, not on the motion controller. To load arc VIs from onboard variables, you must use the 16.16 fixed-point representation for all angles.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Load Circular Arc VI defines an arc in the xy plane of a 2D or 3D vector space.



Note Load the velocity and acceleration/deceleration before loading an arc.

The arc is specified by a radius, starting angle, and travel angle and like all vector space moves, uses the loaded values of vector acceleration and vector velocity to define the motion along the path of the arc. The following figure defines a circular arc.

1 Circular Arc 3 Starting Position 5 Radius

2 Travel Angle 4 Start Angle 6 Ending Position

Circular arcs are not limited to ±360°. Moves of over 4,000 circular revolutions in either direction can be started with one call to this VI.

Note If you are doing onboard programming and are using Inp Vect to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by Inp Vect in the following order: Radius, Start Angle, Travel Angle.

Refer to NI 73xx Arc Angles in Degrees for more information about representing angles numerically.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The range for **Radius (2)** is 2 to 2^{31} –1 counts (steps).
- The range for **Start Angle (0)** is 0 to 359.999313° where angle 0 is along the positive x axis and values increase counterclockwise from the positive x axis in the xy plane.
- The range for **Travel Angle (0)** is –1,474,560° to +1,474,200° (–4,096 to +4,095 revolutions).

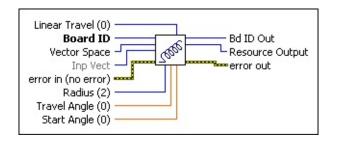
NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The range for **Radius (2)** is 0 to 2^{31} –1 counts (steps).
- The range for **Start Angle (0)** is 0 to 360° where angle 0 is along the positive x axis and values increase counterclockwise from the positive x axis in the xy plane.
- The range for **Travel Angle (0)** is the full 64-bit range.

Load Helical Arc

Loads parameters for making a helical arc move in a 3D vector space.



Device Compatibility

Details Remarks

Linear Travel (0) is the linear travel of the z axis in counts (servo axes) or steps (stepper axes).

Note Loading a zero (0) for Linear Travel reduces the helical arc to a circular arc.

Board ID is a unique number assigned by Measurement & AutomationExplorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Vector Space is the vector space to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

Note Note

The conversion from floating-point to fixed-point is performed on the host computer, not on the motion controller. To load arc VIs from onboard variables, you must use the 16.16 fixed-point representation for all angles.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs

The Load Helical Arc VI defines an arc in 3D vector space that consists of a circle in the xy plane and synchronized linear travel in the z axis.

Note Load the velocity and acceleration/deceleration before loading an arc.

The arc is specified by a radius, starting angle, travel angle, and z axis linear travel, and like all vector space moves, uses the loaded values of vector acceleration and vector velocity to define the motion along the helical path of the arc. The following figure defines a helical arc.

- 1 Helical Arc 3 Start Angle 5 Radius 7 Ending Position
- 2 Travel Angle 4 Starting Position 6 Linear Travel

Like circular arcs, helical arcs are not limited to ±360°. Moves of up to 4,096 helical twists in either direction can be started with one call to this VI.

Note If you are doing onboard programming and are using **Inp Vect** to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by **Inp Vect** in the following order: Radius, Start Angle, Travel Angle, Linear Travel.

Refer to NI 73xx Arc Angles in Degrees for more information about representing angles numerically.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The range for **Linear Travel (0)** is $-(2^{31})$ to $+(2^{31}-1)$ counts (steps).
- The range for **Radius (2)** is 2 to 2^{31} –1 counts (steps).
- The range for **Start Angle (0)** is 0 to 359.999313° where angle 0 is along the positive x axis and values increase counterclockwise from the positive x axis in the xy plane.
- The range for **Travel Angle (0)** is –1,474,560° to +1,474,200° (–4,096 to +4,095 revolutions).

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The range for **Linear Travel (0)** -($\pm 2^{31} 1$).
- The range for **Radius (2)** is 0 to 2^{31} –1 counts (steps).
- The range for **Start Angle (0)** is 0 to 360° where angle 0 is along the positive x axis and values increase counterclockwise from the positive x axis in the xy plane.
- The range for **Travel Angle (0)** is the full 64-bit range.

Load Spherical Arc

Loads parameters for making a spherical arc move in a 3D vector space.

Plane Yaw (0)
Plane Pitch (0)

Board ID

Vector Space

Inp Vect
error in (no error)

Radius (2)

Travel Angle (0)

Start Angle (0)

Device Compatibility

Details Remarks

Plane Yaw (0) is the double precision floating point value, in degrees, of the angle between the x' and x axes when the entire x'y'z' vector space is rotated around the z axis. The z' axis remains aligned with the z axis. When Plane Yaw equals 90°, the positive x' axis is aligned with the positive y axis.

Note Loading zeros for Plane Pitch and Plane Yaw reduces the spherical arc to a circular arc.

Internally, the floating-point values for Plane Pitch, Plane Yaw, Start Angle, and Travel Angle are represented as scaled, fixed point numbers.

Plane Pitch (0) is the double precision floating point value, in degrees, of the angle between the x' and x axes when the entire x'y'z' vector space is rotated around the y axis. The y' axis remains aligned with the y axis. When Plane Pitch equals 90°, the positive x' axis is aligned with the negative z axis.

The Load Spherical Arc VI defines an arc in the x'y' plane of a coordinate system that has to be transformed by rotation in pitch and yaw from the normal 3D vector space (xyz). In the transformed x'y'z' space, the spherical arc is reduced to a simpler circular arc. It is specified by a radius, starting angle, and travel angle, and like all vector space moves, uses the loaded values of vector acceleration and vector velocity to define the motion along the path of the arc in the x'y' plane.

Note Load the velocity and acceleration/deceleration before loading an arc.

The following figure shows a graphic representation of the transformation between the x'y'z' and xyz coordinate spaces. The formal definitions of Plane Pitch and Plane Yaw are listed in the previous section.

Pitch and yaw transformations are inherently confusing because they interact. To avoid ambiguities, you can think about spherical arcs and coordinate transformations as follows:

- The spherical arc is defined as a circular arc in the x'y' plane of a transformed vector space x'y'z'. The original vector space xyz is defined by the Configure Vector Space VI.
- The transformed vector space x'y'z' is defined in orientation only, with no absolute position offset. Its orientation is with respect to the xyz vector space and is defined in terms of pitch and yaw angles.
- Yaw angle rotation comes before pitch angle rotation.
- When rotating through the yaw angle, the y' axis never leaves the original xy plane as the newly defined x'y'z' vector space rotates around the original z axis.
- When rotating through the pitch angle, the y and y' axes stay aligned with each other while the x'z' plane rotates around them.
- At the beginning of the move, the axes are considered to be already on the arc in the x'y' plane. This avoids any impossible situations where the end point of the last mayo and the beginning of the arg mayo are no

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The range for **Radius (2)** is 2 to 2^{31} –1 counts (steps).
- The range for **Plane Pitch (0)** is 0 to 90°.
- The range for **Plane Yaw (0)** is 0 to 359.999313°.
- The range for **Start Angle (0)** is 0 to 359.999313° where angle 0 is along the positive x axis and values increase counterclockwise from the positive x axis in the xy plane.
- The range for **Travel Angle (0)** is –1,474,560° to +1,474,200° (–4,096 to +4,095 revolutions).

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The range for **Radius (2)** is 0 to 2^{31} –1 counts (steps).
- The range for **Plane Pitch (0)** is 0 to 360°.
- The range for **Plane Yaw (0)** is 0 to 360°.
- The range for **Start Angle (0)** is 0 to 360° where angle 0 is along the positive x axis and values increase counterclockwise from the positive x axis in the xy plane.
- The range for **Travel Angle (0)** is the full 64-bit range.

Gearing and Camming VIs

You can use Gearing and Camming VIs to set up and control master-slave gearing and camming on the NI motion controller. It includes VIs to configure a gear master, load a gear ratio, configure a camming master, enable camming, and so on.

Gearing and camming are advanced features of NI-Motion and are used in applications where either the master axis is not under control of the NI motion controller or when extremely tight synchronization between multiple axes is required.

Note If you are gearing a stepper motor and the gear ratio is such that the slave axis exceeds the maximum stepper velocity when the master does not, the slave axis does not move and may trip on following error. To prevent this, ensure the gear ratio does not permit the slave axis to exceed the maximum stepper velocity. Refer to your motion controller documentation for information about maximum stepper velocities.

Click the icons for VI descriptions.



Configure Gear Master

Load Gear Ratio

Enable Gearing

Enable Gearing Single Axis

Configure Camming Master

Load Camming Parameter

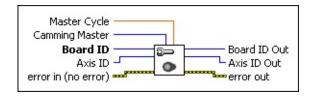
Enable Camming

Enable Camming Single Axis

Configure Camming Master

Configures the master device for camming operation.

Device Compatibility



Details Remarks

	Master Cycle is	the repeat period	for the master dev	vice.
--	-----------------	-------------------	--------------------	-------

Camming Master is the axis, encoder, or ADC channel to use as the master.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis ID is the axis to configure.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRIJE

Use this VI to assign a master axis, encoder, or ADC channel to the slave axis selected for the camming application. Any number of slave axes can have the same master, but each slave axis must have only one master.

Before starting the camming operation, you must configure the master device. Using an axis as the master device allows the slave axis to follow the trajectory generation output of the master axis. This functionality is useful when the master axis is an open-loop system or when it is used for handling the latency of a feedback device (encoder or ADC), which causes high following error.

You also can use a feedback device, such as an encoder or ADC, as the master device so that any movement made by the master is immediately propagated to the slave.

The master device position is interpreted, or modularized, within the master cycle, which allows the camming operation to repeat continuously as the master device moves. Along with slave positions, the master cycle is used to calculate the camming ratio at any time.

For example, if the master cycle is 5,000 and the current master position is 12,500, the master position inside the master cycle is

```
position = ((12,500 \text{ modulus } 5,000) = 2,500)
```

Using the previous master cycle but with a master position of -4,000, the master position inside the cycle is

```
position = ((-4,000 \text{ modulus } 5,000) = 1,000)
```

You also can use the <u>Load Camming Parameter</u> VI to change the camming cycle.

Refer to the <u>Camming</u> section of the *NI-Motion Help* for more information about camming operations.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

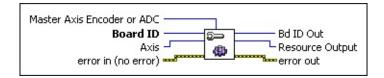
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• The NI SoftMotion Controller does not support using analog feedback as the camming master.

Configure Gear Master

Assigns a master axis, encoder, or ADC channel for master-slave gearing.

Device Compatibility



Details Remarks

Master Axis Encoder or ADC is the axis (1 through 15) trajectory
generator, encoder (0x21 through 0x2F), or ADC channel (0x51 through
0x5F) to use as the master for this slave axis. A zero (0) value means no
master is assigned (default).
Note Axis and Master Axis, Encoder, or ADC cannot be the same

Note Axis and Master Axis, Encoder, or ADC cannot be the same because an axis cannot be its own master.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis is the slave axis to control.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

The Configure Gear Master VI assigns a master axis, encoder, or ADC channel to the slave axis selected. Any number of slave axes can have the same master, but each slave axis can have only one master. A slave axis may also serve as master axis for any number of slave axes.

You must call the Configure Gear Master VI prior to enabling master-slave gearing with the <u>Enable Gearing</u> VI. Typically, the source for the master position is an independent encoder, an ADC channel, or the feedback resource of an enabled axis. In each case, you assign the resource, not the axis, as the master. The gear master, which can be an axis, encoder, or ADC, must be enabled before you can use this VI. Use the <u>Enable Axes</u>, <u>Enable Encoders</u>, or <u>Enable ADCs</u> VIs to enable the gear master.

When an axis is assigned as the master, its trajectory generator output (not its feedback position) is used as the master position command. This mode of operation can eliminate the following error skew between the master and slave axes and is especially useful in gantry applications. The master axis can operate in any mode (including as a slave to another master).

Master-slave functionality of slave axes is in addition to their normal mode of operation. This allows a point-to-point move to be <u>superimposed</u> upon the slave while the slave axis is in motion due to being geared to its master. This functionality is useful for registration and reference offset moves. Refer to the <u>Gearing</u> section of the *NI-Motion Help* for information about using superimposed moves/registration applications. Also, refer to the <u>Rotating Knife</u> section of the *NI-Motion Help* for example code that includes superimposed moves.

Refer to the <u>Load Gear Ratio</u> and <u>Enable Gearing</u> VIs for more information about master-slave gearing.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

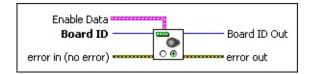
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• The NI SoftMotion Controller does not support using analog feedback as the gear master (analog gearing).

Enable Camming

Enables camming operation for multiple axes.

Device Compatibility



Details

	Ena	ble Data is an array of clusters that include the following elements:
		Axis is the axis you want to start or stop camming operation on.
[86)	TF	Enable indicates if the camming is started (True) or stopped (False) on the selected Axis .
		Position is the position to start or stop the camming at. The valid range is -1 or 0 through (master cycle -1).
	_	

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE,

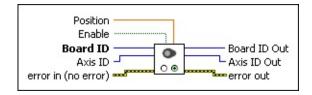
Use this VI to simultaneously start or stop a camming operation on multiple axes. The default value for the **Position** element of the **Enable Data** array is -1, which starts camming immediately. To start camming at a specific master position, set this parameter to a value between 0 and (master cycle -1).

Refer to the <u>Camming</u> section of the *NI-Motion Help* for more information about camming operations.

Enable Camming Single Axis

Enables camming operation for the current axis.

Device Compatibility



Details

- **Position** is the position to begin camming at. The default is -1, and the valid range is -1 or 0 through (master cycle -1).
- Enable enables/disables camming on the specified axis. Set Enable to TRUE to enable camming or FALSE to disable camming.
- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
- **ISSET Axis ID** is the axis to configure.
 - **error in (no error)** describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:
- **status** is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code

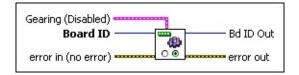
Use this VI to start or stop a camming operation on an axis. You can start or stop the camming operation immediately by setting **Position** to -1. To configure the camming operation to take place after the master device crosses a specific position inside the master cycle, set **Position** to a value between 0 and (master cycle -1).

Refer to the <u>Camming</u> section of the *NI-Motion Help* for more information about camming operations.

Enable Gearing

Enables slave axes for masterslave gearing.

Device Compatibility



Details

Gearing (Disable) is the bitmap of slave axes to enable for gearing.

- **Axis 1**—If true, gearing is enabled. If false, gearing is disabled (default).
- **Axis 2**—If true, gearing is enabled. If false, gearing is disabled (default).
- **Axis 3**—If true, gearing is enabled. If false, gearing is disabled (default).
- **Axis 4**—If true, gearing is enabled. If false, gearing is disabled (default).
 - **Axis** 5—If true, gearing is enabled. If false, gearing is disabled (default).
 - **Axis 6**—If true, gearing is enabled. If false, gearing is disabled (default).
 - **Axis** 7—If true, gearing is enabled. If false, gearing is disabled (default).
 - **Axis 8**—If true, gearing is enabled. If false, gearing is disabled (default).

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI

The Enable Gearing VI enables and disables master-slave gearing functionality of slave axes. When gearing is enabled, the positions of the slave axes and their corresponding masters are recorded as their absolute gearing reference. For axes and encoders, the current position is the value recorded. For ADCs, the binary value is recorded. From then on, as long as the gear ratio remains absolute, every incremental change of a master position is multiplied by the corresponding absolute gear ratio and applied to the slave axis. Refer to the Load Gear Ratio VI for information about absolute versus relative gear ratios.

Note Changing the absolute gear ratio causes the slave axes to jump to a new position specified by the new gear ratio. With servo motor systems the slave jumps immediately. With stepper systems, if the new slave position is within 4,000 steps, it jumps to the new position. Otherwise, the slave axis waits until the commanded slave position moves to within 4,000 steps and then jumps to the new position. The slave axis then resumes following the master axis according to the gear ratio.

You must call the <u>Configure Gear Master</u> and <u>Load Gear Ratio</u> VIs prior to enabling master-slave gearing. In addition, you must enable and activate the slave axes before enabling gearing. An error is generated if a slave is killed when gearing is enabled. These checks ensure that the slave axis enables in a controlled fashion.

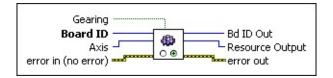
Note If you are running a slave axis move with a stepper axis, it is possible that the slave axis geared velocity exceeds the maximum step rate of the controller. In this case, the controller kills the axis and sets the following error status to true. Refer to the *Specifications* section of your controller user manual for the maximum step rate of your controller.

You can call the Enable Gearing VI at any time to disable gearing or to reenable gearing with new absolute gearing reference positions. If gearing is disabled on a moving axis, the axis immediately stops but remains active. If the slave axis was also implementing a superimposed move, the superimposed

Enable Gearing Single Axis

Enables slave axis for master-slave gearing.

Device Compatibility



Details

Gearing	g (Disal	ble) ind	licates w	hether	to e	nable or	disable t	the slav	e axis
🚥 for gear	ing. If t	rue, gea	aring is o	enabled	l. If	false, ge	aring is o	disabled	1
(default).						_		
- I			1		11	3.6	. 0	Δ.	. •

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis is the slave axis to enable or disable.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code. Use the error handler VIs to look up the meaning of

This VI is similar to the Enable Gearing VI, but allows you to enable or disable gearing on a single axis without affecting the other axes.

The Enable Gearing Single Axis VI enables and disables master-slave gearing functionality of a slave axis. When gearing is enabled, the position of the slave axis and its corresponding master is recorded as its absolute gearing reference. From then on, as long as the gear ratio remains absolute, every incremental change of a master position is multiplied by the corresponding absolute gear ratio and applied to the slave axis. Refer to the <u>Load Gear Ratio</u> VI for information about absolute versus relative gear ratios.

Note Changing the absolute gear ratio causes the slave axes to jump to a new position specified by the new gear ratio. With servo motor systems the slave jumps immediately. With stepper systems, if the new slave position is within 4,000 steps, it jumps to the new position. Otherwise, the slave axis waits until the commanded slave position moves to within 4,000 steps and then jumps to the new position. The slave axis then resumes following the master axis according to the gear ratio.

You must call the <u>Configure Gear Master</u> and <u>Load Gear Ratio</u> VIs prior to enabling master-slave gearing. In addition, you must enable and activate the slave axis before enabling gearing. An error is generated if the slave is killed when gearing is enabled. These checks ensure that the slave axis enables in a controlled fashion.

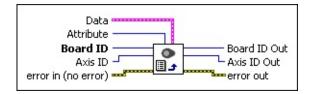
Note If you are running a slave axis move with a stepper axis, it is possible that the slave axis geared velocity exceeds the maximum step rate of the controller. In this case, the controller kills the axis and sets the following error status to true. Refer to the *Specifications* section of your controller user manual for the maximum step rate of your controller.

You can call the Enable Gearing Single Axis VI at any time to disable gearing or to re-enable gearing with new absolute gearing reference positions. If

Load Camming Parameter

Loads other camming parameters, such as master axis position offset and slave axis position offset.

Device Compatibility



Details

Data is a cluster of variables that contains the values for the selected **Attribute**.

Integer is 32-bit integer data.

Double is 64-bit floating point data.

Boolean is Boolean data.

Attribute is the attribute to configure. The following list includes valid attributes:

- Master Cycle—Set the Double element of the Data cluster to the
 master cycle in user units. The master cycle indicates how often the
 camming cycle repeats itself. For example, if you specify 8,000
 counts for the Master Cycle, the camming repeats itself every 8,000
 counts.
- Master Offset—Double element of the Data cluster to the master offset in user units. Master offset takes into account the position difference between the current position of the master axis and the position at which the camming begins. Range is 0 to (Master Cycle − 1).

Use this VI to configure various parameters of the camming operation:

• **Master Cycle** defines when the camming operation is going to repeat based on the master position. You can configure the master cycle at the same time you configure the camming master using the <u>Configure Camming Master VI</u>.

Note You cannot change the master cycle after camming is enabled.

• **Master Offset** is applied to the master position when camming is enabled. This attribute shifts the position modulus to compensate for the fact that the camming cycle may not start at 0. For example, if you have a master cycle of 4,000 and a master offset of 0, the camming operation repeats at ..., -8,000, -4,000, 0, 4,000, 8,000, ... If you change the master offset to 1000, the camming operation repeats at ..., -7,000, -3,000, 1,000, 5,000, 9,000,

The default value for the master offset is 0, and the valid range of values for the master offset is 0 to (Master Cycle -1).

Tip The master offset is applied only when camming is started or enabled.

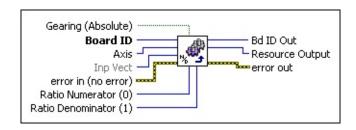
• **Slave Offset** is applied when the camming cycle begins. This offset is used to adjust the camming profile when the beginning and ending positions for the slave axis are not the same. You can change the slave offset at anytime, but the change takes effect on the next camming cycle.

The default value is 0.

Refer to the <u>Camming</u> section of the *NI-Motion Help* for more information about camming operations.

Load Gear Ratio

Loads the gear ratio for master-slave gearing.



Device Compatibility

Details

parameters:

TF	Gearing (Absolute) selects absolute FALSE or relative TRUE gearing between the master and slave.
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis is the slave axis to control.
	Inp Vect contains the type, numerator, and denominator parameters that are the source of data for this VI.
	Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Load Gear Ratio VI loads the gear ratio of the slave axis relative to its master and selects whether this ratio is absolute or relative. The ratio is loaded as a numerator and denominator because it is a natural format for a ratio (numerator: denominator) and it allows a broad range of ratios, from 1:32,767 to 32,767:1. The ratio is always specified as slave relative to master (slave:master).

When you execute the <u>Enable Gearing</u> VI, the positions of the slave and its master are recorded as their absolute gearing reference. From then on, as long as the gear ratio remains absolute, every incremental change of the master's position is multiplied by the absolute gear ratio and applied to the slave axis.

If a relative gear ratio is selected and loaded after gearing is enabled, the position of the master is recorded as its relative reference point and every incremental change from this reference point is multiplied by the relative gear ratio and applied to the slave axis or axes.

Note While changing an absolute gear ratio on the fly is allowed, you must be careful because the slave axis jumps with full torque to the position defined by the new ratio even when the master position has not changed. Similarly, resetting the position on a master axis also causes its slave axes to jump to a new position relative to the new position of the master axis.

The Load Gear Ratio VI must be called prior to enabling master-slave gearing with the <u>Enable Gearing</u> VI. Often the positions of the master and slave are reset to zero or some known position prior to enabling gearing, though this is not always required. The execution of the Enable Gearing VI stores both positions as offsets and gears them from that point onward.

Relative gearing changes the specified position of the slave axis only when the master moves. It uses the length of the current move of the master axis to calculate how far to move the slave axis. At the end of the move, if the slave axis is not at the exact position, the position error is discarded. This may

Buffered Operations VIs

Buffered operations allow you to create and manage memory buffers on the NI motion controller. You can create buffers that can hold up to 16,364 points, read and write data to and from these buffers, and check the status of a buffer. Buffers can even be stored in ROM for later use.

Onboard buffers allow you to store much more data than the 120 onboard variables. Data can be moved back and forth between buffers and onboard variables by using input and return vectors.

For example, an onboard buffer can be created to hold a series of target positions to move to, one after another. An onboard program is written to read a position from the buffer to an onboard variable, then use that value to load a target position, start a move, and wait for the move to complete. The same process is repeated for each target position until the entire buffer of positions is consumed. This approach allows for a large number of moves to execute independent of the host computer.

Besides single-shot applications where the buffer holds all of the data or is filled with all of the data only once, it is possible to use an onboard buffer in a continuous fashion, allowing for very large amounts of data to transfer to or from the motion controller. Use the Check Buffer VI to determine the number of points that can be safely written to or read from the onboard buffer at any point in time.

The 7350 and 7340 controllers have the following onboard memory:

Controller	$\mathbf{R}\mathbf{A}\mathbf{M}$	ROM
7340	1 64k sector	2 64k sectors
7350	2 64k sectors	4 64k sectors

You can access a buffer from either RAM or ROM, but you cannot split buffers between the two, and you cannot split buffers between the 64 KB ROM sectors.

Contouring is a special movement mode implemented on top of the buffered operations. Contouring allows you to specify a series of positions n milliseconds

apart for the motion controller to spline through.

You can use the Buffered Operations VIs to perform buffered high-speed captures and buffered breakpoints. These allow you to achieve faster rates than would otherwise be possible by automatically storing captured positions or automatically loading breakpoints. Refer to High-Speed Capture Overview and **Breakpoint Overview** for more information.

Note Configuring and clearing buffers is a processor-intensive operation on the NI motion controller that requires the allocation and de-allocation of memory. Configure and clear buffers only when motors are not moving and onboard programs are not running.

Click the icons for VI descriptions.



Configure Buffer

Write Buffer

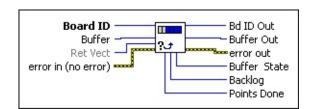
Read Buffer

Check Buffer

Clear Buffer

Check Buffer

Returns information about the current state of the buffer.



Vectors for more detailed information.

Device Compatibility

Details

Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Buffer is the buffer to configure. Valid buffer numbers are 1 through 255 (0x01 through 0xFF).
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data
to a variable (0x01 through 0x78), return data to an indirect variable (0x81

through 0xF8), and do not return data (0). Refer to Input and Return

Board ID is a unique number assigned by Measurement & Automation

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE,

The Check Buffer VI provides you with current information about the state of the buffer. In single iteration situations, meaning **Total Points** is less than or equal to **Buffer Size** in the <u>Configure Buffer</u> VI, the **Buffer State** parameter is most useful in determining that the operation has completed.

For a finer-resolution view of the state of the buffer, the **Backlog** parameter indicates the number of points that can safely be written to or read from the buffer without violating the **Old Data Stop** condition. Refer to **Configure** Buffer for more information.

Points Done is the number of points read from the buffer by the controller, in the case of an output buffer, or written to the buffer by the controller, in the case of an input buffer. **Points Done** is from the point of view of the controller, rather than the user.

As an example, in the case of contouring, Backlog helps you determine when you can write more data to the buffer, while **Points Done** tells you how many points have actually been splined together by the controller. **Points Done** reflects what is happening in the real world, where motion occurs.



Note This VI returns data in the following order: **Backlog**, **Buffer State**, and **Points Done**.

Output Buffer Example

After you configure a 10-point buffer for contouring, the contents are indeterminate and both the read and write heads are at position 0.

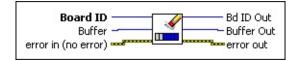
Write: 0 Read: 0

In writing the values 1 through 5 to a buffer, the write head moves to the sixth position in the buffer, where the next write takes place. The read head is still at position 0 because it has read no data from the buffer. Likewise, **Points Done** is

Clear Buffer

Clears the previously configured buffer and clears any associations between resources and the specified buffer.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Buffer is the buffer to configure. Valid buffer numbers are 1 through 255
(0x01 through 0xFF).

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

After a buffered operation completes, it is important to clear the buffer by calling the Clear Buffer VI. Otherwise, the memory allocated for the buffer in the onboard RAM is not available for other uses, such as the allocation of another buffer or the storage of an onboard program.

A buffer cannot be cleared while it is in use. If you try to clear a buffer while it is in use, the Clear Buffer VI generates a Buffer In Use error.

Note Configuring and clearing buffers is a processor-intensive operation on the motion controller that requires the allocation and deallocation of memory. You must configure and clear buffers only when motors are not moving and onboard programs are not running. For example, if you wish to execute three simultaneous contouring operations on axis 1, axis 2, and vector space 1 (with axes 3 and 4), you must first configure all three buffers before starting any of the operations. You can start the contour operations independently, and at different times, but wait until all operations are complete before clearing any of the buffers.

Advanced Uses

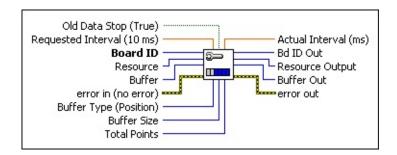
This VI only clears a buffer from RAM. If you save a buffer to ROM using the Object Memory Management VI, you must delete the buffer from ROM using the Object Memory Management VI. If you call the Clear Buffer VI and specify a buffer that resides in ROM, the association between the resource and the buffer is cleared, but the buffer remains in ROM.

Note You cannot clear a buffer that is attached to an axis that is moving. You can clear a buffer that is active, such as a high-speed capture buffer that is waiting for a trigger, but NI-Motion returns an error in this case.

If multiple resources refer to the same buffer, all associations from all resources to the specified buffer are cleared. Refer to the *Advanced Uses* section in the <u>Configure Buffer</u> VI description for more information.

Configure Buffer

Configures a buffer for use in buffered operations.



Device Compatibility

Details Remarks

Old Data Stop (True) indicates to the motion controller whether or not to stop reading from or writing to the buffer when older, unread data is encountered. This is only relevant when Total Points is greater than Buffer Size. In the output case, such as contouring, when Old Data Stop is true and Total Points is greater than Buffer Size, you must write new data to the buffer before the motion controller finishes the buffer and starts over at the beginning. Likewise, for an input buffer, if the controller (or an onboard program running on the controller) is writing data to the buffer, such as buffered high-speed capture data, you must read data out of the buffer before the controller wraps around and writes new data on top of the old data.

If you set Old Data Stop to false, the motion controller does not enforce the old data stop condition, and simply writes data to or reads data from the buffer as needed.

Requested Interval (10 ms) is additional data for the Position buffer type. The Position buffer type requires a Requested Interval parameter, and indicates the time between contouring data points in milliseconds. For all other buffer types, the Dequested Interval parameter is ignored.

After a buffer is configured, you can write to the buffer with the <u>Write Buffer</u> VI and read from the buffer with the Read Buffer VI.

A contour operation that uses a buffer is initiated when the <u>Start Motion</u> VI is called. You check the state of the buffer with the Check Buffer VI to determine how many points have been read from or written to the buffer by the motion controller.

After a buffered operation is complete, it is important to clear the buffer by calling the <u>Clear Buffer</u> VI. Otherwise, the memory allocated for the buffer in the onboard RAM is not available for other uses, such as the allocation of another buffer. Buffers must only be cleared when no motors are moving and no programs are running.

A buffer cannot be reconfigured or cleared while the buffer is in use. The VI generates a Buffer In Use error in such a case.

A buffer can only be used by one resource at a time. Multiple resources can be associated with a single buffer, but only one resource can use that buffer at any given time. Likewise, a single resource can make use of many buffers, but for buffers of the same type, only one of them can be associated with that resource at a given time. Refer to the NI 73xx Controller Considerations section of this VI description for more information.



Note Configuring and clearing buffers is a processor-intensive operation on the NI motion controller that requires the allocation and de-allocation of memory. Configure and clear buffers only when motors are not moving and onboard programs are not running.

Example Uses

General-Purpose Input and General-Purpose Output Examples

Configure a 300-point output buffer and fill it with position, velocity, and

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

Using input vectors and return vectors with the <u>Read Buffer</u> and <u>Write Buffer</u> VIs, you can write to and read from the buffer with data stored in onboard variables. You must limit the number of points to read or write because there are only 120 onboard variables. Data is read from or written to onboard variables sequentially starting at the variable specified by **Inp Vect** or **Ret Vect** in the <u>Write Buffer</u> VI or <u>Read Buffer</u> VI, respectively.

Specify a Buffer Size of 0 to associate a resource (like an axis or vector space) to an existing buffer, without affecting the size of the buffer or the data contained in the buffer. The following examples illustrate the use of this technique:

- A single resource, for example, an axis or vector space, can use buffer
 1 for a contouring operation, and then switch to buffer 2 for a
 subsequent operation. You can set up the two buffers ahead of time,
 and when the first operation is complete, you can make an additional
 call to Configure Buffer with a Buffer Size of 0, the same value for
 Resource, and the new buffer number for Buffer.
- If you wish to reuse a buffer with another resource, simply call the Configure Buffer VI with the same buffer number, a Buffer Size of 0, and a new value for Resource. Both resources can still use the single buffer, but not at the same time.

A **Buffer Size** of 0 allows you to associate a resource with an existing buffer, without changing the contents of the existing buffer. Also, a **Buffer Size** of 0 can be used to leave the current buffer alone, but

Read Buffer

Reads data from a previously configured buffer.

Board ID
Buffer Out
Ret Vect
error in (no error)
Number of Points (0)

Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation

Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Puffer is the buffer to configure. Valid buffer numbers are 1 through 255

Buffer is the buffer to configure. Valid buffer numbers are 1 through 255 (0x01 through 0xFF).

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78) or variable array (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

When returning data to Ret Vect, you must limit Number of Points to a number that fits in a sequence of onboard variables starting with the variable specified by Ret Vect.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

An input buffer can be written to after it has been configured, usually by an onboard program. After data is available, you read the data from the buffer with the Read Buffer VI.

When the buffer is configured for a single iteration, you can wait until the buffer is full by checking its status using the <u>Check Buffer</u> VI, and then read the entire contents of the buffer. You also can read smaller chunks of data from the buffer while the operation is in progress. The number of points available to read can be determined with the Backlog parameter returned by <u>Check Buffer</u>.

When using the buffer as a circular buffer (Total Points > Buffer Size) with Old Data Stop turned on, you must read data from the buffer faster than the buffer is filled with new data. You can determine the number of points you can safely read from the buffer by calling the <u>Check Buffer</u> VI.

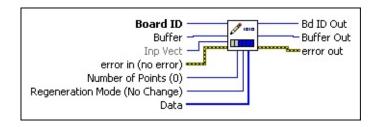
The Backlog parameter tells you how many points you can safely read from the buffer.

If you try to read data that has not yet been written to the buffer by the controller, the data is not read, and the controller generates an Old Data Stop error. If you configure the buffer with Old Data Stop set to false, however, you can read data freely from the buffer, and the controller does not enforce the Old Data Stop condition.

Read Buffer generates an illegal buffer operation error if you try to read data from an output buffer after the specified number of points (Total Points) has already been read from the buffer.

Write Buffer

Writes data to a previously configured buffer.



Device Compatibility

Details

parameters:

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Buffer is the buffer to configure. Valid buffer numbers are 1 through 255 (0x01 through 0xFF).
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

After a buffer has been configured for an output operation such as contouring, you must write data to the buffer before starting the operation. When all of the data fits in the buffer (Buffer Size Total Points), you need only fill the buffer one time with the appropriate data.

When using the buffer as a circular buffer (Total Points > Buffer Size), with Old Data Stop turned on, you need to write fresh data to the buffer faster than the motion controller consumes the data. After filling the buffer once, and starting the buffered operation, you can determine the number of points you can safely write to the buffer by calling the Check Buffer VI. The Backlog parameter tells you how many points you can safely write to the buffer without overwriting data that has yet to be consumed by the controller.

If you try to write new data on top of old data, which is data that has not been consumed by the controller, the write command does not execute, and the controller generates an Old Data Stop error. If you configure the buffer with Old Data Stop set to false, however, the controller does not enforce the Old Data Stop condition and you can write data freely to the buffer.

Example Uses

Basic Uses

Suppose you have a contour move of 25,107 points, and configure a buffer with Buffer Size of 1,000 points and Total Points of 25,107. You set Old Data Stop in the Configure Buffer VI to True, and use the Check Buffer and Write Buffer VIs as you write data to the buffer in small chunks while the data is consumed by the motion controller. When you write the last points to the controller, you can still set Regeneration Mode to 0 (no change) and the motion controller knows you are done writing data to the buffer because you have written 25,107 points in total. When the controller is unable to read any more because the read mark has caught up to the write mark, the controller checks Regeneration Mode, recognizes this stop is intentional because the total points written to the buffer is equal to Total Points, and completes the operation without generating

Start & Stop Motion VIs

You can use Start & Stop Motion VIs to start, blend, and stop motion.

You can execute all of the NI-Motion start and stop VIs on an individual axis, simultaneously on multiple axes, on a vector space, or simultaneously on multiple vector spaces. These VIs give complete control over the state of the motors in the system and with the addition of the Find Reference VIs, are the only NI-Motion VIs that can actually initiate motion.

Click the icons for VI descriptions.



Start Motion

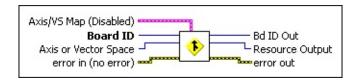
Blend Motion

Stop Motion

Blend Motion

Blends motion on a single axis, single vector space, multiple axes, or multiple vector spaces.

Device Compatibility



Details

Axis/VS Map is the bitmap of axes or vector spaces to blend. It is only required when you select multiple axes or vector spaces with the Axis or Vector Space parameter.

When blending axes (Axis or Vector Space = 0):

- **TITIE** Axis/VS 1—If true, blend axis. If false, do not blend axis.
- **TIP Axis/VS 2**—If true, blend axis. If false, do not blend axis.
- **EXECUTE:** Axis/VS 3—If true, blend axis. If false, do not blend axis.
- **TIME** Axis/VS 4—If true, blend axis. If false, do not blend axis.
- **TITIE Axis**/**VS 5**—If true, blend axis. If false, do not blend axis.
- **EXECUTE:** Axis/VS 6—If true, blend axis. If false, do not blend axis.
- **TEP Axis/VS** 7—If true, blend axis. If false, do not blend axis.
- **TITIE** Axis/VS 8—If true, blend axis. If false, do not blend axis.

When blending vector spaces (Axis or Vector Space = 0x10):

- **Axis/VS 1**—If true, blend vector space. If false, do not blend vector space.
- Axis/VS 2—If true, blend vector space. If false, do not blend vector

The primary difference between a Start Motion VI and a Blend Motion VI is that the Start Motion VI is immediate and preemptive, while the Blend Motion waits and starts the next move upon the completion of the previous move.

The Blend Motion VI is used to blend motion profiles on axes or vector spaces, either simultaneously or individually. A blend is similar to a normal start and has the same requirements for valid trajectory parameters as the <u>Start Motion</u> VI. The blended move uses the most recently loaded values of acceleration, velocity, target position, s-curve, operation mode and so on to generate the motion profile.

Note If a stepper axis is in a killed state (not energized), halt the axis using the Stop Motion VI, with Stop Type set to halt stop, before you execute a Start Motion or Blend Motion VI. After you halt the axis, you might need to wait before executing a Start Motion or Blend Motion VI, so that the stepper drive comes out of reset state. If the stepper drive does not come out of reset state before you execute the VI, the stepper axis might lose some steps during acceleration. To determine whether you need to wait before executing the VI, refer to the stepper drive documentation or vendor.

Blend starting smoothly blends two move segments on an axis, axes, or vector space(s). There are three types of blends, controlled by the blend factor:

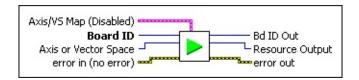
- Blend moves by superimposing the deceleration profile of the previous move with the acceleration profile of the next move (blend factor = -1).
- Blend moves by starting the next move at the exact point when the previous move has stopped (blend factor = 0).
- Start the next move after a programmed delay time between the end of the previous move and the start of the next move (blend factor > 0 ms).

Refer to the <u>Load Blend Factor</u> VI for more information about how blend factor controls the blending of motion profiles.

Start Motion

Starts motion on a single axis, single vector space, multiple axes, or multiple vector spaces.

Device Compatibility



Details

Axis/VS Map is the bitmap of axes or vector spaces (coordinate spaces) to wait on. It is only required when multiple axes or vector spaces are selected with the **Axis or Vector Space** parameter.

When starting axes (**Axis or Vector Space** = 0):

- **Axis/VS Map 1** —If true, start axis. If false, do not start axis.
- **TITIE Axis/VS Map 2** —If true, start axis. If false, do not start axis.
- **EXECUTE: Axis/VS Map 3** —If true, start axis. If false, do not start axis.
- **Axis/VS Map 4** —If true, start axis. If false, do not start axis.
- **Axis/VS Map 5** —If true, start axis. If false, do not start axis.
- **TITIE Axis/VS Map 6** —If true, start axis. If false, do not start axis.
- **Axis/VS Map 7** —If true, start axis. If false, do not start axis.
- **Axis/VS Map 8** —If true, start axis. If false, do not start axis.

When starting vector spaces (**Axis or Vector Space** = 0x10):

Axis/VS Map 1 —If true, start vector space. If false, do not start vector space.

Axis/VS Map 2 —If true, start vector space. If false, do not start

The Start Motion VI is used to start a motion profile on axes or vector spaces, eit and uses the most recently loaded values of acceleration, velocity, target position motion profile.

Note If a stepper axis is in a killed state (not energized), halt the axis using t before you execute a Start Motion or <u>Blend Motion</u> VI. After you halt the ax Motion or Blend Motion VI, so that the stepper drive comes out of reset state before you execute the VI, the stepper axis might lose some steps during acc executing the VI, refer to the stepper drive documentation or vendor.

You also can use the Start Motion VI to update trajectory parameters to a move to after the start take effect immediately upon the next start without requiring the movelocity profiling and other continuous motion applications. The execution of a parameter of move, and the controller. Refer to the following table for more information

Move Type	Controller Type	Same Direction
Single Axis Move	73xx Controller	Move starts immediately ¹
Single Axis Move	NI SoftMotion Controller	Move starts immediately
Vactor Space Move	73xx Controller	Motion decelerates to a stop, tl
Vector Space Move	NI SoftMotion Controller	Move starts immediately

¹ In most cases, the move starts immediately. However, if the new target position position, the move decelerates to a stop, then starts. Refer to the figures below for

The first image demonstrates the case where Start Motion is called with a new ta position as the current position. In this case, motion decelerates to a stop, then stawhere Start Motion is called with a new target position on the opposite side of the case, the move continues to the new target position without stopping.

² The time required to start the next move depends on the deceleration time. If the longer than expected to start.

ces or vector spaces, either simultaneously or individually. A start is preemptive velocity, target position, s-curve, operation mode, and so on to generate the

ed), halt the axis using the <u>Stop Motion</u> VI, with Stop Type set to halt stop, I. After you halt the axis, you might need to wait before executing a Start comes out of reset state. If the stepper drive does not come out of reset state some steps during acceleration. To determine whether you need to wait before ation or vendor.

parameters to a move that is already in process. Trajectory parameters loaded without requiring the motion to come to a stop. You can use this feature for is. The execution of a preemptive start depends on the parameters entered, the ble for more information.

Same Direction

Opposite Direction

tarts immediately ¹	Motion decelerates to a stop, then starts ²
tarts immediately	Motion decelerates to a stop, then starts ²
decelerates to a stop, then starts ²	Motion decelerates to a stop, then starts ²
tarts immediately	Motion decelerates to a stop, then starts ²

the new target position is closer to the current position than the previous target to the figures below for more information.

deceleration time. If the deceleration time is very slow, the next move may take

is called with a new target position on the same side of the original target erates to a stop, then starts again. The second image demonstrates the case the opposite side of the original target position as the current position. In this ut stopping.

Stop Motion

Stops motion on a single axis, single vector space, multiple axes, or multiple vector spaces.

Three types of stops can be executed:

- Decelerate to stop
- Halt stop
- Kill stop

Stop Type (Decel)
Axis/VS Map (Disabled)
Board ID
Axis or Vector Space
error in (no error)

<u>Details</u> <u>Remarks</u>

Device Compatibility

Stop Type (Decel) is the type of stop to execute, as follows.

Value Definition



- Decelerate
- 1 Halt
- 2 Kill

Axis/VS Map is the bitmap of axes or vector spaces to stop. It is only required when multiple axes or vector spaces are selected with the Axis or Vector Space parameter. NI-Motion ignores additional axes for controllers that operate less than eight axes.

When stopping axes (**Axis or Vector Space** = 0):

TET Axis/VS Map 1—If true, stop axis. If false, do not stop axis.

This VI is used to stop a motion profile on axes or vector spaces, either simultantypes of stops with the Stop Motion VI: decelerate stop, halt stop, and kill stop.

Decelerate Stop

When a decelerate stop is executed (decelerate to stop), the axis, axes, or vector sportion of their trajectory profile as controlled by previously loaded deceleration therefore dependent upon this deceleration trajectory.

Halt Stop

In contrast, a halt stop has the following behavior depending on the state of the a

- If the axis is not moving and not energized, the axis is energized.
- If the axis is not moving and energized, calling a halt stop has no effect.
- If the axis is moving:
 - o On servo axes, full torque is applied to stop the motor(s) as soon as the
 - On stepper axes, the step pulses are ceased as quickly as possible. How
 following error after halting the move. This following error will be cor
 In both cases, NI-Motion attempts to stop the motor(s) with a near infinite r
 motion is not controlled by previously loaded deceleration and S-curve para

Note For stepper axes configured for p-command mode, ensure that the previously killed to avoid position error between the controller and the moving resets your reference position to an undefined location, and you executing another move.

Kill Stop

On servo axes, a kill stop disables the control loop and zeros the output DAC, all stepper axes, a kill stop ceases the stepper pulse generation as soon as the commatypes, there is no trajectory profile during a kill stop. If enabled, the inhibit output stepper drive. You can enable the inhibit outputs and set their polarity to active h

spaces, either simultaneously or individually. You can execute three different alt stop, and kill stop.

ie axis, axes, or vector space(s) immediately begin to follow the deceleration sly loaded deceleration and s-curve parameters. The actual stopped position is

ing on the state of the axis:

is energized. t stop has no effect.

motor(s) as soon as the command is received by the motion controller. uickly as possible. However, it is possible that there may be a small amount of owing error will be compensated for with a pull-in move during the next move. s) with a near infinite rate of deceleration. There is no trajectory profile and ration and S-curve parameters.

In the controller and the drive. In this case, a call to halt stop while the axis is a stiffined location, and you should perform a Find Home reference move before

ros the output DAC, allowing frictional forces alone to stop the motion. On n as soon as the command is received by the motion controller. On both axis abled, the inhibit output is activated to inhibit (disable) the servo amplifier or heir polarity to active high or active low with the Configure Inhibit Outputs VI.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

• If an axis that is part of a vector space is individually killed, the other axes in the vector space are decelerated to stop. If a program attempts to start axes that have been manually stopped by the host computer, it is overruled and put into the paused state.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- If an axis that is part of a vector space is individually killed, the other axes in the vector space are also killed.
- You can execute this VI only on an axis that is enabled.

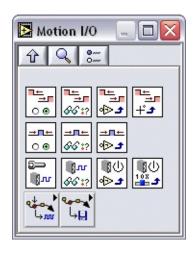
Motion I/O VIs

You can use Motion I/O VIs to set up and control the motion I/O features of the NI motion controller. Motion I/O VIs include VIs to set polarity and enable limit and home inputs, high-speed capture inputs and inhibit outputs, VIs to configure and control breakpoint outputs, and VIs to read the status of all the motion I/O signals, high-speed captured position, and software limit status.

All of the dedicated motion I/O also can function as general-purpose digital I/O when they are not being used for their motion specific features. You can set and reset outputs, you can read inputs at any time, and you can set and change their polarity as required.

Motion I/O VIs include limits and other basic Motion I/O VIs, <u>Breakpoint VIs</u>, and <u>High-Speed Capture VIs</u>.

Click the icons for VI descriptions.



Enable Limits
Read Limit Status
Set Limit Input Polarity
Load Software Limit Positions
Enable Home Inputs
Read Home Input Status

Set Home Input Polarity
Configure Drive Signal
Read Drive Signal Status
Configure Inhibit Outputs
Set Inhibit MOMO

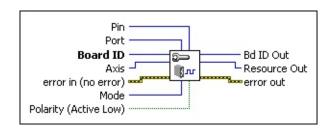
Subpalettes

Breakpoint VIs

High-Speed Capture VIs

Configure Drive Signal

Configures the drive signal.



Device Compatibility

Details

Pin is the specific bit (0 to 7) in the port specified by port. For the HSC
port, trigger 1 = pin 0, trigger 2 = pin 1, trigger 3 = pin 2, trigger 4 = pin 3
trigger 5 = pin 4, trigger 6 = pin 5, trigger 7 = pin 6, and trigger 8 = pin 7.
Port is the general purpose I/O port (1 to 8) or HSC software port (10) to control.
Board ID is a unique number assigned by MAX used to send and receive commands and data to or from a specific NI motion controller.
Axis is the axis to configure.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:
parameters.

status is TRUE if an error occurred before this VI was called, or

code is a number identifying an error or warning. If status is TRUE,

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

status is FALSE, **code** is zero or a warning code.

Use this VI to configure and reserve I/O lines to receive signals from the drive. The following drive signals are available: In-Position and Drive Fault (Servo Alarm).

Tip Use Configure Motion I/O Map instead of this VI for additional drive signal support.

Use the In-Position drive signal when the drive is controlling a servo motor and the input to the drive is step/direction. This signal tells the motion controller when the drive considers the motor to be at the commanded position.

When the In-Position drive signal is configured, the Move Complete status is linked to the state of the In-Position signal. The move is not considered complete unless the In-Position signal is active.

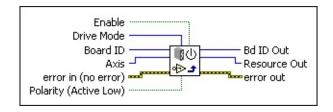
Note Refer to Write Trajectory Data to add or remove this signal from the move complete criteria.

Use the Drive Fault drive signal to connect a servo alarm signal to the motion controller so that when an alarm or other drive fault occurs, the motion is stopped using a kill stop. A kill stop disables the control loop and zeroes the DAC so that frictional forces stop the motion.

Configure Inhibit Outputs

Sets polarity and enables the per-axis inhibit outputs.

Device Compatibility



Details Remarks

parameters:

TF	Enable enables/disables inhibit output on the selected axis.
	Drive Mode specifies either Open Collector (default) or Totem Pole mode. Drive Mode is available only on the NI 7350 and is not configurable on NI 7330/40/90 motion controllers.
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis is the axis to configure.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following

status is TRUE if an error occurred before this VI was called, or

code is a number identifying an error or warning. If status is TRUE,

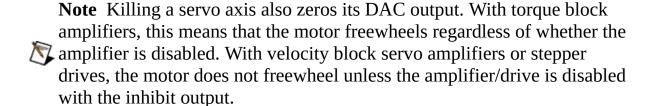
FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

status is FALSE, **code** is zero or a warning code.

The Configure Inhibit Outputs VI enables/disables the inhibit outputs, sets the **Polarity** (active low or active high) of the axis inhibit outputs, and sets the **Drive Mode**. When enabled, a per-axis inhibit output is linked to the motor off state of the corresponding axis. A killed axis (motor off) forces the corresponding inhibit output On. When the axis is active, the inhibit output is Off. Notice that this is the logical state. The actual state depends on the polarity of the system.

Inhibit outputs are typically used to disable the servo amplifier or stepper drive for power saving, safety, or specific application reasons.

For more information about **Open Collector** and **Totem Pole** drive modes, refer to the *NI 7350 Hardware User Manual*. Configuring **Drive Mode** is supported only on 7350 controllers and is not configurable on NI 7330/40/90 motion controllers.



You also can use inhibit outputs as general-purpose outputs. Disabled inhibit outputs ignore the state of their corresponding axis and can be directly controlled through the <u>Set Inhibit MOMO</u> VI.

You can configure the polarity of each inhibit output as active low or active high. When configured as active low, the output is active when there is a low signal on the output pin. Conversely, active high means that the output is active when there is a high signal on the output pin. The inhibit polarity is always in effect, whether the inhibit is linked to its axis (enabled) or directly controlled through the Set Inhibit MOMO VI.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

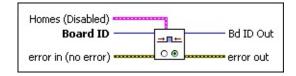
The following includes considerations you must make when you are using this VI with a 73xx motion controller:

The PCI-7390 does not support the **Polarity** parameter. To set the active state on a PCI-7390 use Write Digital I/O Data.

Enable Home Inputs

Enables/disables the home inputs.

Device Compatibility



Details

Homes (Disabled) is the bitmap of home inputs to enable:

- **Axis 1**—If true, home input is enabled. If false, home input is disabled (default).
- **Axis 2**—If true, home input is enabled. If false, home input is disabled (default).
- **Axis** 3—If true, home input is enabled. If false, home input is disabled (default).
- **Axis 4**—If true, home input is enabled. If false, home input is disabled (default).
 - **Axis** 5—If true, home input is enabled. If false, home input is disabled (default).
 - **Axis 6**—If true, home input is enabled. If false, home input is disabled (default).
 - **Axis** 7—If true, home input is enabled. If false, home input is disabled (default).
 - **Axis 8**—If true, home input is enabled. If false, home input is disabled (default).

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

The Enable Home Inputs VI enables/disables any combination of axis home inputs. An enabled home input causes a halt stop on the axis when the input becomes active. You can configure each home input as active low or active high with the <u>Set Home Input Polarity</u> VI. You also can use a home input as a general-purpose input and read its status with the <u>Read Home Input Status</u> VI.

Home inputs are an enhancement on the NI motion controller and are not required for basic motion control. You can operate all motion control VIs without enabling or using the home inputs except the <u>Find Reference</u> VI, which requires enabled limit and home inputs for operation when the Find Reference type is "home."

To configure the property of Find Reference Home, such as **Initial search direction** and **Home edge to stop on**, use the <u>Load Reference Parameter</u> VI.



Note An active and enabled home input transition on an axis that is part of \mathbb{N} a vector space move causes that axis to halt stop and the other axes in the vector space to decelerate to a stop.

Example

To enable the home inputs for axes 2 and 4, call the Enable Home Inputs VI with **Homes**, as follows.

Home 1 Home 2 Home 3 Home 4 Home 5 Home 6 Home 7 Home 8 False True False True False False False False

Enable Limits

Enables/disables either the forward and reverse limit inputs or the forward and reverse software position limits.

Limit Type (Limit Inputs)

Board ID

error in (no error)

Forward Limits (Disabled)

Reverse Limit (Disabled)

Device Compatibility

Details Remarks

Limit Type (Limit Inputs) selects the type of limit to enable, either the hardware limit switch inputs or the software position limits, as follows.

Value Definition

- 0 Limit inputs
- 1 Software limits

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE,

The Enable Limits VI enables/disables any combination of axis limits. You can ephysical limit inputs (hardware) or the logical position limits (software) depending Limit Type selected. You can enable or disable forward and reverse limits separate enable both software and hardware limits on an axis or axes, call this VI twice.

The limit inputs are typically connected to end-of-travel limit switches or sensors enabled limit input causes a halt stop on the axis when the input becomes active. configure each limit input as active low or active high with the <u>Set Limit Input P</u> Active limit inputs also prohibit attempts to start motion that would cause addition the direction of the limit. You also can use limit inputs as general-purpose inpured their status with the <u>Read Limit Status</u> VI.

Note For the end-of-travel limits to function correctly, the forward limit swi sensor must be located at the positive (count up) end of travel and the reverse the negative (count down) end of travel.

Similarly, software limits are often used to restrict the range of travel further and hitting the hardware limit switches. An enabled software limit causes the axis to decelerate to a stop when the limit position is reached or exceeded. To enable the limits, you must first find the reference point using the <u>Find Reference</u> VI.

Even when disabled, you can use the host computer to poll the software limits to out-of-range position. Refer to the <u>Load Software Limit Positions</u> and the <u>Read I</u> VIs for information about loading and reading the forward and reverse software

Hardware limit inputs and software position limits are enhancements on the NI n controller and are not required for basic motion control. You can operate all moti VIs without enabling or using these limits except the <u>Find Reference</u> VI, which r enabled limit and home inputs for operation. Refer to the following VIs for more information about enabling home inputs: <u>Set Home Input Polarity</u>, <u>Enable Home</u> and <u>Read Home Input Status</u>.

Example

his VI

Limits VI enables/disables any combination of axis limits. You can enable the nit inputs (hardware) or the logical position limits (software) depending upon the selected. You can enable or disable forward and reverse limits separately. To software and hardware limits on an axis or axes, call this VI twice.

puts are typically connected to end-of-travel limit switches or sensors. An it input causes a halt stop on the axis when the input becomes active. You can ach limit input as active low or active high with the <u>Set Limit Input Polarity</u> VI. t inputs also prohibit attempts to start motion that would cause additional travel ion of the limit. You also can use limit inputs as general-purpose inputs and attus with the <u>Read Limit Status VI</u>.

or the end-of-travel limits to function correctly, the forward limit switch or nust be located at the positive (count up) end of travel and the reverse limit at ative (count down) end of travel.

oftware limits are often used to restrict the range of travel further and avoid ever lardware limit switches. An enabled software limit causes the axis to smoothly o a stop when the limit position is reached or exceeded. To enable the software must first find the reference point using the Find Reference VI.

disabled, you can use the host computer to poll the software limits to warn of an e position. Refer to the <u>Load Software Limit Positions</u> and the <u>Read Limit Status</u> rmation about loading and reading the forward and reverse software limits.

mit inputs and software position limits are enhancements on the NI motion nd are not required for basic motion control. You can operate all motion control enabling or using these limits except the <u>Find Reference</u> VI, which requires it and home inputs for operation. Refer to the following VIs for more about enabling home inputs: <u>Set Home Input Polarity</u>, <u>Enable Home Inputs</u>, <u>ome Input Status</u>.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- An active (and enabled) limit input transition on an axis that is part of a vector space move causes that axis to halt stop and the other axes in the vector space to decelerate to a stop.
- If any axis in a vector space move exceeds an enabled software limit position, all axes in the vector space decelerate to a stop.

NI SoftMotion Controller Considerations

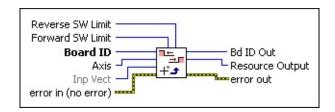
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• An active (and enabled) limit on an axis that is part of a vector space causes all of the axes in the vector space to halt stop.

Load Software Limit Positions

Loads the forward and reverse software limit positions for an axis.

Device Compatibility



Details

Reverse SW Limit is the reverse software limit position in counts (servo axes) or steps (stepper axes). Software limit positions can be anywhere within the 32-bit position range, $-(2^{31})$ to $+(2^{31}-1)$. The default value for the reverse software limit is $-(2^{30})$ counts (steps).

Note The Forward SW Limit cannot be less than the Reverse SW Limit.

Forward SW Limit is the forward software limit position in counts (servo axes) or steps (stepper axes). Software limit positions can be anywhere within the 32-bit position range, $-(2^{31})$ to $+(2^{31}-1)$. The default value for the forward software limit is $+(2^{30})-1$ counts (steps).

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis is the axis to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

The Load Software Limit Positions VI sets the forward and reverse position limit values for the selected axis. When enabled with the <u>Enable Limits</u> VI, a software limit causes the axis to smoothly decelerate to a stop when the limit position is reached or exceeded.

Even when disabled, you can poll the software limits by the host computer or use an onboard program to warn of an out of range position. Refer to the <u>Read Limit Status</u> VI for information about reading the software limit status.

The forward software limit is considered active if the current position is greater than or equal to the forward software limit position. The reverse software limit is considered active if the current position is less than or equal to the reverse software limit position.

Software limits are often used to restrict the range of travel and avoid hitting the hardware end-of-travel limit switches. For example, you can travel at a high velocity until hitting the software limit switch, and then move more slowly until hitting the hardware limit switch.

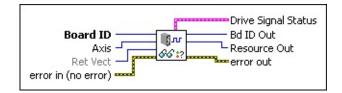
Caution After an axis has stopped due to encountering a software limit switch, you can still move further in the same direction if you command the axis to do so. This behavior is not possible with hardware limits, but is appropriate for software limits.

Note If you are doing onboard programming and are using Inp Vect to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by Inp Vect in the following order: Forward SW Limit, Reverse SW Limit.

Read Drive Signal Status

Returns the status of the drive signal when an active signal is detected.

Device Compatibility



Details

Board ID is a unique number assigned by MAX used to send and receive
commands and data to or from a specific NI motion controller.
Axis is the axis to configure.
Ret Vect is the destination for the return data.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already
occurred, this VI returns the value of error in in error out . The VI runs
normally only if no incoming error exists. Otherwise, the VI passes the
error in value to error out . The error in cluster contains the following
parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

source is a string that indicates the origin of the error, if any.

Typically **source** is the name of the VI in which the error occurred.

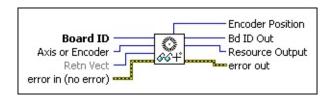
Use this VI to read the status of the drive signal when an active drive signal is detected. If In-Position is active, bit 0 = True. If Drive Fault is active, bit 1 = True.

Tip Use Read Motion I/O Data instead of this VI for more options.

Read Encoder Position

Reads the position of an encoder.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Axis or Encoder is the axis or encoder to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE. **code** is zero or a

The Read Encoder Position VI returns the quadrature count value of the encoder selected. The encoder must be enabled, either directly through the Enable Encoders VI or automatically, by being mapped to an enabled axis.

The Read Encoder Position VI is typically used to read the value of an encoder that is not part of an axis. This encoder could be a master encoder used for master-slave gearing or an independent position or velocity sensor.

For reading encoders mapped to axes, you can call this VI on the axis or directly on its mapped encoder. For servo axes, both approaches return the same value as the <u>Read Position</u> VI. On stepper axes however, this VI can return additional useful information.

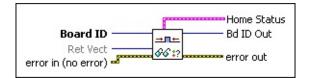
During axis set up, you can operate the closed-loop stepper axis in open-loop mode and use this VI to directly measure the counts per revolution and steps per revolution for the axis. These values must be loaded in advance for subsequent closed-loop operation. Refer to the <u>Load Counts/Steps per Revolution</u> VI for more information.

You also can use this VI to return a finer reading of position in cases where the encoder resolution greatly exceeds the step resolution of the closed-loop stepper axis.

Read Home Input Status

Reads the instantaneous status of the home inputs.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return</u> Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

The Read Home Input Status VI returns the logical state of the home inputs. You can execute this VI at anytime to monitor the home inputs, whether they are enabled or not. A home input enabled with the **Enable Home Inputs** VI causes a halt stop on an axis when its home input becomes active (True). You also can use a home input as a general-purpose input and read its status with this VI. The home input status during a **Find Reference** move is undefined.

To determine the home found status, use Read Reference Status and set the reference type to "home".



Note This VI returns undefined values for axes that are currently performing find reference moves.



Note This VI reads the logical state (On or Off, True or False) of the home inputs. The polarity of the home inputs determines whether an On state is active high or active low. Refer to the **Set Home Input Polarity** VI for more information.

Read Limit Status

Reads the instantaneous state of either the hardware limit inputs or the software limits.

Board ID
Ret Vect
error in (no error)

Reverse Limits
Forward Limits

Device Compatibility

Details

Limit Type (Limit Inputs) selects the type of limit status to read, either the hardware limit switch inputs or the software position limits, as follows.

Value Definition

- 0 Limit inputs
- 1 Software limits

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return</u> Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI

The Read Limit Status VI returns either the hardware limit input status or the software position limit status, depending on the limit type selected. When **Limit Type** = Limit Inputs (0), this VI returns the logical state of the forward and reverse limit inputs.



Note This VI returns undefined values for axes that are currently performing find reference moves.



Note The polarity of the limit inputs determines whether an On state is active high or active low. Refer to the Set Limit Input Polarity VI for more information.

Alternatively, when Limit Type = Software Limits (1), this VI returns the state of the forward and reverse software limits. A True (On) indicates that the forward or reverse limit position for the corresponding axis has been reached or exceeded. The limit status during a <u>Find Reference</u> move is undefined.

You can read the status of the limit inputs and the software position limits at any time, whether the limits are enabled or not. Enabled limits cause axes to stop when their state transitions True. Refer to the **Enable Limits** VI for more information.

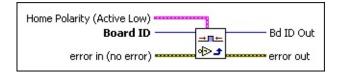


Note If you are doing onboard programming and are using Retn Vect to write data, this VI writes the variables starting at the memory address pointed to by Retn Vect in the following order: Forward Limits, Reverse Limits.

Set Home Input Polarity

Sets the polarity of the home inputs as either active low/active open or active high/active closed.

Device Compatibility



Details

Home Polarity is the bitmap of active polarities for the home inputs:

- **Axis 1**—If TRUE, active low/active open (default). If FALSE, active high/active closed.
- **Axis 2**—If TRUE, active low/active open (default). If FALSE, active high/active closed.
- **Axis 3**—If TRUE, active low/active open (default). If FALSE, active high/active closed.
- **Axis 4**—If TRUE, active low/active open (default). If FALSE, active high/active closed.
 - **Axis** 5—If TRUE, active low/active open (default). If FALSE, active high/active closed.
 - **Axis 6**—If TRUE, active low/active open (default). If FALSE, active high/active closed.
 - **Axis** 7—If TRUE, active low/active open (default). If FALSE, active high/active closed.
 - **Axis 8**—If TRUE, active low/active open (default). If FALSE, active high/active closed.

Board ID is a unique number assigned by Measurement & Automation

The Set Home Input Polarity VI defines the active state for each home input as either active low/active open or active high/active closed.

When configured as active low, the input is active when there is a low signal on the input pin. Conversely, active high means that the input is active when there is a high signal on the input pin.

Configuring an active state of active open or active closed does not correspond to the level of the signal on the input pin. Instead, an active open state means that the input is active when current is not flowing through the optocoupled input. Conversely, an active closed state means that the input is active when current is flowing through the optocoupled input.

You can enable home inputs to cause halt stops when the input becomes active with the <u>Enable Home Inputs</u> VI. You also can use a home input as a general-purpose input and read its status with the <u>Read Home Input Status</u> VI.

To find the home input reference on a system, use <u>Find Reference</u> and configure the search type as "home".

Example

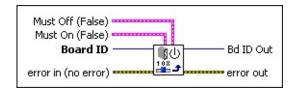
To set the active state of the home inputs on axes 1, 3 and 4 as active low and the rest of the home inputs as active high, call the Set Home Input Polarity VI with Home Polarity as follows.

Home 1 Home 2 Home 3 Home 4 Home 5 Home 6 Home 7 Home 8 True False True False False False

Set Inhibit MOMO

Sets the inhibit outputs using the Must On/Must Off protocol.

Device Compatibility



Details

Must Off (False) is the bitmap of inhibit outputs to force off:

- **Inhibit Output 1**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
- **Inhibit Output 2**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
- **Inhibit Output 3**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
- **Inhibit Output 4**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
 - **Inhibit Output 5**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
 - **Inhibit Output 6**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
 - **Inhibit Output 7**—If true, inhibit output forced off. If false, inhibit output unchanged (default).
 - **Inhibit Output 8**—If true, inhibit output forced off. If false, inhibit output unchanged (default).

Must On (False) is the bitmap of inhibit outputs to force on:

The Set Inhibit MOMO VI controls disabled inhibit outputs being used as genera outputs. You can directly set the inhibit outputs to a logical On or Off state.

Note This VI has no effect on enabled inhibit outputs. These outputs are direby their corresponding axes.

Using the Must On/Must Off protocol allows you to set or reset individual inhibi affecting the other inhibit outputs. This gives you tri-state control over each outp Unchanged. A True in a location of the **Must On** bitmap turns the inhibit On, wh corresponding location of the **Must Off** bitmap turns the inhibit Off. A False in ϵ no affect, so leaving both the **Must On** and **Must Off** at False is effectively a ho of the inhibit output is unchanged. If you set both the **Must On** and **Must Off** bit interpreted as a **Must On** condition and the inhibit is turned On.

Note This VI sets the logical state of an inhibit output On or Off (True or Fa polarity of the inhibit outputs determine whether an On state is active high or Refer to the Configure Inhibit Outputs VI for more information.

The Set Inhibit MOMO VI allows individual control of the inhibit outputs without shadow value to remember the state of other outputs not being set or reset with the state of other outputs.

Example

To turn inhibit output 1 On, output 4 off, and leave the other inhibit outputs unch Set Inhibit MOMO VI with the following parameters:

Must On corresponds to the following:

Must On 1 Must On 2 Must On 3 Must On 4 Must On 5 Must On 6 Must C

VI

OMO VI controls disabled inhibit outputs being used as general-purpose directly set the inhibit outputs to a logical On or Off state.

has no effect on enabled inhibit outputs. These outputs are directly controlled sponding axes.

n/Must Off protocol allows you to set or reset individual inhibit outputs without r inhibit outputs. This gives you tri-state control over each output: On, Off, or ie in a location of the **Must On** bitmap turns the inhibit On, while a True in the ration of the **Must Off** bitmap turns the inhibit Off. A False in either bitmap has ng both the **Must On** and **Must Off** at False is effectively a hold, and the state rut is unchanged. If you set both the **Must On** and **Must Off** bits to True, it is **lust On** condition and the inhibit is turned On.

1 0

On Off

On Unchanged

sets the logical state of an inhibit output On or Off (True or False). The inhibit outputs determine whether an On state is active high or active low. Configure Inhibit Outputs VI for more information.

OMO VI allows individual control of the inhibit outputs without requiring a emember the state of other outputs not being set or reset with the VI.

tput 1 On, output 4 off, and leave the other inhibit outputs unchanged, call the O VI with the following parameters:

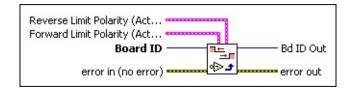
onds to the following:

On 2 Must On 3 Must On 4 Must On 5 Must On 6 Must On 7 Must On 8

Set Limit Input Polarity

Sets the polarity of the forward and reverse limit inputs as either active low/active open or active high/active closed.

Device Compatibility



Details

Reverse Limit Polarity is the bitmap of active polarities for the reverse limit inputs:

- **Axis 1**—If TRUE (default), active low/active open. If FALSE, active high/active closed.
- **Axis 2**—If TRUE (default), active low/active open. If FALSE, active high/active closed.
- **Axis 3**—If TRUE (default), active low/active open. If FALSE, active high/active closed.
- **Axis 4**—If TRUE (default), active low/active open. If FALSE, active high/active closed.
 - **Axis** 5—If TRUE (default), active low/active open. If FALSE, active high/active closed.
 - **Axis 6**—If TRUE (default), active low/active open. If FALSE, active high/active closed.
 - **Axis 7**—If TRUE (default), active low/active open. If FALSE, active high/active closed.

Avis Q If TDIE (default) active level etims on an If EALCE active

The Set Limit Input Polarity VI defines the active state for each forward and reverse limit input as either active low/active open or active high/active closed.

When configured as active low, the input is active when there is a low signal on the input pin. Conversely, active high means that the input is active when there is a high signal on the input pin.

Configuring an active state of active open or active closed does not correspond to the level of the signal on the input pin. Instead, an active open state means that the input is active when current is not flowing through the optocoupled input. Conversely, an active closed state means that the input is active when current is flowing through the optocoupled input.

You can enable limit inputs to cause halt stops when the input becomes active with the <u>Enable Limits</u> VI. You also can use a limit input as a general-purpose input and read its status with the <u>Read Limit Status</u> VI.

Example

To set the active state of the forward and reverse limit inputs on axes 1, 2, 3, and 4 as active low and the forward and reverse limit inputs on the remaining axes as active high, call the Set Limit Input Polarity VI with the following parameters:

Forward Limit Polarity corresponds to the following.

| Forward |
|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| True | True | True | True | False | False | False | False |

Reverse Limit Polarity corresponds to the following.

| Reverse |
|---------|---------|---------|---------|---------|---------|---------|---------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Breakpoint VIs

Position breakpoints are an enhancement to the encoder decoders on the NI motion controller and are available when the encoders operate as axis feedback or as independent encoder resources. Breakpoint functionality is available on servo and closed-loop stepper axes. You can use Breakpoint VIs to configure, enable, and read the position breakpoint status. Advanced VIs, such as periodic breakpoint and buffered breakpoint can be used to achieve higher-frequency breakpoints. To configure buffers for buffered breakpoints, refer to **Buffer VIs**. You also can load a breakpoint position modulus. Like all motion I/O, breakpoint outputs also can function as general-purpose outputs with the Set Breakpoint Output MOMO VI.

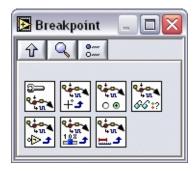


Note All breakpoints can be affected by jitter in the motion system. For example, if you have breakpoint positions very close to each other, the jitter In the motion system could cause the position to change enough to reach the breakpoint when a breakpoint is not intended. Increase the value of the breakpoint window using <u>Set u32</u> to compensate for system jitter.

Refer to the following table for the maximum breakpoint rates for each type of breakpoint. Not all breakpoints are available on all controllers.

Breakpoint Type	Maximum Frequency
Single	150 Hz
Buffered, PID rate of 62.5 μs to 250 μs	2 kHz
Buffered, PID rate greater than 250 µs	1 kHz
Periodic	4 MHz

Click the icons for VI descriptions.



Configure Breakpoint

Load Breakpoint Position

Enable Breakpoint Output

Read Breakpoint Status

Configure Breakpoint Output

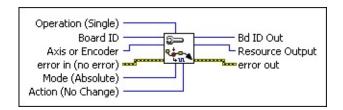
Set Breakpoint Output MOMO

Load Breakpoint Modulus

Configure Breakpoint

Configures a position breakpoint on an axis or encoder.

Device Compatibility



Details

	Operation (Single) selects between single and buffered breakpoint
0	Operation (Single) selects between single and buffered breakpoint operation. Valid values are SINGLE and BUFFERED operation.
	Board ID is a unique number assigned by Measurement & Automation
	Explorer (MAX) used to send and receive commands and data to or from
	a specific NI motion controller.

Axis or Encoder is the axis or encoder to control. You can load **Breakpoint Position** on encoders mapped to axes 1 through 8 or directly on encoders 0x21 through 0x28.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE,

The Configure Breakpoint VI configures the breakpoint to the appropriate mode, operation, and action. It also defines the action to perform when the breakpoint is reached—leave the breakpoint output unchanged, reset the breakpoint output to low, set the breakpoint output to high, or toggle the state of breakpoint output. Refer to the **Synchronization** section of the *NI-Motion Help* for more information about the types of breakpoints you can configure.

Note For modulo breakpoints, the magnitude of the breakpoint value must Note: be less than the breakpoint modulus. If this range is exceeded, a modal error is generated when you execute the **Enable Breakpoint Output** VI.

Enable mode determines how the previously loaded breakpoint position is interpreted. Absolute breakpoints can be anywhere in the 32-bit position range. Relative breakpoints are relative to the instantaneous encoder position when the breakpoint is enabled. Modulo breakpoints are interpreted within the range of the loaded breakpoint modulus. For more information about modulo breakpoints, refer to the <u>Load Breakpoint Modulus</u> VI.

When an enabled breakpoint is reached, a breakpoint event occurs. You can use the <u>Read Breakpoint Status</u> VI to see if a breakpoint has occurred.

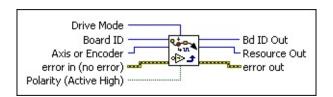
A breakpoint event also can cause the state of the corresponding breakpoint output to change. Action On Breakpoint selects whether the output goes low, goes high, toggles state, or does not change when the breakpoint event occurs. If the breakpoint output is presently in the state defined by Action On Breakpoint, it is forced to the opposite state when the breakpoint is enabled. This guarantees that the appropriate transition occurs when the breakpoint is reached.

Note NI-Motion does not support breakpoint and high-speed capture In functionality on an axis when the primary feedback for that axis is used as analog feedback.

Configure Breakpoint Output

Device Compatibility

Configures the drive mode and polarity of the breakpoint output.



Details

Drive Mode specifies either Open Collector or Totem Pole mode.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis or Encoder is the axis or encoder to configure.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

status is FALSE, **code** is zero or a warning code.

Use the Configure Breakpoint Output VI to configure the **Drive Mode** and **Polarity** of the breakpoint signal. Call this VI at the beginning of the application before enabling the breakpoint with the <u>Enable Breakpoint Output</u> VI.

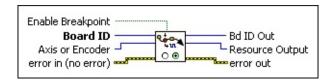
For more information about **Open Collector** and **Totem Pole** drive modes, refer to the documentation for your 7350 device. Configuring **Drive Mode** is supported only on 7350 controllers.

You can configure breakpoint output as active low or active high. When configured as active low, the output is active when there is a low signal on the output pin. Conversely, active high means that the output is active when there is a high signal on the output pin.

Enable Breakpoint Output

Enables a position breakpoint on an axis or encoder.

Device Compatibility



Details

TF	Enable Breakpoint indicates whether to enable or disable the breakpoint output. Values are True or False.
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis or Encoder is the axis or encoder to control. You can load Breakpoint Position on encoders mapped to axes 1 through 6 or directly on encoders 0x21 through 0x24.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

This VI enables or disables the breakpoint with configuration set by the Configure Breakpoint VI. For single breakpoints, you must re-enable if you want another breakpoint. Buffered and periodic breakpoints only need to be enabled once. When an enabled breakpoint is reached, a breakpoint event occurs. You can use the **Read Breakpoint Status** VI to see if a breakpoint has occurred.

You can enable only one breakpoint per encoder or axis at a time. When an enabled breakpoint is reached, the breakpoint is automatically disabled. You must explicitly re-enable the breakpoint to use it again. If you need to disable a previously enabled breakpoint, run this VI with Enable Breakpoint set to False.



Note Enabled breakpoints are also automatically disabled when you execute a <u>Reset Position</u> or <u>Reset Encoder Position</u> VI on the corresponding axis.

Breakpoints are fully functional on independent encoders that are not mapped to axes. In this case, you enable breakpoints directly on the encoder resource.

Note If you use modulus breakpoints or enable the same breakpoint repeatedly at low velocity (< 50 counts/sec), you may get a breakpoint output immediately after you re-enable it, because the position may not have changed from the last breakpoint.



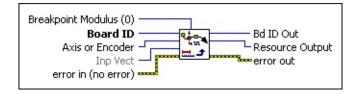
To avoid this problem, use MAX or the <u>Set u32</u> VI to change the breakpoint window. The breakpoint window is a buffer around the breakpoint position in which a breakpoint is not enabled while the current position remains inside the buffer.

Note All breakpoints can be affected by jitter in the motion system. For example, if you have breakpoint positions very close to each other, the 💦 jitter in the motion system could cause the position to change enough to reach the breakpoint when a breakpoint is not intended. Increase the value of the breakpoint window using <u>Set u32</u> to compensate for system jitter.

Load Breakpoint Modulus

Load the breakpoint modulus for a position breakpoint.

Device Compatibility



Details

Breakpoint Modulus (0) is the period, for periodic breakpoints, or the modulus, for modulus breakpoints. The range for period is 1 to 2^{15} –1. The range for modulus is 0 to 2^{31} –1. A modulus value of 0 makes a modulus breakpoint behave the same as an absolute breakpoint on 7330/40/90 controllers.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Axis or Encoder is the axis or encoder to control. You can load Breakpoint Position on encoders mapped to axes 1 through 6 or directly on encoders 0x21 through 0x24.
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the

This VI loads a position modulus/period, depending on the breakpoint mode. This value is double-buffered in the controller and does not take effect until you execute the **Enable Breakpoint Output** VI.

Modulo breakpoints are used in applications that require repetitive, equally spaced breakpoints. When using a breakpoint modulus, it is no longer necessary to load ever increasing (or decreasing) breakpoint positions. It is still necessary, however, to re-enable the breakpoint after each use.



Note For closed-loop stepper systems in which the encoder counts-per-N revolution differs from the steps-per-revolution, the breakpoint position is loaded in encoder counts.

When you enable a modulo breakpoint, two breakpoint positions, one in front and one behind the present encoder position, are enabled.

Note If you use modulus breakpoint or enable the same breakpoint repeatedly at low velocity (< 50 counts/sec), you may get a breakpoint output immediately after you re-enable it, because the breakpoint is reenabled before you have moved from the previous breakpoint position.



To avoid this problem, use MAX or the Set u32 VI to change the breakpoint window. The breakpoint window is a buffer around the breakpoint position in which a breakpoint is not enabled while the current position remains inside the buffer.

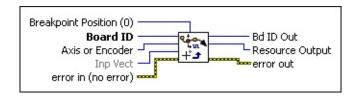
Example

An application requires breakpoints every 2,000 counts offset at -500 counts: ...–4,500, –2,500, –500, 1,500, 3,500, and so on. To accomplish this, you load a breakpoint position of -500 with the Load Breakpoint Position VI and a breakpoint modulus of 2,000. If the instantaneous encoder position is 2,210 counts when you execute the Enable Breakpoint Output VI (in modulo mode).

Load Breakpoint Position

Loads the breakpoint position for an axis or encoder in counts.

Device Compatibility



Details

parameters:

Breakpoint Position (0) is the breakpoint position in quadrature counts. Breakpoint positions can be anywhere within the 32-bit position range, – (2^{31}) to $+(2^{31})-1$. The default value is zero (0). **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller. **Axis or Encoder** is the axis or encoder to control. You can load **Breakpoint Position** on encoders mapped to axes 1 through 8 or directly on encoders 0x21 through 0x28. **Inp Vect** indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information. **error in (no error)** describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the

error in value to **error out**. The **error in** cluster contains the following

The Load Breakpoint Position VI loads the breakpoint position value for the axis or encoder specified. The breakpoint position is interpreted differently depending on the breakpoint mode and the encoder position when the breakpoint is enabled. You can specify position breakpoints as either absolute, relative, or with respect to a modulus range when the breakpoint is enabled. Breakpoint position is double-buffered and not actually used until you execute the Enable Breakpoint Output VI.

Note For modulo breakpoints, the magnitude of the breakpoint value must be less than the breakpoint modulus. If this range is exceeded, a modal error is generated when you execute the Enable Breakpoint Output VI.

Note To load breakpoint positions for buffered breakpoints, refer to the Write Buffer VI. The Load Breakpoint Position VI does not have any affect on buffered breakpoints.

When the breakpoint position is reached, a breakpoint event is generated and the associated high-speed breakpoint output immediately transitions.

High-speed breakpoint functionality is performed by the encoder resources themselves. When this VI is sent to an axis, it is actually being sent to the mapped encoder resource.

When the same breakpoint position is used on a repetitive basis, it is not necessary to reload the position each time. It is necessary, however, to re-enable the breakpoint after each use.

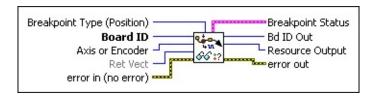
Note For closed-loop stepper systems in which the encoder counts-per-revolution differs from the steps-per-revolution, the captured position is in encoder counts.

Read Breakpoint Status

Reads the breakpoint status for all axes or encoders.

Note You cannot use this VI to read the status of buffered breakpoints. Use Check Buffer to get the status of the buffer.

Device Compatibility



Details

()	Breakpoint Type (Position) is a reserved input that must be set to Position Breakpoint (0).
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis or Encoder is the axis control or encoder control selector. For multi-axis status, use 0 (zero). For multi-encoder status, use 0x20.
	Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.
	error in (no error) describes error conditions that occur before this VI

runs. The default input of this cluster is no error. If an error already

This VI allows you to see if a breakpoint has occurred or is pending. When you enable a breakpoint, the corresponding status bit is reset to indicate that the breakpoint is pending. For single breakpoints, this status is set when a breakpoint occurs. Periodic breakpoints only set the breakpoint status after the first breakpoint has occurred, and the status remains asserted across the subsequent breakpoints. When the breakpoint position is reached, its status bit is set to True (1).

Example

Executing the Read Breakpoint Status VI with Axis or Encoder = 0x20 and Breakpoint Type = 0 returns Breakpoint Status, which corresponds to the following bitmap.

Axis/	Axis/	Axis/	Axis/	Axis/	Axis/	Axis/	Axis/
Encoder	Encoder	Encoder	Encoder	Encoder	Encode	r Encoder	Encoder
1	2	3	4	5	6	7	8
True	False	False	True	False	False	False	False

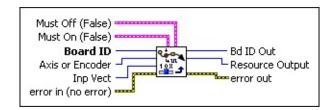
On encoders 1 and 4, breakpoints have occurred, but on encoders 2 and 3, breakpoints are pending or were never enabled.

Note Read Breakpoints Status returns the breakpoint status for either *all* axes or *all* encoders, based on which value is passed into Axis or Encoder.

Set Breakpoint Output MOMO

Sets the breakpoint outputs using the Must On/Must Off protocol.

Device Compatibility



Details

Must Off is the bitmap of breakpoint outputs to force Off:

- 1—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
- 2—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
- **3**—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
- 4—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
 - 5—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
 - **6**—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
 - 7—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).
 - **8**—If true, breakpoint output is forced Off. If false, breakpoint output is unchanged (default).

The Set Breakpoint Output MOMO VI directly controls the breakpoint outputs and sets them high or low. You can use this VI to set breakpoint outputs to a known state or to control them as general-purpose outputs in non-breakpoint applications. You can only control the output using this VI if the breakpoint is disabled.

Note The Set Breakpoint Output MOMO VI only affects the output state of the breakpoint. It does not change the breakpoint status.

Breakpoint functionality is performed by the encoder resources themselves. When this VI is sent to axes, the NI-Motion firmware consults the mapping of axes to encoders and actually sends the command to the mapped encoder resources. Breakpoints are only available on encoder resources. Refer to **Encoders** for encoder resource IDs.

Using the **Must On/Must Off** protocol allows you to set or reset individual breakpoint outputs without affecting the other breakpoint outputs. This gives you tri-state control over each output: On, Off, or Unchanged. A True in a location of the **Must On** bitmap sets the breakpoint high, while a True in the corresponding location of the **Must Off** bitmap resets the breakpoint low. A False in either bitmap has no effect, so leaving both the **Must On** and **Must Off** bits at zero is effectively a hold, and the state of the breakpoint output is unchanged. If you set both the **Must On** and **Must Off** bits to True, it is interpreted as a **Must On** condition and the breakpoint is set high.

Note If the breakpoint is enabled using the <u>Enable Breakpoint Output</u> VI, Nou cannot change the output value using Set Breakpoint Output MOMO. The command is ignored by the motion controller.

Note The first input vector source indicates the value for the **Must On** input and the second vector source indicates the value for the **Must Off** input. It is not a bitmap of the breakpoint output.



If you are doing onboard programming and are using **Inp Vect** to get the

High-Speed Capture VIs

High-speed capture inputs are an enhancement to the encoder FPGA and are available when the encoders are operating as axis feedback or as independent encoder resources. High-speed capture functionality is available on servo and closed-loop stepper axes.

You can use High-Speed Capture VIs to configure, enable, and read the status of high-speed inputs. Buffered high-speed capture is also available on the NI-7350 motion controllers. Buffered capture is a more advanced use of high-speed capture where the captured positions are stored in a previously-configured buffer. This helps the high-speed capture perform at higher capture frequencies. Refer to the <u>Buffered Operations Overview</u> for details on buffer operations in buffered high-speed capture for the 7350 controller.

The high-speed capture inputs also can function as latching general-purpose inputs. Configure the inputs as you would for high-speed capture operation, but ignore the captured position. You can then read the state of the latched inputs.

Click the icons for VI descriptions.



Configure High-Speed Capture

Enable High-Speed Capture

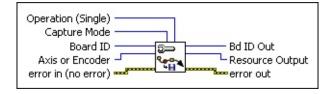
Read Captured Position

Read High-Speed Capture Status

Configure High-Speed Capture

Configures the high-speed capture input for the appropriate signal behavior.

Device Compatibility



Details Remarks

Operation (Single) selects between single and buffered high-speed capture.

Capture Mode selects among the following values to specify the manner in which the incoming high-speed capture signal is interpreted:

Non-Inverting Level

Inverting Level

Low-to-High Edge

High-to-Low Edge

Non-Inverting Digital Input

Inverting Digital Input

Non-Inverting Level, Inverting Level, Low-to-High Edge, and High-to-Low Edge modes define the state of the input signal that results in a high-speed capture event. Non-Inverting Level means that an active high input causes a capture to occur. Conversely, Inverting Level means that an active low input causes a capture to occur. Notice that if the signal is

After configuring the high-speed capture input, you enable the high-speed capture circuitry with the <u>Enable High-Speed Capture</u> VI. The <u>Read High-Speed Capture Status</u> VI tells you when the capture event has occurred. Finally, you read the captured value with the <u>Read Captured Position</u> VI.

High-speed capture inputs are an integral part of the encoder resources. You can execute this VI indirectly on axes or directly on encoder resources.

The controller captures the instantaneous encoder position when the input becomes active, as defined in the Capture Mode parameter.

High-speed capture functionality is available in both single and buffered mode. In single high-speed capture, the input trigger is configured to capture a single encoder position, and must be re-enabled after each capture. You can use buffered high-speed capture to support higher capture frequencies. Buffered high-speed capture allows captures positions to be stored into a buffer and automatically re-enabled by the controller. To configure a buffer for high-speed capture, refer to the Configure Buffer VI.

You can use a high-speed capture input as a general-purpose input and read its status with the Read High-Speed Capture Status VI. In Non-Inverting Level, Inverting Level, Low-to-High Edge, and High-to-Low Edge modes, the input has latching behavior, meaning that after the high-speed capture input is enabled, if the input signal ever becomes active (as defined by the Capture Mode parameter), the high-speed capture status is true until the input is reenabled. In Non-Inverting Digital Input and Inverting Digital Input modes, the inputs simply reflect the current state of the input signal. The input does not need to be enabled or reenabled at any time in these modes.

Note NI-Motion does not support breakpoint and high-speed capture functionality on an axis when the primary feedback for that axis is used as analog feedback.

Note When configured as a digital input, <u>Enable High-Speed Capture</u> and <u>Pand Captured Position have no effect</u>. The high speed capture input is

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- You can synchronize NI 73xx motion controllers with other National Instruments devices using the RTSI bus. Refer to the <u>Select Signal</u> VI for information about using the RTSI bus as the high-speed capture trigger.
- NI 73xx controllers do not support multiple simultaneous high-speed captures per axis.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• The NI SoftMotion Controller supports multiple simultaneous high-speed captures per axis.

To use this VI to configure the NI SoftMotion Controller for multiple high-speed captures on an axis, set the upper four bits of the resource ID to the following.

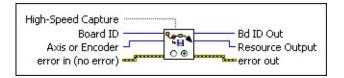
Type Value Primary HS Capture 0X0 Secondary HS Capture 0XA0

For example, to configure primary high-speed capture on axis 1, set the resource ID to 0X01. To configure secondary high-speed capture on axis

Enable High-Speed Capture

Enables or disables the specified highspeed capture input.

Device Compatibility



Details Remarks

TF	nigh-speed Capture indicates whether to enable or disable the high-
	speed capture input. Values are True or False.
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Axis or Encoder is the axis or encoder selector.
	error in (no error) describes error conditions that occur before this VI
	runs. The default input of this cluster is no error. If an error already
	occurred this VI returns the value of arrow in in arrow out. The VI runs

runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

The Enable High-Speed Capture VI enables high-speed capture inputs to capture instantaneous encoder position when an input becomes active. The position capture is implemented in the encoder resource to reduce capture latency to the sub-100 nanosecond range.

Note If your application uses the <u>Select Signal</u> VI to route the high-speed N capture source, you must call the Select Signal VI before enabling the high-speed capture.

High-speed capture functionality is performed by the encoder resources themselves. When this VI is sent to an axis, it is actually being sent to the mapped encoder resource.

The high-speed inputs have programmable polarity and edge behavior. You can set the polarity of the input with the Configure High-Speed Capture VI. You can determine the results of the high-speed capture from the Read High-Speed <u>Capture Status</u> and <u>Read Captured Position</u> VIs.

For single high-speed capture, you can enable only one high-speed capture per axis or encoder at a time. When an enabled event is captured, the high-speed capture is automatically disabled. You must explicitly re-enable high-speed capture to use it again. To disable a previously enabled operation, run this VI again with High-Speed Capture set to False.

For buffered high-speed capture, you need to enable the input trigger only once. Subsequent high-speed captures are automatically re-enabled by the controller. This VI has no effect when the input trigger is configured as a digital input.

If you are using buffered high-speed capture, ensure that the buffer is configured before you execute this VI. Refer to Configure Buffer for more information.

High-speed capture is useful in registration and synchronization applications. You can calculate subsequent moves relative to the captured position. Refer to

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- If your application includes a high-speed capture that takes place after a Find Reference move that finds an index, you must enable the high-speed capture *after* the find reference move. If your application includes a buffered high-speed capture, you must also configure the buffer after the find reference move.
- NI 73xx controllers do not support multiple simultaneous high-speed captures per axis.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• The NI SoftMotion Controller supports multiple simultaneous high-speed captures per axis.

To use this VI to configure the NI SoftMotion Controller to enable multiple high-speed captures on an axis, set the upper four bits of the resource ID to the following.

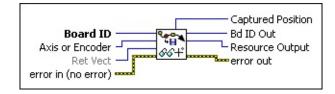
Type Value
Primary HS Capture 0X0
Secondary HS Capture 0XA0

For example, to enable the primary high-speed capture status on axis 1 set

Read Captured Position

Reads a captured position value from an axis or encoder.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Axis or Encoder is the axis or encoder to read.

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

The Read Captured Position VI returns the value in the high-speed capture register of the axis or encoder selected. This value was captured when an enabled high-speed capture input went active.

High-speed capture functionality is performed by the encoder resources themselves. When this VI is sent to an axis, the value returned is actually from the mapped encoder resource.

For buffered high-speed capture, the captured positions are stored in a buffer. Use <u>Read Buffer</u> to read the captured positions.

Refer to the Enable High-Speed Capture and Read High-Speed Capture Status VIs for more information about the high-speed capture inputs and typical applications.



Note For closed-loop stepper systems in which the encoder counts-per-N revolution differs from the steps-per-revolution, the captured position is in encoder counts.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

• NI 73xx controllers do not support multiple simultaneous high-speed captures per axis.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• The NI SoftMotion Controller supports multiple simultaneous high-speed captures per axis.

To use this VI to read multiple high-speed capture positions on an axis, set the upper four bits of the resource ID to the following.

Type Value Primary HS Capture 0X0 Secondary HS Capture 0XA0

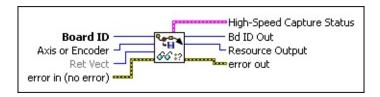
For example, to enable the primary high-speed capture status on axis 1, set the resource ID to 0X01. To enable the secondary high-speed capture status on axis 1, set the resource ID to 0XA1.

Read High-Speed Capture Status

Reads the high-speed position capture status for all axes or encoders.

Note You cannot use this VI to read the status of buffered high-speed captures. Use Check Buffer to get the status of the buffer.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
 Axis or Encoder is the axis control or encoder control selector. For multiaxis status, use 0 (zero). For multi-encoder status, use 0x20.
 Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.
 error in (no error) describes error conditions that occur before this VI

runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the

The Read High-Speed Capture Status VI allows you to see if a position capture has occurred or is pending. When configured as a single high-speed capture, the corresponding status bit is set when an input trigger becomes active, as configured in Configure High-Speed Capture. The status bit, along with the captured encoder position, is latched until the input trigger is re-enabled. When the position capture occurs, its status bit is set to True (1). Refer to the Read Captured Position VI for information about retrieving the captured position value.

Example

Executing the Read High-Speed Capture Status VI with Axis or Encoder = 0 returns High Speed Capture Status, as shown in the following table.

```
Axis 1 Axis 2 Axis 3 Axis 4 Axis 5 Axis 6 Axis 7 Axis 8 False True True False False False False False
```

On encoders mapped to axes 2 and 3, high-speed captures have occurred, but all other captures are pending or were never enabled.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The high-speed capture circuitry is also used during Find Index execution. When an index is found successfully, the capture status for the corresponding encoder and axis is set to True as a side effect.
- Executing the <u>Find Reference</u> Index VI automatically leaves the corresponding high-speed capture input disabled after the index is found.

Find Reference VIs

You can use these VIs to initialize the motion system and establish a repeatable reference position.

Typical closed-loop motion systems use incremental feedback to keep track of position. At power-up, this position is meaningless until a zero reference position is established. Open-loop stepper systems must also be initialized at power-up.

NI-Motion provides two built-in VIs, <u>Find Reference</u> and <u>Wait Reference</u>, to accomplish these tasks. Find Reference performs search sequences to find and stop on a specific edge of the home input, find the next instance of the encoder index, find the center of travel between forward and reverse limits, or find the forward/reverse limit. Wait Reference returns when an initiated find has completed. In this way a repeatable reference position that is accurate to one encoder count is established.

Click the icons for VI descriptions.



Find Reference

Check Reference

Wait Reference

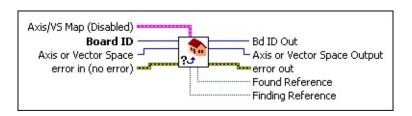
Read Reference Status

Load Reference Parameter

Get Reference Parameter

Check Reference

Checks the status of a search sequence initiated by <u>Find Reference</u>.



Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Axis/VS Map (Disabled) is the bitmap of axes or vector spaces (coordinate spaces) to use with this VI. It is only required when multiple axes or vector spaces are selected with the **Axis or Vector Space** parameter. Otherwise, this parameter is ignored.

When searching on multiple axes (**Axis or Vector Space** = Axis Control):

- **Axis 1**—If TRUE, check reference on the specified axis. If FALSE, do not check reference on the specified axis.
- **Axis 2**—If TRUE, check reference on the specified axis. If FALSE, do not check reference on the specified axis.
- **Axis 3**—If TRUE, check reference on the specified axis. If FALSE, do not check reference on the specified axis.
- **Axis 4**—If TRUE, check reference on the specified axis. If FALSE, do not check reference on the specified axis.
- **Axis** 5—If TRUE, check reference on the specified axis. If FALSE, do not check reference on the specified axis.

This VI returns whether the reference on the indicated axes is found. There is no need to specify what type of find to wait for because the controller knows what searches are being done on what axes. This VI waits for and reports on the success of the current or last executed find on an axis, vector space, or a given set of axes or vector spaces as a whole.

If a <u>Find Reference</u> is executed with the <u>Run Sequence</u> option the Check Reference VI does not require an Axis or Vector Space or Axis/VS Map. When waiting on a Run Sequence, the Check Reference VI returns the status of the entire sequence. The Found Reference Boolean returns True only if and when all of the operations in the sequence are successful.

Wait until the **Finding Reference** parameter is false before checking the status of the **Found Reference** parameter. The status of the **Found Reference** parameter is undefined when **Finding Reference** is true. The <u>Wait Reference</u> VI does this for you, but you cannot execute any other VIs while Wait Reference is running.

Example

You initiated a <u>Find Home</u> on Axis 1, and <u>Find Center</u> on Axis 2 and Axis 3, and you want to check if all axes have found their reference. Call Check Reference with the following parameters:

Axis or Vector Space = Axis Control (0) **Axis/VS Map** = 00001110 (axis 1, 2, & 3 enabled)

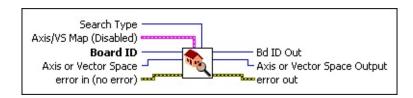
Find Reference

Executes a search operation to find a reference position:

- <u>home</u>
- forward limit
- <u>center</u>

- <u>index</u>
- reverse limit
- run sequence

Device Compatibility



Details Remarks

Search Type is the type of search executed by the Find Reference VI.

	Value	Definition	Description
	0	Find Home	Executes a Find Home on the given set of axes.
	1	Find Index	Executes a Find Index on the given set of axes.
<u> </u>	2	Find Center	Finds the center of travel on a given set of axes.
	3	Find Forward Limit	Finds the forward limit on a given set of axes.
	4	Find Reverse Limit	Finds the reverse limit on a given set of axes.
	5	Run Sequence	Runs the loaded Find Sequence.
Board ID is a unique number assigned by Measurement & Automation			

This VI returns immediately after starting the given find operation. Use this VI in conjunction with <u>Wait Reference</u> to determine when the find operation completes and if the find was successful.

You can manually stop a search operation using MAX or by calling the Stop Motion VI. When encountering a limit or home input, a search operation performs a halt stop and then continues based on the **Search Type**. An unexpected limit condition during a find stops the sequence and generates a modal error. Refer to Errors and Error Handling for information about errors.



Note You must set unused limit and home inputs to their inactive state using MAX or with the <u>Enable Limits</u> and <u>Enable Home Inputs</u> VIs.



Note Forward is defined as the direction of increasing position. The N Forward and Reverse Limits must be located at the proper ends of travel for Find Reference to function properly.

You can access settings for Find Reference through MAX, Load Reference Parameter and Get Reference Parameter. For a detailed description on how these settings affect a given **Search Type**, refer to the section on each type. Most of these settings are on a per-**Search Type** basis.

Note If you start a coordinate (vector) space, Find Reference actually starts the axes individually, so it is not a coordinated vector move.

Search Types

Find Home

Find Index

Find Center

Find Forward and Reverse Limits

Run Sequence

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

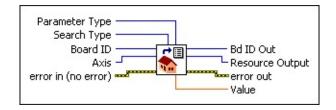
The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- If your application includes a high-speed capture that takes place after a <u>Find Reference</u> move that finds an index, you must enable the high-speed capture *after* the find reference move. If your application includes a buffered high-speed capture, you must also <u>configure the buffer</u> after the find reference move.
- When performing a reset position as part of a Find Reference operation, add at least one of the following to the move complete criteria to ensure that the motor is not moving when the reset position executes:
 - Add a delay after the profile complete status is true to allow sufficient time between the output and reset position for the motor to stop. Use <u>Configure Move Complete Criteria</u> to configure a delay.
 - Use the in-position signal as a criteria for move complete. The NI PCI-7390 controller uses this signal by default. To add this signal to the move complete criteria for controllers other than the NI PCI-7390, use Configure Motion I/O Map to map the signal to a pin and then use Write Trajectory Data to add in-position to the move complete criteria. In addition, configure the in-position range on the drive to a small enough number to ensure that the target position is reached. Refer to your drive documentation for information about configuring the in-position range on the drive.

Get Reference Parameter

Gets the value for the specified find reference parameter.

Device Compatibility



Details Remarks

Parameter Type is the type of parameter to get. Valid values are:

Initial Search Direction

Final Approach Direction

Edge To Stop On

Smart Enable

Enable Reset Position

Offset Position

Primary Reset Position

Secondary Reset Position

Approach Velocity Percent

Sequence Search Order

Enable Search Distance

Search Distance

Phase A Reference State

Phase B Reference State

Search Type is the type of reference to get. Possible Search Type values

Use this VI to get the parameters for <u>Find Reference</u>.

- **Initial Search Direction**—If 1, search reverse. If 0 (default), search forward.
- **Final Approach Direction**—1 is true and 0 is false (default). The following table lists the behavior of the Final Approach Direction parameter for each type of find:

Find Type	Final Approach Direction is True (1)	Final Approach Direction is False (0)
Find Home	reverse approach	forward approach
Find Center Find Forward Limit Find Reverse Limit	opposite to direction of travel into limit	same as the direction of travel into limit
Find Index	N/A	N/A



Note For Find Center, this value refers to how the controller behaves when approaching and marking the position of the limits to find their center.

- **Edge to Stop On**—Applies only to Find Home. If 1, reverse edge. If 0 (default), forward edge.
- **Approach Velocity Percent**—The percent of the loaded velocity that a given axis uses to perform final adjustments. Refer to the *Remarks* section for valid values.

Find Index uses this percent velocity throughout the entire search routine.

Smart Enable—If 1, appropriately enables or disables the limits and home input before executing a find. The limits and home input are returned to their original state after the find operation is complete. To determine what state **Smart Enable** sets the limits or home input to, refer

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The resolution of **Approach Velocity Percent** is approximately 0.4%
- For the NI 73xx controllers, **Approach Velocity Percent** is a single-precision floating-point value from 0.4% to 150%. This value directly scales the programmed velocity. The default value is 20%.

NI SoftMotion Controller Considerations

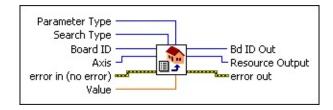
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- Secondary Reset Position is not supported by the NI SoftMotion Controller.
- Phase A Reference State and Phase B Reference State are not supported on CANopen devices.
- Primary Reset Position must be set to 0 when you are using the NI SoftMotion Controller for the Copley Controls CANopen drives, Accelnet and Xenus.
- The NI SoftMotion Controller supports the full 64-bit floating point range for the **Approach Velocity Percent** parameter.
- The NI SoftMotion Controller requires **Search Distance**, and thus, **Enable Search Distance** is always 1 (true).

Load Reference Parameter

Loads the value for the specified find reference parameter.

Device Compatibility



Details Remarks

Parameter Type is the type of parameter to load. Valid values are:

- Initial Search Direction
- Final Approach Direction
- Edge To Stop On
- Smart Enable
- Enable Reset Position
- Offset Position

(1)

- Primary Reset Position
- Secondary Reset Position
- Approach Velocity Percent
- Sequence Search Order
- Enable Search Distance
- Search Distance
- Phase A Reference State
- Phase B Reference State

Search Type is the type of reference to load. Possible **Search Type** values are <u>Find Home</u>, <u>Find Index</u>, <u>Find Center</u>, and <u>Find Forward & Reverse</u>

Use this VI to load the parameters for <u>Find Reference</u>. The following are valid values for each **Parameter Type**:

- **Initial Search Direction**—Enter 1 to search in the reverse direction. Enter 0 (default) to search in the forward direction.
- **Final Approach Direction**—Enter 1 for true and 0 for false (default). The following table lists the behavior of the Final Approach Direction parameter for each type of find:

Find Type	Final Approach Direction is True (1)	Final Approach Direction is False (0)
Find Home	reverse approach	forward approach
Find Center Find Forward Limit Find Reverse Limit	opposite to direction of travel into limit	same as the direction of travel into limit
Find Index	N/A	N/A



Note For Find Center, this value refers to how the controller behaves $extstyle ilde{N}$ when approaching and marking the position of the limits to find their center.

- **Edge to Stop On**—Applies only to Find Home. Enter 1 to stop on the reverse edge. Enter 0 (default) to stop on the forward edge.
- Approach Velocity Percent—The percent of the loaded velocity that a given axis uses to perform final adjustments. Refer to the Remarks section for valid values.
 - Find Index uses this percent velocity throughout the entire search routine.
- **Smart Enable**—A value of 1 (true) appropriately enables or disables the limits and home input before executing a find. The limits and home input are returned to their original state after the find operation is complete. To

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The resolution of **Approach Velocity Percent** is approximately 0.4%
- For the NI 73xx controllers, **Approach Velocity Percent** is a single-precision floating-point value from 0.4% to 150%. This value directly scales the programmed velocity. The default value is 20%.

NI SoftMotion Controller Considerations

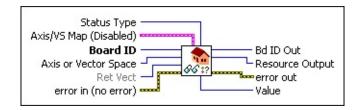
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- Secondary Reset Position is not supported by the NI SoftMotion Controller.
- Phase A Reference State and Phase B Reference State are not supported on CANopen devices.
- Primary Reset Position must be set to 0 when you are using the NI SoftMotion Controller for the Copley Controls CANopen drives, Accelnet and Xenus.
- The NI SoftMotion Controller supports the full 64-bit floating point range for the **Approach Velocity Percent** parameter.
- The NI SoftMotion Controller requires **Search Distance**, and thus, **Enable Search Distance** is always 1 (true).

Read Reference Status

Reads the currently selected reference status for the given set of axes or coordinate (vector) space.

Device Compatibility



Details

Status Type is the selector for the type of reference status to read.

You can choose from the following status types:

	Status Type	Description
	Home Found	returns True if the last <u>Find Home</u> executed successfully
	Index Found	returns True if the last <u>Find Index</u> executed successfully
•	Center Found	returns True if the last <u>Find Center</u> executed successfully
	Forward Limit Found	returns True if the last <u>Find Forward Limit</u> executed successfully
	Reverse Limit Found	returns True if the last <u>Find Reverse Limit</u> executed successfully
	Reference Found	returns True if the last <u>Find Reference</u> executed successfully

This VI reads the currently selected reference status for the given set of axes or coordinate (vector) space.

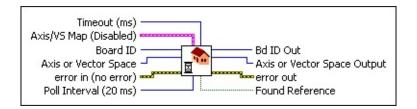
If multiple axes in a coordinate space are selected using the axis bitmap, all axes must have successfully found their last reference for Home Found, Index Found, Center Found, Forward Limit Found, Reverse Limit Found, and Reference Found to be true.

If any of the selected axes are currently executing a Find Reference, Finding Reference is true. Wait until Finding Reference is false before checking the status of any other attribute. The status of the Home Found, Index Found, Center Found, Forward Limit Found, Reverse Limit Found, and Reference Found status types are undefined when Finding Reference is true. The Wait Reference VI does this for you, but you cannot execute any other VIs while Wait Reference is running.

Wait Reference

Waits for a search sequence initiated by <u>Find</u> <u>Reference</u> to complete and returns the status. Wait Reference also can be used to query the status of a search.

Device Compatibility



Details

Timeout (ms) specifies how long, in milliseconds, the Wait Reference call waits before returning. If the timeout is reached before the search completes the Wait Reference Returns. In this manner, Wait Reference can be used to check the status of a search by passing a timeout of 0. By default, the timeout is 4294967295 or 0xFFFFFFFF. This value is handled differently, signifying an infinite wait time.

Axis/VS Map (Disabled) is the bitmap of axes or vector spaces (coordinate spaces) to use with this VI. It is only required when multiple axes or vector spaces are selected with the **Axis or Vector Space** parameter. Otherwise, this parameter is ignored.

When searching on multiple axes (**Axis or Vector Space** = Axis Control):

- **Axis 1**—If TRUE, wait on the specified axis. If FALSE, do not wait on the specified axis.
- **Axis 2**—If TRUE, wait on the specified axis. If FALSE, do not wait on the specified axis.
- **Axis 3**—If TRUE, wait on the specified axis. If FALSE, do not wait on the specified axis.

This VI returns when the find on the indicated axes finishes or the timeout is reached, whichever occurs first. There is no need to specify what type of find to wait for because the controller knows what searches are being done on what axes. This VI waits for and reports on the success of the current or last executed find on an axis, vector space, or a given set of axes or vector spaces as a whole.

If a <u>Find Reference</u> is being executed with the Run Sequence option the Wait Reference VI does not require an Axis or Vector Space or Axis/VS Map. When waiting on a Run Sequence, the Wait Reference VI returns when the entire sequence has finished. The Found Reference Boolean returns True only if all of the operations in the sequence were successful.

Note You cannot execute any other processes while the Wait Reference VI is running. Use <u>Check Reference</u> or <u>Read Reference Status</u> if you want to run other processes simultaneously.

Example

You initiated a <u>Find Home</u> on Axis 1, and <u>Find Center</u> on Axis 2 and Axis 3, and you want to wait for all find sequences to finish before proceeding with the rest of the program. Simply call Wait Reference with the following parameters:

Axis or Vector Space set to Axis Control (0) **Axis Map** set to 00001110 (axis 1, 2, & 3 enabled) **Timeout** is Unwired

Analog & Digital I/O VIs

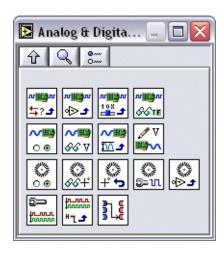
You can use the analog and digital I/O VIs to control the general-purpose analog and digital I/O resources on the NI motion controller. These resources include up to 32 bits of general-purpose digital I/O, PWM outputs, RTSI lines, and any extra encoders, ADC channels, and DAC outputs that are not mapped to an axis.

The 32 bits of digital I/O are available on the Digital I/O Connector. These bits are organized into 8-bit ports that you can configure as inputs or outputs on a port-wise basis, or on a bitwise basis. Each bit has individually programmable polarity that you can configure as active low or active high. You can use the general-purpose digital I/O for system integration applications including operator panel switch inputs and outputs, relay and solenoid activation, trigger I/O between other controllers and/or instruments in the system, and so on.

You can use encoders, ADC channels, and DAC outputs that are not mapped to an axis for general-purpose I/O. Typical uses for encoder inputs include velocity monitoring, masters for master-slave gearing, and digital potentiometer applications.

You can use unused ADC inputs and DAC outputs for any analog I/O that is within their specifications. Typical analog input applications include analog joysticks, potentiometers, force, pressure, level and strain sensors, and so on. Analog output applications vary from heater element control to laser intensity modulation.

Click the icons for VI descriptions.



Set I/O Port Direction Enable Encoders

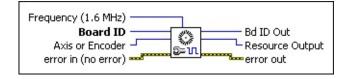
Set I/O Port PolarityRead Encoder PositionSet I/O Port MOMOReset Encoder PositionRead I/O PortConfigure Encoder FilterEnable ADCsConfigure Encoder PolarityRead ADCConfigure PWM OutputSet ADC RangeLoad PWM Duty Cycle

<u>Load DAC</u> <u>Select Signal</u>

Configure Encoder Filter

Selects the maximum count frequency for an encoder channel by configuring its digital filter.

Device Compatibility



Details

Frequency (1.6 MHz) selects the maximum count frequency for the specified encoder.

Frequency Value	Maximum Count Frequency
0	25.6 MHz
1	12.8 MHz
2	6.4 MHz
3	3.2 MHz
4	1.6 MHz (default)
5	800 kHz
6	400 kHz
7	200 kHz
8	100 kHz
9	50 kHz
10	25 kHz
11	Disabled

Board ID is a unique number assigned by Measurement & Automation

Setting the maximum allowable count frequency for an encoder is useful for reducing the effect of noise on the encoder lines. Noise on the encoder lines can be interpreted as extra encoder counts. By setting the frequency to the lowest possible setting required for the motion application, you can ensure the highest degree of accuracy in positioning. In choosing the appropriate value, you must take into account the counts per revolution of the encoder and the maximum velocity for the axis in question.

For example, with a 20,000 counts per revolution encoder and a maximum velocity of 3,000 RPM (50 revolutions per second), the encoder signal could be as high as 1,000,000 counts per second. A frequency value of 4, which would correspond to a maximum count frequency of 1.6 MHz would be appropriate in this case.

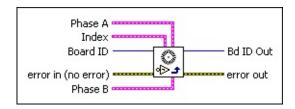
If you never call this VI, a default value of 4 (1.6 MHz) is used.

Tip When the encoder filter is changed, reset the position to prevent an erroneous count.

Configure Encoder Polarity

Configures the encoder **Phase A**, **Phase B**, and **Index** line polarities.

Device Compatibility



Details

Phase A sets the polarity for encoder **Phase A** to active low or active high. **Encoder 1**—If true, active high (default). If false, active low. **Encoder 2**—If true, active high (default). If false, active low. **Encoder 3**—If true, active high (default). If false, active low. **Encoder 4**—If true, active high (default). If false, active low. **Encoder** 5—If true, active high (default). If false, active low. **Encoder 6**—If true, active high (default). If false, active low. **Encoder** 7—If true, active high (default). If false, active low. **Encoder 8**—If true, active high (default). If false, active low. **Index** sets the polarity for the encoder **Index** to active low or active high. **Encoder 1**—If true, active high (default). If false, active low. **Encoder 2**—If true, active high (default). If false, active low. **Encoder 3**—If true, active high (default). If false, active low. **Encoder 4**—If true, active high (default). If false, active low. **Encoder** 5—If true, active high (default). If false, active low. **Encoder 6**—If true, active high (default). If false, active low.

Encoder 7—If true, active high (default). If false, active low.

Encoder 8—If true, active high (default). If false, active low.

The Configure Encoder Polarity VI configures the polarities for the encoder **Phase A**, **Phase B**, and **Index** lines. Configure encoder polarities before calling the Enable Encoders VI.

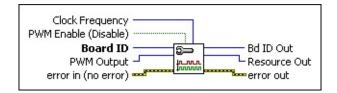
You can configure the encoder polarity as active low or active high. When configured as active low, the input is active when there is a low signal on the input pin. Conversely, active high means that the input is active when there is a high signal on the input pin.

Note Setting **Phase A** and **Phase B** incorrectly causes the feedback to appear to move in the opposite direction. For example, if **Phase A** or **Phase B** is the wrong polarity when moving in the forward direction, the position appears to move in the reverse direction. The opposite occurs when moving in the reverse direction. If both are set incorrectly, motion occurs in the correct direction but the index might not work correctly.

Configure PWM Output

Enables and disables PWM outputs, and sets the PWM clock frequency.

Device Compatibility



Details

Clock Frequency specifies the clock frequency for the PWM output.

The base clock frequency for the PWM outputs is 24.81 MHz on the 7340 and 7330 controllers, 24.58 MHz on the 7350 controller, and 20.48 on 7334/42/44 controllers. This base clock frequency is divided down depending on the clock value selected.

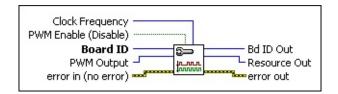
The following table describes the PWM clock frequency settings:

Clock Divider Value	Clock Value*		7330/40 Clock Frequency	7350 Clock Frequency
512	0	40.00 kHz	48.46 kHz	48.01 kHz
1K	1	20.00 kHz	24.23 kHz	24.00 kHz
2K	2	10.00 kHz	12.11 kHz	12.00 kHz
4K	3	5.00 kHz	6.06 kHz	6.00 kHz
8K	4	2.50 kHz	3.03 kHz	3.00 kHz
16K	5	1.25 kHz	1.51 kHz	1.50 kHz
33K	6	625.00 Hz	757 14 Hz	750 09 Hz

Configure PWM Output

Enables and disables PWM outputs, and sets the PWM clock frequency.

Device Compatibility



Details

Clock Frequency specifies the clock frequency for the PWM output.

The base clock frequency for the PWM outputs is 24.81 MHz on the 7340 and 7330 controllers, 24.58 MHz on the 7350 controller, and 20.48 on 7334/42/44 controllers. This base clock frequency is divided down depending on the clock value selected.

The following table describes the PWM clock frequency settings:

Clock Divider Value	Clock Value*		7330/40 Clock Frequency	7350 Clock Frequency
512	0	40.00 kHz	48.46 kHz	48.01 kHz
1K	1	20.00 kHz	24.23 kHz	24.00 kHz
2K	2	10.00 kHz	12.11 kHz	12.00 kHz
4K	3	5.00 kHz	6.06 kHz	6.00 kHz
8K	4	2.50 kHz	3.03 kHz	3.00 kHz
16K	5	1.25 kHz	1.51 kHz	1.50 kHz
33K	6	625 00 Hz	757 14 Hz	750 09 Hz

The PWM outputs on the NI motion controller are digital pulse-train outputs that have a frequency specified by the clock parameter of this VI and a duty cycle specified by the <u>Load PWM Duty Cycle</u> VI. These outputs can be used to control devices that require a PWM input, such as a laser whose intensity is controlled by a PWM signal, or can be used to generate isolated analog outputs by passing the PWM output through an optocoupler, and then filtering the digital pulse train to produce an analog output voltage.

When you configure a PWM output, the clock frequency applies to both PWM outputs. If you configure one PWM output for a clock value of 3, and then the second PWM output for a clock value of 4, the value of 4 applies to both PWM outputs.

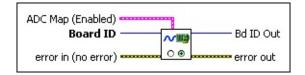
The only exception is when the clock settings for the two PWM outputs are 0 and 8, 1 and 9, 2 and 10, and so on, in which case each output has a different frequency. This is because clock values 0–7 and 8–15 are paired. Clock values 0 and 8 use the common time base of 10.24 MHz (12.8 MHz on the 7350). The frequency of clock value 0 is found by dividing the time base by 256, while clock value 8 is found by dividing the time base by 32768. If you have an NI 7344 controller and you switch channel B to clock value 10, the time base changes to 2.56 MHz. This change also switches channel A to 2.56/256 = 10 kHz. Therefore, if the value in one set changes, it switches the other channel to the pairing clock value. If the two values are in the same group, the two channels have the same frequency.

To use an external clock (clock values of 7 or 15), connect the external clock signal to the PCLK input on the Digital I/O connector.

Enable ADCs

Enables one or more of the unmapped ADC channels.

Device Compatibility



Details

- **ADC Map (Enabled)** is the bitmap of ADC channels to enable:
- If **ADC Map 0** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
- If **ADC Map 1** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
- If **ADC Map 2** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
- If **ADC Map 3** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
 - If **ADC Map 4** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
 - If **ADC Map 5** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
 - If **ADC Map 6** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
 - If **ADC Map 7** is TRUE (default) the ADC channel is enabled. If it is FALSE the ADC channel is disabled.
- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
 - **error in (no error)** describes error conditions that occur before this VI

The Enable ADCs VI enables one or more independent ADC channels for use as general-purpose analog inputs. The controller returns an error if you attempt to enable or disable an ADC being used as an axis feedback when the axis is enabled. These feedback channels are automatically enabled/disabled when you enable or disable their corresponding axis with the Enable Axes VI. You must first disable the axis using analog feedback, then enable or disable ADCs.

The NI-Motion Analog-to-Digital Converter (ADC) multiplexes between channels with a scan rate of approximately 40 µs per channel for the 7340 motion controller, and 25 µs per channel for the 7350 motion controller. Therefore, the time between samples for a specific ADC channel is as follows:

7330/40: ADC sample time = 40 μ s/channel × (number of enabled channels)

7350:

ADC sample time = $25 \mu s$ /channel × (number of enabled channels)

By default, all channels are enabled at power-up. You must disable unused channels to increase the scan rate and decrease the sample time.

The 40 μ s/channel scan rate is fast enough to support analog feedback at the fastest PID update rates as long as no additional ADC channels are enabled.

Example

To enable ADC channels 1, 3, 5, and 7 on the 7350 motion controller, call the Enable ADCs VI with **AxisMap**, as follows.

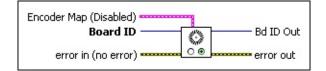
0 (ADC	1 (ADC	2 (ADC	3 (ADC	4 (ADC	5 (ADC	6 (ADC	7 (ADC
1)	2)	3)	4)	5)	6)	7)	8)
True	False	True	False	True	False	True	False

Under normal conditions, because ADC channels 2, 4, 6, and 8 are set to False, they are disabled when you execute this VI. However, if ADC channel 2 is

Enable Encoders

Enables one or more of the unmapped encoder resources.

Device Compatibility



Details

Encoder Map (Disabled) is the bitmap of encoder resources to enable.

- **Encoder 1**—If true, the encoder is enabled. If false, the encoder is disabled (default).
- **Encoder 2**—If true, the encoder is enabled. If false, the encoder is disabled (default).
- **Encoder 3**—If true, the encoder is enabled. If false, the encoder is disabled (default).
- **Encoder 4**—If true, the encoder is enabled. If false, the encoder is disabled (default).
 - **Encoder 5**—If true, the encoder is enabled. If false, the encoder is disabled (default).
 - **Encoder 6**—If true, the encoder is enabled. If false, the encoder is disabled (default).
 - **Encoder 7**—If true, the encoder is enabled. If false, the encoder is disabled (default).
 - **Encoder 8**—If true, the encoder is enabled. If false, the encoder is disabled (default).
- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI

The Enable Encoders VI enables one or more independent encoder channels for a general-purpose encoder inputs. It has no effect on encoders that are mapped to a being used for axis feedback. These feedback encoders are automatically enabled when their corresponding axis is enabled or disabled with the <u>Enable Axes</u> VI. B locations corresponding to mapped encoders are ignored, as are encoder resource not exist. For example, encoder resources 5 through 8 are ignored for controllers support a maximum of four axes.

Typical uses for independent encoder inputs include velocity monitoring, master master-slave gearing, and digital potentiometer applications.

Example

To enable encoders 3 and 4 on the motion controller, call the Enable Encoders V. following Encoder Map.

Encoder 1 Encoder 2 Encoder 3 Encoder 4 Encoder 5 Encoder 6 Encoder 7 I False False True False False False I

Normally, because encoders 1, 2, 5, and 6 are set to False, they is disabled when execute this VI. However, if encoder 2 is already being used as feedback for axis axis 1 is not being used, the disable False for Encoder 2 is ignored resulting in th following enabled encoders.

Encoder 1 Encoder 2 Encoder 3 Encoder 4 Encoder 5 Encoder 6 Encoder 7 I False True True False False False I

There is a limit on the number of enabled encoders supportable at the faster upda Attempting to enable too many encoders generates an error. Refer to the Enable for more information about update rate limitations.

This VI

e Encoders VI enables one or more independent encoder channels for use as rpose encoder inputs. It has no effect on encoders that are mapped to axes and for axis feedback. These feedback encoders are automatically enabled/disabled corresponding axis is enabled or disabled with the Enable Axes VI. Bit orresponding to mapped encoders are ignored, as are encoder resources that do or example, encoder resources 5 through 8 are ignored for controllers that naximum of four axes.

es for independent encoder inputs include velocity monitoring, masters for ve gearing, and digital potentiometer applications.

encoders 3 and 4 on the motion controller, call the Enable Encoders VI with the Encoder Map.

Encoder 2 Encoder 3 Encoder 4 Encoder 5 Encoder 6 Encoder 7 Encoder 8 False True False False False False

because encoders 1, 2, 5, and 6 are set to False, they is disabled when you is VI. However, if encoder 2 is already being used as feedback for axis 2 and ot being used, the disable False for Encoder 2 is ignored resulting in the enabled encoders.

Encoder 2 Encoder 3 Encoder 4 Encoder 5 Encoder 6 Encoder 7 Encoder 8 True True False False False False

limit on the number of enabled encoders supportable at the faster update rates. g to enable too many encoders generates an error. Refer to the Enable Axes VI Iformation about update rate limitations.

Load DAC

Loads an output value to an unmapped DAC resource.

DAC Value (0)

Board ID

DAC

Inp Vect
error in (no error)

Device Compatibility

Details Remarks

DAC Value (0) is the value sent to the DAC. The parameter range is -32,768 to +32,767, corresponding to the full ± 10 V output range.

Note DAC torque limits and offsets do not apply when directly loading a DAC.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

DAC is the DAC to control.

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code

The Load DAC VI is used to send a value directly to an unmapped DAC resource. DACs not mapped as servo axis outputs are available for general-purpose analog output applications.

Caution	You must not execute this VI on a DAC mapped to an axis. Doing
so causes	the DAC output to glitch momentarily before returning to axis
control.	

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

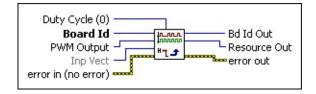
The following includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

If the axis is enabled, the control loop has ownership of the drive command output, and the value that this VI sends is ignored.

Load PWM Duty Cycle

Sets the duty cycle for a PWM output.

Device Compatibility



Details

parameters:

Duty Cycle (0) is a value between 0 and 255 that specifies the amount of time the PWM output is high.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
PWM Output selects the PWM Output to configure (1 or 2).
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

The Duty Cycle determines the amount of time the PWM output is high. A Duty Cycle of 0 corresponds to a 0 V output, and a Duty Cycle of 255 corresponds to a pulse train that is high for 255/256 = 99.6% of the time. Use the Configure PWM Output VI to set the frequency of the PWM output signal.

You can set the Duty Cycle before or after configuring a PWM output. By default, the Duty Cycle is 0, so if you call the Configure PWM Output VI to configure a PWM output, the output is low until you set the Duty Cycle differently. If you set the Duty Cycle first, the PWM output reflects this Duty Cycle immediately after calling the Configure PWM Output VI.

Note If you are doing onboard programming and are using Inp Vect to get the data this VI needs, remember that this VI reads the variables starting at the memory address pointed to by Inp Vect in the following order: PWM Output, Duty Cycle.

Read ADCs

Reads the converted value from an ADC input channel.

Board ID

ADC

Ret Vect
error in (no error)

Analog Value

Bd ID Out
Resource Output
error out

Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
ADC is the Analog-to-Digital Converter channel to read. Valid ADC

ADC is the Analog-to-Digital Converter channel to read. Valid ADC resources are 0x51 (81) through 0x5F (95).

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

The Read ADC VI returns the converted voltage from any of the analog input channels. You can only read values from channels that have been either directly enabled by the <u>Enable ADCs</u> VI or automatically enabled by being mapped to an enabled axis.

For an ADC channel mapped to an axis, this VI returns the actual ADC value. In contrast, the <u>Read Position</u> VI executed on the owner axis returns an ADC value that has been offset by a reset value stored when the <u>Reset Position</u> VI was executed. ADC channels are never internally reset so their DC values are preserved.

Read I/O Port

Reads the logical state of the bits in an I/O port.

Board ID
Port Data
Bd ID Out
Resource Out
Retn Vect
error in (no error)

Device Compatibility

Details Remarks

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Port (1) is the general-purpose I/O port (1–8) or RTSI software port (9) to read.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

The Read I/O Port VI reads the logical state of the bits in the general-purpose I/O port selected. You can execute this VI at any time to monitor the signals connected to an input port. Reads of ports configured as outputs return the last value written to the port with the <u>Set I/O Port MOMO</u> VI.

Note This VI reads the logical state (On or Off, True or False) of the bits in a port. The polarity of the bits in the port determines whether an On state is active high or active low. Refer to the <u>Set I/O Port Polarity</u> VI for more information.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following includes considerations you must make when you are using this VI with a 73xx controller:

- The PCI-7390 has dedicated direction I/O ports. Refer to <u>73xx Controller General-Purpose I/O Port IDs</u> for the appropriate port number to use for **Port**.
 - For example, to read the state of input port 3, **Port** is 0x03 (3). To read the value previously set on output port 2, **Port** is 0x82 (130)
- When reading the RTSI port, the value read is the latched data, so you can
 detect active pulses on the RTSI bus. After reading the latched data value,
 the VI resets the latch. Use the Set I/O Port Polarity VI to specify the
 polarity—and therefore the active state—for latching.

NI SoftMotion Controller Considerations

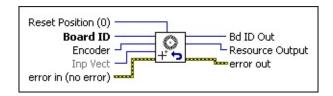
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The NI SoftMotion Controller has dedicated direction I/O ports. Refer to <u>NI SoftMotion Controller General-Purpose I/O Port IDs</u> for the appropriate port number to use for **Port**.
 - For example, to read the state of output port 2, **Port** is 0x82 (130). To read the state of input port 3, **Port** is 0x03 (3).
- The RTSI port is not supported by the NI SoftMotion Controller.

Reset Encoder Position

Resets the position of an unmapped encoder to the specified value.

Device Compatibility



Details

Reset Position (0) is the reset value for the encoder resource. You can reset position to any value in the total position range of -(2^{31}) to +(2^{31} – 1). **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller. **Encoder** is encoder to reset. **Inp Vect** indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information. error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

The Reset Encoder Position VI resets the position of the selected encoder. You can reset position to zero or to any value in the 32-bit position range. Normally, this VI is only used on independent encoders that are not mapped to axes. For encoders mapped to axes, use the <u>Reset Position VI</u> instead.

Note Attempting to reset an encoder that is mapped to an axis generates an error.



Enabled breakpoints are also automatically disabled when you execute Reset Encoder Position on the corresponding axis.

Encoder position can be reset at any time. However, it is recommended that you reset position only while the encoder is stopped. An encoder reset while it is moving does not have a repeatable reference position.



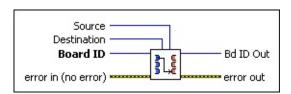
Note Non-zero reset values are useful for defining a position reference offset.

Select Signal

Specifies the source and destination for various motion signals:

- trigger inputs
- high-speed capture circuits
- breakpoint outputs
- RTSI lines
- RTSI software ports

Device Compatibility



Details

Source is the source of the signal to route to **Destination**.

For a **Destination** value of High-speed Capture[1..8], the valid Source values are as follows.

Source	Comments
RTSI[07]	RTSI lines 0 through 7
PXI Star Trigger	PXI star trigger line
Trigger Input	Trigger input for the corresponding axis

When the destination is RTSI[0..7] or PXI Star Trigger, the motion controller drives the RTSI line as an output. When the destination is High-speed Capture[1..8], the RTSI line serves as an input for the high-speed capture circuitry. This VI is used to set the direction of the RTSI lines. The RTSI lines can always be read using the Read I/O Port VI, regardless of the way they are currently configured.

To manually set the state of the RTSI lines, set the Source for the RTSI line to be **RTSI Software Port**. Use the <u>Set I/O Port MOMO</u> VI to set the state of the lines.

The signal seen on the RTSI 4 pin is a high pulse of 120 to 150 ns duration. The action specified in the Enable Breakpoint Output VI only applies to the breakpoint output pin on the motion I/O connector, not to RTSI pins.

Example 1

To use the signal coming in on RTSI pin 3 to trigger the high-speed capture on encoder/axis 1, connect Destination to High-Speed Capture 1 and Source to RTSI 3.

The polarity of the high-speed capture input is specified by the <u>Configure High-Speed Capture</u> VI.

Example 2

To output the breakpoint signal for axis 2 on RTSI pin 4, connect Destination to RTSI 4 and Source to Breakpoint 2.

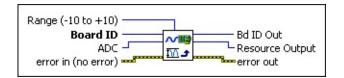
Example 3

To drive RTSI pin 5 with the corresponding bit (bit 5) of the RTSI software port, connect Destination to RTSI 5 and Source to RTSI Software Port. To set the state of the RTSI software port, use the <u>Set I/O Port MOMO</u> VI.

Set ADC Range

Sets the voltage range for the analog to digital converters, on a per-channel basis.

Device Compatibility



Details

Range (–10 to +10) specifies the input voltage range over which the ADC converts input voltages to digital values. Voltages outside of the range clamps at the extremes, which are -2048 or +2047 (7330/40) and -32,768 to +32,767 (7350) for the -5 to +5 V and -10 to +10 V ranges, and 0 to +4,096 (7330/40) and 0 to +65,535 (7350) for the 0 to +5 V and 0 to +10 V ranges. You can choose from the following values for the range:

0

Binary Values

Range Values	7350	7330/40
0 to 5	0 to +65,535	0 to +4,095
–5 to +5	-32,768 to +32,767	7-2,048 to +2,047
0 to +10	0 to +65,535	0 to +4,095
-10 to +10 (default)	-32,768 to +32,767	7-2,048 to +2,047

- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
- **ADC** is the analog-to-digital converter channel to configure. Valid ADC resources are 0x51 through 0x5F.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs

If you do not call this VI, the range defaults to -10 to +10 V. If you know that the input voltage falls within a more restrictive range, you can effectively increase the resolution of the measurements by selecting an appropriate range from the previous list.

Note With the 7350 motion controller, changing the range of an ADC changes all other ADCs to the same range. Other NI motion controllers can set the range independent of other ADCs.

For example, if you are using a 7340 motion controller, the input signal ranges from -3 to +3 V, and you select the -5 to +5 V range, the 4,096 discrete values for the ADC are 2.44 mV apart instead of the 4.88 mV apart when using the -10 to +10 V range. If you are using a 7350 motion controller, the input signal ranges from -3 to +3 V, and you select the -5 to +5 V range, the 65,536 discrete values for the ADC are 152 μ V apart instead of the 305 μ V apart when using the -10 to +10 V range.

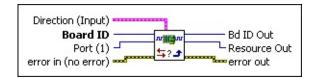
ADC ranges cannot be changed on any ADC while an axis that is using analog feedback is enabled. You must first disable the axis using analog feedback, then change the range.

For more information, refer to the **Read ADCs** and **Enable ADCs** VIs.

Set I/O Port Direction

Sets the direction of a general-purpose I/O port as input or output.

Device Compatibility



Details

Direction (Input) is the bitmap of directions for the bits in the I/O port:

- If **Direction Bit 0** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
- If **Direction Bit 1** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
- If **Direction Bit 2** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
- If **Direction Bit 3** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
 - If **Direction Bit 4** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
 - If **Direction Bit 5** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
 - If **Direction Bit 6** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.
 - If **Direction Bit 7** is TRUE (default) the bit is set to input. If it is FALSE the bit is set to output.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from

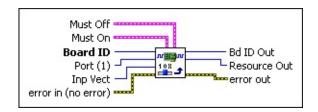
The Set I/O Port Direction VI configures the bits in a general-purpose I/O port as input or output. After setting the direction, use the Read I/O Port VI to read the port, the Set I/O Port MOMO VI to write to the port, and the Set I/O Port Polarity VI to set the polarity of each bit in the port to active high or active low.

Note The direction of bits in the RTSI software port (port 9) is controlled with the <u>Select Signal</u> VI.

Set I/O Port MOMO

Sets an I/O port value using the Must On/Must Off protocol.

Device Compatibility



Details Remarks

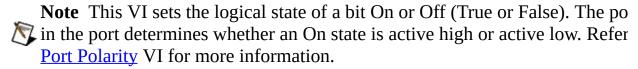
parameters:

Must Off is the bitmap of I/O port bits to force off.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Must On is the bitmap of I/O port bits to force on.
Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to <u>Input and Return Vectors</u> for more detailed information. Port (1) is the general-purpose I/O port (1–8) or RTSI software port (9) to
control.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following

status is TRUE if an error occurred before this VI was called, or

The Set I/O Port MOMO VI sets the logical state of bits in the general-purpose I.

Using the Must On/Must Off protocol allows you to set or reset individual bits w other output bits in the port. This gives you tri-state control over each bit: on, off **True** in a bit location of the **Must On** bitmap turns the bit on, while a **True** in the location of the **Must Off** bitmap turns the bit off. A **False** in either bitmap has not leaving both the **Must On** and **Must Off** bits at zero is effectively a hold and the is unchanged. If you set both the **Must On** and **Must Off** bits to **True**, it is interpretable to the protocol of the protocol o



Note When executing the Set I/O Port MOMO VI in a LabVIEW program, delay of up to 0.5 ms before the output lines are physically updated. The Lab flow continues, even though the output may not be updated yet.

Note If the motion controller is programmed to use the port for output when configured for input, the action is ignored.

The Set I/O Port MOMO VI allows individual control of general-purpose output requiring a shadow value or a read of the port to remember the state of other bits reset with the VI.

Example

Tip This section applies to all NI motion controllers.

In I/O port 2, to set bits 1 and 3 On, bits 0 and 5 Off, and to leave the other bits (annument of the set of t

Port = 2

Must On corresponds to the values shown in the following table.

VI

MOMO VI sets the logical state of bits in the general-purpose I/O port selected.

On/Must Off protocol allows you to set or reset individual bits without affecting in the port. This gives you tri-state control over each bit: on, off or unchanged. A tion of the **Must On** bitmap turns the bit on, while a **True** in the corresponding lust Off bitmap turns the bit off. A **False** in either bitmap has no effect, so **Must On** and **Must Off** bits at zero is effectively a hold and the state of the bit you set both the **Must On** and **Must Off** bits to **True**, it is interpreted as a Must the bit is turned on.

I sets the logical state of a bit On or Off (True or False). The polarity of the bits termines whether an On state is active high or active low. Refer to the <u>Set I/O</u> VI for more information.

executing the Set I/O Port MOMO VI in a LabVIEW program, there may be a o 0.5 ms before the output lines are physically updated. The LabVIEW program es, even though the output may not be updated yet.

notion controller is programmed to use the port for output when it has been or input, the action is ignored.

MOMO VI allows individual control of general-purpose output bits without w value or a read of the port to remember the state of other bits not being set or

ion applies to all NI motion controllers.

et bits 1 and 3 On, bits 0 and 5 Off, and to leave the other bits (2, 4, 6, and 7) his VI with the following parameters:

onds to the values shown in the following table.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- A line configured for input is not affected by this VI.
- You can always write to the RTSI software port, but the actual RTSI lines on the physical RTSI port are only affected if the RTSI line has been configured properly by using the <u>Select Signal</u> VI. By default, none of the RTSI lines are configured to output their corresponding bits in the RTSI software port; you must configure each RTSI line individually using the Select Signal VI, rather than the <u>Set I/O Port Direction</u> VI.
- The PCI-7390 has dedicated direction I/O ports. Refer to <u>73xx Controller</u> <u>General-Purpose I/O Port IDs</u> for the appropriate port number to use for **Port**.

For example, to set the state of output port 2, **Port** is 0x82 (130). To set the state of input port 3, **Port** is 0x03 (3).

NI SoftMotion Controller Considerations

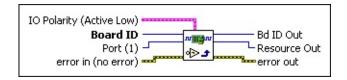
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- Refer to <u>General-Purpose I/O Ports</u> for information about how the NI SoftMotion Controller supports digital input and output lines for each axis.
- For **Inp Vect**, the NI SoftMotion Controller supports only the immediate vector (0XFF).
- The NI SoftMotion Controller has dedicated direction I/O ports. Refer to <u>NI SoftMotion Controller General-Purpose I/O Port IDs</u> for the

Set I/O Port Polarity

Sets the bit polarity in a general-purpose I/O port.

Device Compatibility



Details Remarks

IO Polarity is the bitmap of active polarities for the I/O port:

If **IO Polarity Bit 0** is TRUE (default) the bit is set to active low/active open. If it is FALSE the bit is set to active high/active closed.

If **IO Polarity Bit 1** is TRUE (default) the bit is set to active low/active open. If it is FALSE the bit is set to active high/active closed.

If **IO Polarity Bit 2** is TRUE (default) the bit is set to active low/active open. If it is FALSE the bit is set to active high/active closed.

If **IO Polarity Bit 3** is TRUE (default) the bit is set to active low/active open. If it is FALSE the bit is set to active high/active closed.

If **IO Polarity Bit 4** is TRUE (default) the bit is set to active low/active open. If it is FALSE the bit is set to active high/active closed.

If **IO Polarity Bit 5** is TRUE (default) the bit is set to active low/active open. If it is FALSE the bit is set to active high/active closed.

The Set I/O Port Polarity VI sets the polarity (active state) of the general-purpose I/O port on an individual bit basis.

When configured as active low, the input or output is active when there is a low signal on the pin. Conversely, active high means that the input or output is active when there is a high signal on the pin.

Configuring an active state of active open or active closed does not correspond to the level of the signal on the input or output pin. Instead, an active open state means that the input or output is active when current is not flowing through the optocoupled input. Conversely, an active closed state means that the input or output is active when current is flowing through the optocoupled input.

Typically, ports and their pins are configured for direction and polarity (active state) at initialization. After configuration, you can then read or write logical states (True or False, On or Off) to ports without worrying about the physical states of signals on the port pins.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The polarity also defines the latching behavior for the RTSI port. In order to detect short pulses on RTSI lines, the hardware latches active-going signals and holds that state until the port is read. For example, if you configure a bit for active low polarity, a transition from high to low is latched until read, even if the signal goes high again. If the signal starts low, it is also latched until read, even if the signal is high when you read the bit.
- The PCI-7390 has dedicated direction I/O ports. Refer to <u>73xx Controller</u> <u>General-Purpose I/O Port IDs</u> for the appropriate port number to use for **Port**.

For example, to configure the polarity of the output port on axis 2, **Port** is 0x82 (130). To configure the polarity of the input port on axis 3, **Port** is 0x03 (3).

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

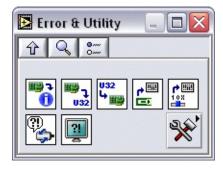
 The NI SoftMotion Controller has dedicated direction I/O ports. Refer to <u>NI SoftMotion Controller General-Purpose I/O Port IDs</u> for the appropriate port number to use for **Port**.

For example, to configure the polarity of the output port on axis 2, **Port** is

Error & Utility VIs

You can use the error handling VIs and utility VIs to get information about the motion controller. Refer to <u>Errors and Error Handling</u> for more information about errors.

Click the icons for VI descriptions.



Get Motion Board Information

Get u32

Set u32

Select Resource

Select MOMO

Motion Error Handler

Error Monitor

Subpalette

Advanced Error & Utility VIs

Error Monitor

Displays the errors returned by the NI-Motion software and controller.

Device Compatibility



terminal of the next VI.

Details

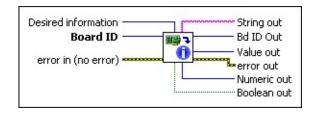
Board	ID is a unique number assigned by Measurement & Automation
Explo	rer (MAX) used to send and receive commands and data to or from
a spec	ific NI motion controller.
stop is	s used to control the monitoring process.
Bd ID	Out is provided for flow control. You can string together NI-
Motio	n VIs by wiring the Bd ID Out terminal of one VI to the Board ID

This VI can be made to run continuously in the background as a separate VI to monitor the motion controller for errors.

Get Motion Board Information

Gets information about the properties and features of the motion controller.

Device Compatibility



Details Remarks

Desired information represents the type of information you want to retrieve about the motion controller. The following table has a complete list of information values, as well as output parameters relevant for each type of information returned.

Desired Information	Relevant Output Terminal	Possible Output Values
Board Family (0)	Numeric out, Value out	NI-Motion Family (0)
Board Name (1)	String out	Name of controller
		PCI_7344 (28) PXI_7344 (27) PXI_7340 (62)

This VI returns selected information about NI motion controllers including controller type and family, bus type, number of axes, and so on.

NI-Motion also has four information types for retrieving the version numbers and release dates of the firmware segments loaded in the onboard Flash ROM. All firmware segments are field upgradable using the Update Firmware option in Measurement & Automation Explorer. Versions are returned in a version-build code format:

Version-build code = MMmmbbbb, where *MM* is the major version number, *mm* is the minor version number, and *bbbb* is the build number.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

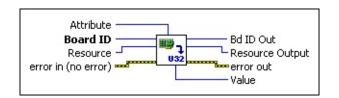
NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The NI SoftMotion Controller does not support the following enumerations for the **Desired Information** parameter:
 - Firmware Version
 - DSP Version
 - FPGA Version
 - FPGA2 Version

Get u32

Gets the general software settings.



Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Resource is the Axis, Vector Space, ADC, or Encoder you wired into the VI. Use Resource Output to pass the resource to another VI and/or to display information about the device.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out. The error in cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Attribute is the selector for the type of information you want to retrieve

The Get u32 VI returns selected data about software settings. This VI can retrieve values stored in integer format.

Attribute	Resource	Description	Possible Values
Breakpoint Window	Axis or encoder	After the axis encounters a breakpoint, the axis must move outside the breakpoint window before it can rearm for another breakpoint. This is used to prevent oscillation from sending multiple breakpoint outputs from the same breakpoint position. The default is 0.	0 to 255
Program Auto Start	N/A	Use this attribute to return the onboard program that is enabled for auto start.	_
Gearing Enable Status	Axis control or axis	Use this attribute to read the gearing enable status on an individual axis or on axes one through eight using the axis control resource. Reading the status on axes one through eight returns a bitmap where bit 0 is the status for axis 1 and so on. Reading the status on an individual axis returns a bitmap that is masked for that axis.	For each bit in the bitmap: 1 = gearing enabled 0 = gearing disabled

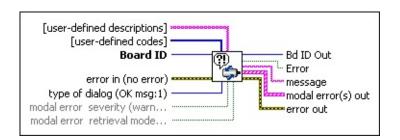
Using Unlisted Resources

Complete the following steps to use this VI on resources that are not in the drop-down list:

- 1. Open the VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the **Resource** terminal and select **Create»Constant**.
- 1 Click the dron-down arrow on the **Resource** constant and select **Other**

Motion Error Handler

Checks for modal errors on the controller.



Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

If the **modal error severity** input is TRUE (error), modal errors are treated as errors. The first modal error is then passed out through the **error out** terminal. If this input is FALSE (warning), modal errors are treated as warnings.

When using subVIs that incorporate the error in/error out (or error I/O) structure, place this handler where you want to inform the user of an error, typically at the end of the I/O data path, as the last action of the program. If the **error in** status is TRUE (a red X), the handler creates a message describing the error and its source.

If the Type of Dialog input is 1 (default), the message is displayed to the user, who can only acknowledge it. If this input is 2, the user can acknowledge the message or abort execution. If the input is 0, no message is displayed; use this value to process the error programmatically.

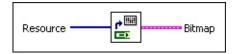
This VI always checks for modal errors on the controller. If the Modal Error Severity input is TRUE (error), modal errors are treated as errors. The first modal error is then passed out through the **error out** terminal. If this input is FALSE (warning), modal errors are treated as warnings.

If the Modal Error Retrieval Mode input is TRUE (single), only one modal error is read from the controller's error stack. If this input is FALSE (all), all of the modal errors on the controller are read. Modal errors that are read from the controller are placed into the Modal Error(s) Out output array.

Select Resource

This utility VI is used to combine resource IDs from an array into a single bitmap.

Device Compatibility



Details

Resource is the <u>Axis</u>, <u>Vector Space</u>, <u>ADC</u>, or <u>Encoder</u> you wired into the VI. Use **Resource Output** to pass the resource to another VI and/or to display information about the device.

Bitmap is the bitmap corresponding to the combination of the resources selected.

Select Resource VI is used to generate input bitmaps for VIs that operate on multiple resources.

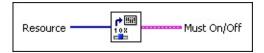
This VI returns a bitmap based on the resource IDs wired in. The order of the resources wired in does not matter. The true/false state of a bit in the bitmap is controlled by the resource ID and not by the wire position.

This VI is useful when used with **Start**, **Stop**, and **Blend** VIs.

Select MOMO

This utility VI is used to combine resource IDs into a single bitmap.

Device Compatibility



Details

- **Resource** is the <u>Axis</u>, <u>Vector Space</u>, <u>ADC</u>, or <u>Encoder</u> you wired into the VI. Use **Resource Output** to pass the resource to another VI and/or to display information about the device.
- **Must On/Off** The bitmap corresponding to the combination of the resources selected in Must On/Must Off format.

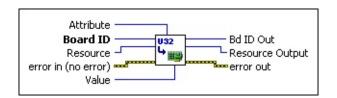
Select MOMO VI is used to generate input bitmaps for VIs that use the Must On /Must Off format protocol.

This VI returns a bitmap based on the resource IDs wired in. The order of the resources wired in does not matter. The true/false state of a bit in the bitmap is controlled by the resource ID and not by the wire position.

This VI is used to generate input bitmaps for VIs that use the MustOn/MustOff format protocol.

Set u32

Sets the general software settings.



Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Resource is the Axis, Vector Space, ADC, or Encoder you wired into the VI. Use Resource Output to pass the resource to another VI and/or to display information about the device.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out. The error in cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

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

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

Value is the selector for the type of data you want to set

status is FALSE, **code** is zero or a warning code.

The Set u32 VI sets selected software settings as specified by Attribute. This VI can set values stored in integer format.

The following table shows the attributes you can set, their descriptions, and possible values:

Attribute	Resource	Constant	Description	Values
Breakpoint Window	Axis or encoder	0x0200	After the axis encounters a breakpoint, the axis must move outside the breakpoint window before it can rearm for another breakpoint. This is used to prevent oscillation from sending multiple breakpoint outputs from the same breakpoint position. The default is 0.	0 to 255
			Note The breakpoint window must be smaller than the breakpoint modulus or period. A modal error is generated when the breakpoint window is larger than the loaded breakpoint modulus or period.	
Pull-in Window	Axis	0x0400	For closed-loop stepper axes, an area around the target position that determines whether or not to do any more pull-in moves. Once within the area, the motion controller stops doing pull-in tries. The default is 1.	0 to 32,767
Pull-in	Avis	0x0401	The maximum number of times the closed-loop stepper axis tries to reach the target position. It will try until it is within the range specified	0 to

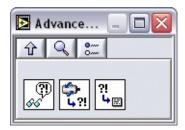
Using Unlisted Resources

Complete the following steps to use this VI on resources that are not in the drop-down list:

- 1. Open the VI in LabVIEW.
- 2. View the block diagram.
- 3. Right-click the **Resource** terminal and select **Create»Constant**.
- 4. Click the drop-down arrow on the **Resource** constant and select **Other**.
- 5. Type the appropriate resource number, in decimal, for the resource you want to configure. Refer to <u>Axes</u>, <u>Vector Spaces</u>, <u>ADC Channels</u>, and <u>Encoders</u> for resource numbers.
- 6. Right-click the **Attribute** terminal, select **Create»Constant**, and select the appropriate attribute from the drop-down list.

Advanced Error & Utility VIs

Click the icons for VI descriptions.



Read Error Message

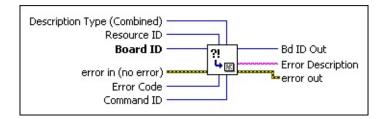
Get Last Error

Get Error Description

Get Error Description

Gets an error, command, and/or resource description string.

Device Compatibility



Details

Description Type (Combined) is the selector for the type of description string to return, as follows:

Value Description



- 0 Error Only
- 1 Command Name Only
- 2 Resource Name Only
- 3 Combined Description
- **Resource ID** is the resource ID of an axis, vector space, encoder, ADC, DAC, or other resource.
- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the

The Get Error Description VI returns a string description of an error. You can use this VI to generate a string for displaying a VI's name, a resource's name, an error code description, or a complete error description string in response to an error code returned from calling the <u>Read Error Message</u> VI.

Not all input parameters are required for each description type. The following parameters are required to return an accurate description string.

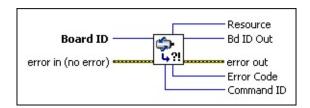
Description Type	Error Code	Command ID	Resource ID
Error only	required	not required	not required
VI name only	not required	required	not required
Resource name only	not required	required	required
Combined description	required	required	required

Because resource IDs are not unique (for example, axis 1 and program 1 both are resource 1), the Command ID is required to set the context and allow this VI to generate the proper resource name string.

Get Last Error

Gets detailed information about the last error generated by a high-level NI-Motion VI in the course of executing other NI-Motion VIs.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

source is a string that indicates the origin of the error, if any.

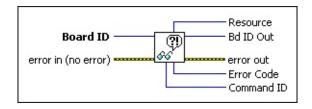
Typically, **source** is the name of the VI in which the error occurred.

If the <u>Initialize Controller</u> VI returns an error, you can call the Get Last Error VI to determine the specific NI-Motion VI and resource generating the error. You can then call the <u>Get Error Description</u> VI, which formats a string for you with the VI name, resource ID and error description.

Read Error Message

Reads the most recent modal error from the Error Message Stack.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

The Read Error Message VI retrieves the most recent modal error from the controller and returns the command ID and resource ID that cause the error, along with the error code.



Note See <u>Error Codes</u>, for a description of error codes and possible causes.

When a modal error occurs, the command ID, resource ID, and error code are automatically stored in a last-in first-out stack and the Error Message (Err Msg) bit in the Communication Status Register is set to indicate that one or more errors are present on the stack.

Modal errors are defined as errors that are not detected at the time of VI execution. For a complete description, refer to Errors and Error Handling. A common source of modal errors is improperly constructed VI calls stored in an onboard program. When the program is run, the errors generate modal error messages.

Because the error messages are stored in a last-in-first-out (LIFO) buffer, the most recent error is available immediately. You can read older errors with additional calls to this VI. When the stack is empty, the Error Message (Err Msg) bit in the Communication Status Register is reset. You can get a string description of the error by using the Get Error Description VI.

Normally, if the application program is functioning correctly, errors are not generated. Reading the error messages from the board is useful during debugging and for handling special conditions.

Example

An application program running on the host computer monitors the Communication Status Register to check for errors. If the Error Message bit is set, the program sends a Read Error Message VI to the controller and then reacts to the error information returned. Depending upon the type of error

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- The **Command ID** read in is always zero (0).
- The errors stack is always in FIFO mode.

Onboard Programming VIs

You can use Onboard Programming VIs to load, execute, and save onboard programs. NI-Motion offers a set of programming VIs and features that allow you to write and control autonomous programs that are completely independent from the host computer. You can execute up to 10 onboard programs simultaneously. The size and number of programs is completely flexible. It is ultimately limited by the 32 total memory objects in the Object Registry or by total available memory, whichever is reached first. Refer to Begin Program Storage for more information. Onboard programs run in a time-sliced manner. Refer to the Onboard Programs section of the *NI-Motion Help* for more information.

Onboard programs support basic math and data operation functions on general-purpose variables. Onboard programs also offer event-based VIs such as <u>Jump to Label on Condition</u> and <u>Wait on Condition</u>, which allow you to control program execution. Programs can even start and stop other programs.

Implementing part or all of the motion application as an onboard program or programs offloads the host computer from handling these real-time events. Onboard programs also can isolate the application from the host computer non-real-time operating system. Only bus power is required to correctly execute an onboard program after it is started.

Programs can be run from RAM or optionally saved to non-volatile Flash ROM. Saved programs are therefore available for execution at any future time, even after power cycles. Use the <u>Read Object Registry</u> VI to determine the size of an onboard program. Refer to <u>Buffered Operations VIs</u> for information about the total amount RAM and ROM on each controller.

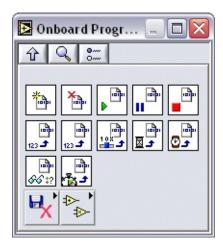
Onboard Programming VIs include <u>Object Management</u> and <u>Data Operations</u> VIs. You can use Onboard Programming VIs to begin and end program storage and to control program execution. You can use Object Management VIs to organize, annotate, and save program objects to ROM. Data Operations VIs include basic math VIs and data operations.

Several VIs are typically run from the host to control the onboard program

execution, such as the Run Program, Pause/Resume, and Stop Program VIs. When run as a host command, these VIs interrupt the onboard program and change the program status accordingly.

 $\hfill Note \ \hfill The Onboard Programming VIs are compatible with NI 7340/44 and NI 7350 controllers only.$

Click the icons for VI descriptions.



Begin Program Storage

End Program Storage

Run Program

Pause/Resume Program

Stop Program

Insert Label

Jump to Label on Condition

Set User Status MOMO

Wait on Condition

Load Program Delay

Read Program Status

Load Program Time Slice

Subpalettes

Object Management VIs

Data Operations VIs

Begin Program Storage

Begins a program storage session.

Subsequent calls are stored as part of the onboard program.

Board ID
Program (1)
Program (no error)

Bd ID Out
Resource Out

Device Compatibility

Details

parameters:

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Program (1) is the onboard program to read the status of.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already
occurred, this VI returns the value of error in in error out . The VI runs

normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

source is a string that indicates the origin of the error, if any.

The Begin Program Storage VI initiates program storage in RAM. After they begin, all subsequent VIs are stored in an object buffer and not executed until the program is run with the <u>Run Program</u> VI. This memory storage continues until you execute the <u>End Program Storage</u> VI. You can store only one program at a time.

The size and number of programs is completely flexible. It is ultimately limited by the 32 total memory objects in the Object Registry or by total available memory, whichever is reached first.

The 7350 and 7340 have the following onboard memory:

RAM ROM
7340 1 64 k sector 2 64 k sectors
7350 2 64 k sectors 4 64 k sectors

You can run programs from either RAM or ROM, but you cannot split programs between the two, and you cannot split programs between the two 64 KB ROM sectors. With an average command size of 10 bytes, a single program can be as large as 6,400 commands. As another example, the 7350 and 7340 controllers can execute 10 programs simultaneously, five from RAM and five from ROM, with each program up to 1,280 commands long.

Notes

• Attempting to store more than 32 programs generates an error. Similarly, an error is generated if you run out of memory during program storage. Both of these cases are extremely unlikely.

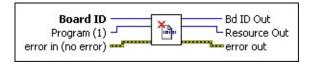


- Do not store onboard programs while other NI-Motion VIs are running. Doing so can result in incorrect program storage.
- To prevent unexpected interference, other applications running with different processes are not able to communicate with the controller during program storage.

End Program Storage

Ends a program storage session.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Program (1) is the onboard program to read the status of.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:
status is TRUE if an error occurred before this VI was called, or FALSE if not. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code.
code is a number identifying an error or warning. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message.
source is a string that indicates the origin of the error, if any.

Typically, **source** is the name of the VI in which the error occurred.

Motion VIs by wiring the **Bd ID Out** terminal of one VI to the **Board ID**

Bd ID Out is provided for flow control. You can string together NI-

The End Program Storage VI ends memory storage of the program. All subsequent VIs are executed normally. You can save a program to non-volatile memory (ROM) using the <u>Object Memory Management</u> VI.

This VI may take longer than 62 ms to process, so it is not guaranteed to be compatible with real-time execution.

Insert Label

Inserts a label in a program.

Board ID

error in (no error)

Board ID

error out

terminal of the next VI.

Device Compatibility

Details

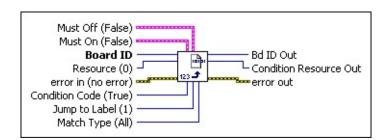
Label Number (1) is any arbitrary label number from 1 to 65,535.
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters: status is TRUE if an error occurred before this VI was called, or FALSE if not. If status is TRUE, code is a nonzero error code. If
status is FALSE, code is zero or a warning code.
code is a number identifying an error or warning. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message.
source is a string that indicates the origin of the error, if any. Typically, source is the name of the VI in which the error occurred.
Bd ID Out is provided for flow control. You can string together NI-

Motion VIs by wiring the **Bd ID Out** terminal of one VI to the **Board ID**

The Insert Program Label VI marks a location in the sequence of a program. The label number identifies this location and uses it in the <u>Jump to Label on Condition</u> VI. Label numbers are arbitrary and do not have to follow a numerical sequence.

Jump to Label on Condition

Inserts a conditional jump in a program.



Device Compatibility

Details

Must Off (False) is the bitmap of bits that must be False to satisfy the condition:

- **Must On 0**—If true, bit must be false. If false, do not care (default).
- **Must On 1**—If true, bit must be false. If false, do not care (default).
- **Must On 2**—If true, bit must be false. If false, do not care (default).
- **Must On 3**—If true, bit must be false. If false, do not care (default).
 - **Must On 4**—If true, bit must be false. If false, do not care (default).
 - **Must On 5**—If true, bit must be false. If false, do not care (default).
 - **Must On 6**—If true, bit must be false. If false, do not care (default).
 - **Must On 7**—If true, bit must be false. If false, do not care (default).
 - **Must On 8**—If true, bit must be false. If false, do not care (default).

Must On (False) is the bitmap of bits that must be True to satisfy the condition:

- **Must On 0**—If true, bit must be true. If false, do not care (default).
- **Must On 1**—If true, bit must be true. If false, do not care (default).
- **Must On 2**—If true, bit must be true. If false, do not care (default).
- With the 2—if true, bit must be true. If false, do not care (default).

The Jump to Label on Condition VI controls the flow of execution in a stored pro conditional jump to any label within the program. In addition to condition codes data operations VI, you can test virtually any instantaneous status of axes or reso jump or not.

There are two distinct groups of conditions. The first group, conditions 0 through recent logical, mathematical, or data transfer operations VI. Refer to the Data Op information about mathematical or data transfer operations. These condition code the latest logical, mathematical, or data transfer VI is less than zero, equal to zero greater than zero, not equal to zero, or greater than or equal to zero. For these con On, Must Off, and Match Type parameters are not required and their values are is



Note You can program unconditional jumps by setting the condition to True

The second group, conditions 7 and above, test a specific multi-axis, multi-encod general-purpose I/O status. Where applicable, you can select the appropriate resc parameter.

Program Complete is similar to the first condition group in that Must On, Must C parameters are not required and their values are ignored. You set resource equal t number to test. The balance of the conditions in this group test status bitmaps and described in the remainder of this section.

The Must On, Must Off, and Match Type parameters work together to define a bi that must be matched to satisfy the condition. The Match Type parameter allows match, where any matching bit is sufficient, and an AND match, where all status True/False bitmap defined by Must On and Must Off.

Using the Must On/Must Off protocol gives you tri-state control over each match care. A True in a bit location of the Must On bitmap sets the match bit to True, w corresponding location of the Must Off bitmap resets the match bit to False. A Fa affect, so leaving both the Must On and Must Off bits at False defines the bit as c the Must On and Must Off bits to True, it is interpreted as a Must On condition a

on VI controls the flow of execution in a stored program by defining a within the program. In addition to condition codes set as the result of a previous st virtually any instantaneous status of axes or resources to decide whether to

of conditions. The first group, conditions 0 through 5, test the result of the most or data transfer operations VI. Refer to the <u>Data Operations VIs</u> for more cal or data transfer operations. These condition codes test whether the result of al, or data transfer VI is less than zero, equal to zero, less than or equal to zero, zero, or greater than or equal to zero. For these conditions, the Resource, Must e parameters are not required and their values are ignored.

nconditional jumps by setting the condition to True (6).

7 and above, test a specific multi-axis, multi-encoder, program, motion I/O, or here applicable, you can select the appropriate resource with the resource

to the first condition group in that Must On, Must Off, and Match Type ad their values are ignored. You set resource equal to the appropriate program f the conditions in this group test status bitmaps and VIs similar to each other as this section.

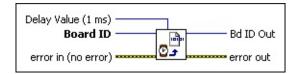
Match Type parameters work together to define a bitmap of True and False bits y the condition. The Match Type parameter allows you to select between an OR it is sufficient, and an AND match, where all status bits must match the Must On and Must Off.

protocol gives you tri-state control over each match bit: True, False, or Don't of the Must On bitmap sets the match bit to True, while a True in the Must Off bitmap resets the match bit to False. A False in either bitmap has no st On and Must Off bits at False defines the bit as do not care. If you set both s to True, it is interpreted as a Must On condition and the match bit is set to

Load Program Delay

Loads a delay into a program sequence.

Device Compatibility



Details

Delay Value (1 ms) is the delay in milliseconds. The range is from 1 to
2^{31} –1 ms.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

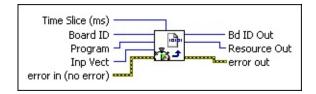
source is a string that indicates the origin of the error, if any.

The Load Program Delay VI suspends program execution for the number of milliseconds loaded. Program execution resumes after the delay. Delays can be as short as one or two milliseconds or as long as hundreds of hours.

Load Program Time Slice

Specifies the minimum time an onboard program has to run per watchdog period.

Device Compatibility



Details

Time Slice (ms) is the execution time for the onboard program per
watchdog period. Default is 2 ms.
Board ID is a unique number assigned by Measurement & Automa

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Program is the program number. Valid program numbers are 1 through 255 (0x01 through 0xFF).

Inp Vect indicates the source of the data for this VI. Available input vectors include immediate (0xFF), variable (0x01 through 0x78), or indirect variable (0x81 through 0xF8). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or

Load Program Time Slice specifies the minimum time an onboard program has to run per watchdog period.



Note This VI can only be run as a part of onboard program; the host cannot change the time slice of a program directly.

Some things to remember when changing the time slice of an onboard program:

- A total of 20 ms is allowed for all running onboard programs.
- Every onboard program loads with a default 2 ms time slice unless Load Program Time Slice is executed at the beginning of the onboard program. The default value of 2 ms is calculated based on maximum 10 onboard programs running simultaneously with equal time slice.
- You can assign different time slices for each stored onboard program as long as the total of time slice for the running onboard program does not exceed 20 ms at any given time.

For example, suppose you have the following onboard programs currently stored:

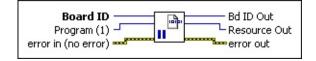
- 1. Onboard program 1, time slice = 10 ms
- 2. Onboard program 2, time slice = 10 ms
- 3. Onboard program 3, time slice = 5 ms
- 4. Onboard program 4, time slice = 4 ms
- 5. Onboard program 5, time slice = 1 ms

You could run programs 1 and 2 simultaneously without error, because the total time slices of the two running programs is 20 ms. If you then start program 3, an NIMC invalidTimeSlice modal error occurs because the new total time slice is 25 ms. The available time slice for an onboard program also depends on the order in which the programs are run, because a user-defined time slice takes effect only when the onboard program actually runs, and not when it is stored.

Pause/Resume Program

Pauses a running program or resumes execution of a paused program.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Program (1) is the onboard program to read the status of.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

The Pause/Resume Program VI suspends execution of a running program or resumes execution of a paused program.

A program can pause or resume execution of another program, and also can pause (but not resume) itself.

Note Pausing a program does not affect a move already started and in progress. It does not implement a Stop Motion VI.

Any run-time (modal) **error in** a program automatically pauses the program in addition to generating the error message. Refer to the <u>Read Error Message</u> VI and <u>Errors and Error Handling</u> for information about errors.

A program also can automatically pause if you execute a <u>Stop Motion</u> VI from the host computer on an axis or axes under control of the onboard program. In these cases, the program pauses when it attempts to execute a <u>Start Motion</u> or <u>Blend Motion</u> VI on the stopped axes. This automatic pause also applies when the stop is due to a limit, home, software limit, or following error condition.

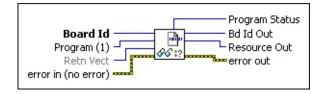
You can effectively single-step through an onboard program by having the program pause after every VI, and then resuming the program from the host computer.

System time does not pause when you pause a program. Pausing merely stops the next VI from executing. This means that a delay counter from Load Program Delay keeps counting down even while the onboard program is paused. For example, if you load both a delay of five seconds and then pause the program for 10 seconds, you have a total of only 10 seconds before execution of the onboard program resumes.

Read Program Status

Reads the status of an onboard program.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Program (1) is the onboard program to read the status of.

Retn Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), to a variable (0x01 through 0x78), to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

You can use this VI to determine the state of an onboard program. You can only read the state of other programs from an onboard program.

Run Program

Runs a previously stored program.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Program (1) is the onboard program to read the status of.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following
parameters:
 status is TRUE if an error occurred before this VI was called, or
code is a number identifying an error or warning. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message.
source is a string that indicates the origin of the error, if any. Typically, source is the name of the VI in which the error occurred.
Bd ID Out is provided for flow control. You can string together NI-Motion VIs by wiring the Bd ID Out terminal of one VI to the Board ID terminal of the next VI.

Resource Out is the Axis Vector Space ADC or Encoder you wired into

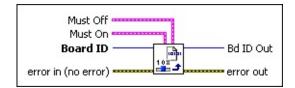
The Run Program VI initiates execution of the VIs stored in the selected program. You can run programs out of either RAM or ROM. You can run up to ten (10) programs simultaneously.

A program can run another program but you cannot have a program run itself. Attempting to store a recursive Run Program VI in a program generates an error and does not store the VI.

Set User Status MOMO

Controls the user status bits in the MCS register.

Device Compatibility



Details

Muct	Off ic	tho	bitmap	of	ucor	ctatue	hite	tο	force	\mathbf{F}_{2}	co.
wiust	OII IS	uie	Diunal) OI	user	Status	DILS	ω	Torce	rai	se:

- **Motion status register 13**—If true, user status bit forced false. If false, user status bit unchanged (default).
- **Motion status register 14**—If true, user status bit forced false. If false, user status bit unchanged (default).
 - **Motion status register 15**—If true, user status bit forced false. If false, user status bit unchanged (default).

Must On is the bitmap of user status bits to force True:

- **Motion status register 13**—If true, user status bit forced true. If false, user status bit unchanged (default).
- **Motion status register 14**—If true, user status bit forced true. If false, user status bit unchanged (default).
 - **Motion status register 15**—If true, user status bit forced true. If false, user status bit unchanged (default).
- **Board ID** is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
 - error in (no error) describes error conditions that occur before this VI

The Set User Status MOMO VI controls the upper three bits in the MCS register using the Must On/Must Off protocol. You can use this VI in programs to report special conditions back to the host computer by setting and resetting one or more of these bits. Refer to the Read Move Complete Status VI for more information about using the MCS register for high-speed polling.

Using the Must On/Must Off protocol allows you to set or reset individual user status bits without affecting the other user status bits. This gives you tri-state control over each bit: True, False, or Unchanged. A True in a bit location of the **Must On** bitmap sets the user status bit high, while a True in the corresponding location of the **Must Off** bitmap resets the user status bit low. A False in either bitmap has no affect, so leaving both the **Must On** and **Must Off** bits at zero is effectively a hold, and the state of the user status bit is unchanged. If you set both the **Must On** and **Must Off** bits to True, it is interpreted as a **Must On** condition and the user status bit is set high.

Example

After a conditional jump in a program, you want the program to flag the host with a success code. This can be accomplished by storing the Set User Status MOMO with Must On and Must Off as follows.

Must On 13 Must On 14 Must On 15

True False True

Must Off 13 Must Off 14 Must Off 15

False True False

This forces user status bits 13 and 15 True and user status bit 14 low. A subsequent poll of the MCS register returns **Move Complete Status** with the relevant bits set as follows.

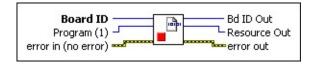
Status 13 Status 14 Status 15

True False True

Stop Program

Stops a running program.

Device Compatibility



terminal of the next VI.

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Program (1) is the onboard program to stop.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following
parameters:
status is TRUE if an error occurred before this VI was called, or FALSE if not. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code.
code is a number identifying an error or warning. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message.
source is a string that indicates the origin of the error, if any. Typically, source is the name of the VI in which the error occurred.
Bd ID Out is provided for flow control. You can string together NI-

Motion VIs by wiring the **Bd ID Out** terminal of one VI to the **Board ID**

Resource Out is the Axis Vector Space ADC or Encoder you wired into

The Stop Program VI terminates execution of a running program. You cannot resume a stopped program but you can re-run the program from the beginning.

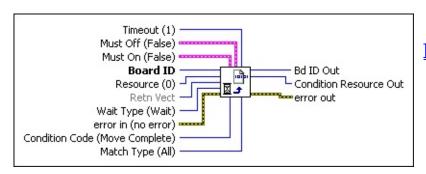
A program can stop another program but you cannot have a program stop itself. Attempting to store a recursive Stop Program VI in a program generates an error and does not store the VI.



Note Stopping a program does not affect a move already started and in progress. It does not implement a Stop Motion VI.

Wait on Condition

Inserts a conditional wait in a program.



Device Compatibility

Details

Timeout (1) is the wait timeout value in 100 millisecond increments. The range is 0 to 65,534. If you specify a timeout of 65,535, it is treated as an infinite timeout, and will wait until the condition is met. **Must Off** is the bitmap of bits that must be False to satisfy the condition: **Must Off 0**—If true, bit must be false. If false, do not care (default). **Must Off 1**—If true, bit must be false. If false, do not care (default). **Must Off 2**—If true, bit must be false. If false, do not care (default). **IIII** Must Off 3—If true, bit must be false. If false, do not care (default). -**IIII** Must Off 4—If true, bit must be false. If false, do not care (default). **Must Off** 5—If true, bit must be false. If false, do not care (default). **Must Off 6**—If true, bit must be false. If false, do not care (default). **Must Off** 7—If true, bit must be false. If false, do not care (default). **Must Off 8**—If true, bit must be false. If false, do not care (default). **Must On** is the bitmap of bits that must be True to satisfy the condition: **IIII Must On 0**—If true, bit must be true. If false, do not care (default). **IIII** Must On 1—If true, bit must be true. If false, do not care (default).

IIII Must On 2—If true, bit must be true. If false, do not care (default).

The Wait on Condition VI controls the flow of execution in a stored program. It suspends program execution and waits until the specified condition is met or the timeout expires. When the condition is met, program execution is resumed with the next VI after the Wait on Condition. Even though the program is suspended, the program status read using Read Program Status is still running.

If the timeout expires before the condition is met, an error is generated and the program goes into the paused state. Refer to the <u>Pause/Resume Program</u> VI for information about resuming a paused program.

If you set a timeout of zero, the condition must already be true or an error is generated.

You can wait on virtually any instantaneous status of axes, encoders, programs, motion I/O, or general-purpose I/O. Where applicable, you can select the specified resource with the resource parameter.

When waiting on a program with the Program Completed condition, **Must On**, **Must Off**, and **Match Type** parameters are not required and their values are ignored. You set resource equal to the program number to wait on. The balance of the conditions test status bitmaps and VI similar to each other as described in the remainder of this section.

The **Must On**, **Must Off**, and **Match Type** parameters work together to define a bitmap of True and False bits that must be matched to satisfy the condition. Depending on the type of event, the **Must On** and **Must Off** bitmaps are either one based or zero based. If the event is tied to an I/O port resource the bitmaps start at bit 0 to correspond with line 0 on an I/O port. If the event is tied to an axis-based resource, such as an encoder or an axis, the bitmaps start at bit 1 to correspond with axis 1. The **Match Type** parameter allows you to select between an OR match, where any matching bit is sufficient, and an AND match, where all status bits must match the True/False bitmap defined by **Must On** and **Must Off**.

Data Operations VIs

Data Operation VIs include the available math VIs on general-purpose variables. Variables can be loaded, added, multiplied, ANDed, and so on before being used as data in a motion control VI.

General-purpose variables are 32 bits long and can be used either signed (i32) or unsigned (u32). All Data Operation VIs operate on 32-bit values and return 32-bit values. You must be careful to avoid overflow and underflow conditions. For example, multiplying two 32-bit variables and returning the result to a 32-bit variable might overflow and wrap around.

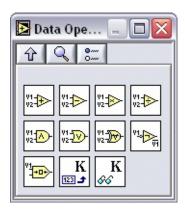
Smaller sized data is right aligned within a 32-bit variable. Bitwise logical VIs always assume this alignment and return similarly aligned results.

Many NI-Motion VIs can take input data from a general-purpose variable by pointing to the variable with the input vector parameter. Similarly, all read VIs can return data to a general-purpose variable by using the return vector parameter, refer to <u>Input and Return Vectors</u>.

All data operation VIs set condition codes (less than, equal to or greater than zero) depending on the result of the operation. Your program can test these conditions with the <u>Jump to Label on Condition</u> VI. Executing a data operations VI with a return vector of zero (0) tells the program to set the condition code and then throw the resulting data away. In this way, you can use all the data operations VIs as tests for conditional branching.

You can use indirect variables as variable inputs or return vectors with all data operation VIs.

Click the icons for VI descriptions.



Add Variables Exclusive OR Variables

Subtract Variables Invert Variable

Multiply Variables Logical Shift Variable

<u>Divide Variables</u> <u>Load Constant to Variable</u>

AND Variables Read Variable

OR Variables

Add Variables

Adds the values in the two variables and returns the result.

Var 2
Var 1

Board ID

Ret Vect
error in (no error)

V1

V2

V2

Provided in the serior out

Device Compatibility

Details

parameters:

Var 2 is the variable holding the second operand. Valid inputs are
variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Add Variables VI adds the values in the two variables and returns the result to the destination specified by the **Ret Vect**.

V1 + V2 = Result

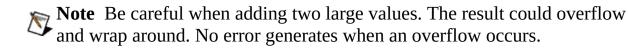
where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

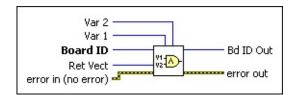
The result can be returned to a new variable or one of the two input variables, returned to the host computer, or thrown away. In all cases the condition codes are set according to the resulting value. This value is compared with 0, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero.



AND Variables

Performs a bitwise AND of the values in the two variables and returns the result.

Device Compatibility



parameters:

Var 2 is the variable holding the second operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
 Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
 Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
 Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.
 error in (no error) describes error conditions that occur before this VI

runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

status is TRUE if an error occurred before this VI was called, or

The AND Variables VI performs a bitwise logical AND of the values in the two variables and returns the result to the destination specified by the Ret Vect.

V1 AND V2 = Result

where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or one of the two input variables, returned to the host computer, or thrown away. The resulting value is compared to zero, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero. The EQUAL condition code is set True if the result equals zero (all bits low) and False if any bit is set. The GREATER THAN and LESS THAN codes are also set but can be confusing after logical bitwise operations.

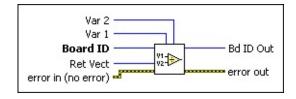
Example

If the values in Var 1 and Var 2 are 0x0000 1234 and 0x0000 EEEE, respectively, the result of the bitwise AND is 0x0000 0224 which is NOT EQUAL to zero.

Divide Variables

Divides the value in the first variable by the value in the second variable and returns the result.

Device Compatibility



Vectors for more detailed information.

Details

Var 2 is the variable holding the second operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host $(0xFF)$, return data to a variable $(0x01 \text{ through } 0x78)$, return data to an indirect variable $(0x81 \text{ through } 0xF8)$, and do not return data (0) . Refer to Input and Return

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Divide Variables VI divides the value in the first variable by the value in the second variable and returns the result to the destination specified by the Ret Vect.

V1/V2 = Result

where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or one of the two input variables, returned to the host computer or thrown away. In all cases the condition codes are set according to the resulting value. The resulting value is compared to zero, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero.

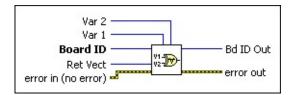
Note This VI does an integer divide and the remainder is lost.

Note If V2 = 0 the VI returns the invalid function data modal error (-70078).

Exclusive OR Variables

Performs a bitwise Exclusive OR (XOR) of the values in the two variables and returns the result.

Device Compatibility



Details

parameters:

Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Var 2 is the variable holding the second operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI

runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Exclusive OR Variables VI performs a bitwise logical XOR of the values in the two variables and returns the result to the destination specified by the Ret Vect.

V1 XOR V2 = Result

where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or one of the two input variables, returned to the host computer or thrown away. The resulting value is compared to zero, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero. The EQUAL condition code is set True if the result equals zero (all bits low) and False if any bit is set. The GREATER THAN and LESS THAN codes are also set but can be confusing after logical bitwise operations.

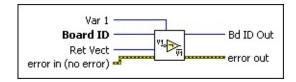
Example

If the values in Var 1 and Var 2 are 0x5A5A 1234 and 0xFFFF 4321, respectively, the result of the bitwise XOR is 0xA5A5 5115 which is NOT EQUAL to zero.

Invert Variable

Performs a bitwise inversion (NOT) on the value in a variable and returns the result.

Device Compatibility



Details

Var 1 is the location of the variable to invert. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
 Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE,

The Invert Variable VI performs a bitwise logical NOT on the value in the selected variable and returns the result to the destination specified by the Ret Vect.

 \sim (V1) = Result

where:

V1 is the value at **Var 1**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or to the input variable, returned to the host computer, or thrown away. In all cases the condition codes are set according to the resulting value. The resulting value is compared to zero, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero. The EQUAL condition code is set to True if the result equals zero (all bits low) and False if any bit is set. The GREATER THAN and LESS THAN codes are also set but can be confusing after logical bitwise operations.

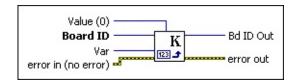
Example

If the value in Var 1 is 0x0000 5A5A, the result of the bitwise NOT is 0xFFFF A5A5. The EQUAL condition code is set to False.

Load Constant to Variable

Loads a constant value into a variable.

Device Compatibility



Details

™ Value ((0) is the	value to	load into	the variable.
-----------	-------------------	----------	-----------	---------------

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Var is the variable to load. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

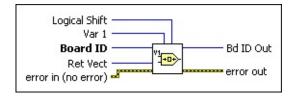
The Load Constant to Variable VI loads a constant value into the selected variable.

The condition codes are set according to the loaded value, GREATER THAN, LESS THAN, or EQUAL to zero.

Logical Shift Variable

Performs a logical shift on the value in a variable and returns the result.

Device Compatibility



Details

	Logical Shift is the number of bits to shift. A positive Logical Shift value
18	shifts Var 1 to the left and a negative value shifts Var 1 to the right. The shift range is -31 through +31 bits.
	Var 1 is the variable holding the value to shift. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
	Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.
	error in (no error) describes error conditions that occur before this VI

runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Logical Shift Variable VI performs a logical shift on the value in the selected variable and returns the result to the destination specified by the Ret Vect.

For positive Logical Shift values:

V1 « Logical Shift = Result

For negative Logical Shift values:

V1 » Logical Shift = Result

where:

V1 is the value at **Var 1**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or to the input variable, returned to the host computer or thrown away. In all cases the condition codes are set according to the resulting value, GREATER THAN, LESS THAN, or EQUAL to zero.

This VI actually performs an arithmetic rather than logical shift if the variable is a signed 32-bit value (i32). Negative values are sign-extended when shifted to the right. You can use this VI to perform division or scaling of signed or unsigned numbers. In this case the VI effectively performs the following:

 $V1 \times 2^{(Logical\ Shift)} = Result$

Example 1

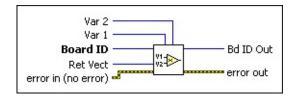
If the value in Var 1 is 0x0000 F002 and Logical Shift = -1, this VI returns 0x00007801.

Example 2

Multiply Variables

Multiplies the values in the two variables and returns the result.

Device Compatibility



Details

parameters:

Var 2 is the variable holding the second operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

status is TRUE if an error occurred before this VI was called, or

The Multiply Variables VI multiplies the values in the two variables and returns the result to the destination specified by the Ret Vect.

 $V1 \times V2 = Result$

where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or one of the two input variables, returned to the host computer or thrown away. In all cases the condition codes are set according to the resulting value. The resulting value is compared to zero, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero.

Note Be careful when multiplying two large values. The result can overflow and wrap around. An error is not generated when an overflow occurs.

OR Variables

Performs a bitwise OR of the values in the two variables and returns the result.

Var 2
Var 1

Board ID

Ret Vect
error in (no error)

Var 2

Var 1

Device Compatibility

Details

parameters:

Var 2 is the variable holding the second operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to <u>Input and Return Vectors</u> for more detailed information.
error in (no error) describes error conditions that occur before this VI

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The OR Variables VI performs a bitwise logical OR of the values in the two variables and returns the result to the destination specified by the Ret Vect.

V1 OR V2 = Result

where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

The result can be returned to a new variable or one of the two input variables, returned to the host computer or thrown away. The resulting value is compared to zero, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero. The EQUAL condition code is set True if the result equals zero (all bits low) and False if any bit is set. The GREATER THAN and LESS THAN codes are also set but can be confusing after logical bitwise operations.

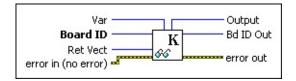
Example

If the values in Var 1 and Var 2 are 0x5A5A 1234 and 0x8282 0000, respectively, the result of the bitwise OR is 0xDADA 1234 which is NOT EQUAL to zero.

Read Variable

Reads the value of a variable and returns the result.

Device Compatibility



Details

Var is the variable to read. Valid inputs are variables ($0x01$ through $0x78$) and indirect variables ($0x81$ through $0xF8$).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI.

Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

The Read Variable VI reads the value of the selected variable and returns it to the destination specified by the Ret Vect.

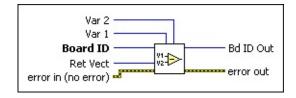
If the value of **Retn Vect** is set to Return to Host (0xFF), this VI also reads the value from the return data buffer and passes it back as output.

The condition codes are set according to the value read: GREATER THAN, LESS THAN or EQUAL to zero.

Subtract Variables

Subtracts the value of second variable from the value of the first variable and returns the result.

Device Compatibility



Details

parameters:

Var 2 is the variable holding the second operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Var 1 is the variable holding the first operand. Valid inputs are variables (0x01 through 0x78) and indirect variables (0x81 through 0xF8).
Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
Ret Vect indicates the destination for the return data generated by this VI. Available return vectors include return data to the host (0xFF), return data to a variable (0x01 through 0x78), return data to an indirect variable (0x81 through 0xF8), and do not return data (0). Refer to Input and Return Vectors for more detailed information.
error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following

The Subtract Variables VI subtracts the value of second variable from the value of the first variable and returns the result to the destination specified by the **Ret Vect**.

V1 - V2 = Result

where:

V1 is the value at **Var 1**.

V2 is the value at **Var 2**.

Result is stored in the location pointed to by **Ret Vect**.

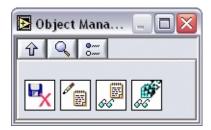
The result can be returned to a new variable or one of the two input variables, returned to the host computer or thrown away. In all cases the condition codes are set according to the resulting value. This value is compared with 0, and the appropriate condition code is set: GREATER THAN, LESS THAN, or EQUAL to zero.

This VI is often used to compare two values prior to executing a conditional jump with the <u>Jump to Label on Condition</u> VI. In this case, the result is typically thrown away by setting Ret Vect = 0.

Object Management VIs

You can use Object Management VIs to organize, annotate, and save program or buffer objects to flash ROM. These advanced VIs are primarily used for applications that require non-volatile program storage. You can run programs out of RAM and create and use buffers in RAM without using any of these VIs.

Click the icons for VI descriptions.



Object Memory Management

Load Memory Object Description

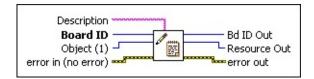
Read Memory Object Description

Read Object Registry

Load Memory Object Description

Loads an ASCII text description for a program or buffer object.

Device Compatibility



Details

abc	Description is an ASCII character array of up to 32 characters that describes the object.
	Board ID is a unique number assigned by Measurement & Automation
	Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
	Object (1) is a program or buffer stored in onboard RAM.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following
	parameters:

status is TRUE if an error occurred before this VI was called, or

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If

status is FALSE, **code** is zero or a warning code.

The Load Memory Object Description VI loads a text description for a program or buffer object. The ASCII text description is useful as a quick reminder of the contents or purpose of an object stored in memory.



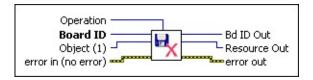
Note This VI must be executed while the object is still in RAM. After the object is saved to ROM, its description cannot be changed.

The description is limited to 32 characters; extra characters are ignored. You can retrieve the stored description with the Read Memory Object Description VI.

Object Memory Management

Saves, deletes, or frees programs or buffers in RAM and ROM.

Device Compatibility



Details

Operation is the operation to perform on the memory object, as follows:

Value Definition



- Save
- 1 Delete
- 2 Free

Board ID is a unique number assigned by Measurement & AutomationExplorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

Object (1) is a program or buffer stored in onboard RAM.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

The Object Memory Management VI is used to save to ROM, delete from ROM, or free from RAM, a program or buffer object. Objects saved to non-volatile Flash ROM are available for use at any future time, even after power cycles.

To save an object to ROM, call this VI with Operation = Save. The object is copied to ROM and exists in both RAM and ROM until the next power cycle, when the RAM image is erased.

To remove an object from ROM, call this VI with Operation = Delete. The object is deleted from both ROM and RAM (if it still exists in RAM).

After you have saved an object to ROM, you can free up its space in RAM by calling this VI with **Operation** = Free. This has no effect on the copy in ROM but deletes the image in RAM, making more memory available for storing additional programs or other objects.

Note You cannot save or delete an object while any program is running or any buffer is in use. Also, you cannot free a program or buffer while it is running. In addition, you cannot save or delete an object when any motor is moving. Attempting to execute this VI in these cases generates an error.

You can delete a buffer that is active, such as a high-speed capture buffer that is waiting for a trigger, but NI-Motion returns an error in this case.

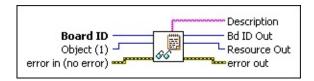
Tip Saving or deleting an object takes 2 to 4 seconds.

The 7340 controller has 64 KB of RAM plus 128 KB of ROM (divided into two 64 KB sectors) for program and buffer storage. You can run programs from either RAM or ROM, but you cannot split programs between the two, and you cannot split programs between the two 64 KB ROM sectors. With an average command size of 10 bytes, a single program can be as large as 6,400 commands. As another example, the 7340 controller can simultaneously execute 10 programs, five from RAM and five from ROM, with each program

Read Memory Object Description

Reads the ASCII text description for a program or buffer object.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation
Explorer (MAX) used to send and receive commands and data to or from
a specific NI motion controller.

Object (1) is a program or buffer stored in onboard RAM.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

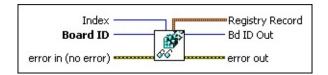
code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message.

The Read Memory Object Description VI returns the ASCII text description for a program or buffer object. The ASCII text description, previously loaded with the <u>Load Memory Object Description</u> VI, is useful as a quick reminder of the contents or purpose of an object stored in memory.

Read Object Registry

Reads a data record for a memory object from the Object Registry.

Device Compatibility



Details

Index is the registry record number. The range for index is 0 to 31.
 Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.
 error in (no error) describes error conditions that occur before this VI

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

The Read Object Registry VI returns a registry record for an object from the Object Registry. The Object Registry contains information about all objects stored in memory. You can store up to 32 objects in RAM and/or ROM. Each time an object is stored, a new record is created to keep track of it.

Note If an object exists both in RAM and ROM, this VI reports the object as existing in RAM. To see the object in ROM, you must first free the object from RAM.

On 7330 and 7340 motion controllers, objects are in RAM if **PStart** \bigcirc 0x100000, and in ROM if **PStart** < 0x100000. On the 7350 controller, objects are in RAM if **PStart** \bigcirc 0x200000, and in ROM if **PStart** < 0x200000.

Registry records are referenced by index and each call to this VI returns information about the referenced object. The index is not the same as the object number. You can use up to 255 unique object numbers (0x01 through 0xFF) but only 32 objects can be stored in memory at one time.

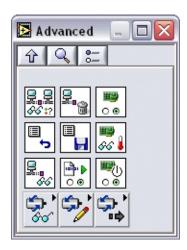
Advanced VIs

You can use the advanced VIs to control the communications between the host computer and NI motion controller. These VIs allow you to check the status of communications, clear the RDB, and manage the low-level communications to the controller. You typically do not have to use any of these VIs because the default configuration is correct for almost all applications. These VIs are available to handle special applications.

The advanced VIs include specialized <u>Read</u> and <u>Write</u> VIs for quickly viewing axis or coordinate data or status information, configuring I/O lines, and so on.

The advanced VIs also include advanced <u>methods</u> and two utility VIs that are regularly used but are different from the rest of the NI-Motion API in that they are not typically included in application code: <u>Clear Power Up Status</u> and <u>Save Default Parameters</u>.

Click the icons for VI descriptions.



Read Communication Status
Flush Return Data Buffer
Clear Power Up Status
Reset Default Parameters
Save Default Parameters

Read Board Temperature
Read Return Data Buffer
Enable Auto Start
Enable Shutdown

Subpalettes

Advanced Read VIs

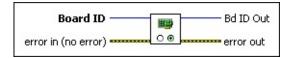
Advanced Write VIs

Advanced Methods

Clear Power Up Status

Clears the Power-Up status bit and boots up the controller, making it ready to accept commands.

Device Compatibility



Details Remarks

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Whenever the motion controller is reset by a power cycle, watchdog timeout, or other means, the controller is suspended in a power-up state and a Power-Up status bit in the Communications Status Register (CSR) is set. The Clear Power Up Status VI is used to clear this bit and ready the controller for motion control communications.

You cannot execute most of the other motion control VIs until the Power-Up status bit is cleared by this VI. This lockout ensures that you are aware of the occurrence of an unexpected reset, as in the case of a watchdog timeout.

You can include this VI one time at the beginning of an initialization routine, but to avoid the possibility of restarting an application unexpectedly after a power cycle or watchdog timeout, you must not include it in other routines. The <u>Initialize Controller</u> VI automatically calls this VI.

Note If you are running a NI-Motion startup application on ETS (PharLap) systems, the first VI in your application must be either the Clear Power Up Status or Initialize Controller VI. Calling any other VI first may result in error –70012 (NIMC_badBoardIDError) because the driver may not have loaded before a NI-Motion VI is called.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following includes considerations you must make when you are using this VI with a 73xx motion controller:

When the motion controller is in the Power-Up state, the MCS register contains a power-up code that describes why the controller is in the Power-Up state. To access this code, execute the Read Move Complete Status VI. The following table describes the power-up codes.

Code	Reset Type	Cause
0x80	Bus reset	Normal PC power cycle
0x40	Power-Up reset	Normal PC power cycle
0x20	Watchdog timeout	Fatal internal error
0x08	Shutdown	Shutdown (E-Stop) input active; refer to the <u>Enable Shutdown</u> VI
0x02	Software reset	Firmware download

NI SoftMotion Controller Considerations

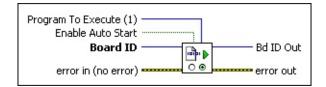
The following includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

The power-up codes do not apply to the NI SoftMotion Controller.

Enable Auto Start

Allows you to automatically run a program when the controller powers up.

Device Compatibility



Details

Program To Execute (1) is the onboard program the controller executes if
the auto start feature is enabled. This must be a valid program number (1-
255), that is stored to FLASH using the Object Memory Management VI.

Enable Auto Start enables or disables the auto start feature. Set this to Enable (true) to enable auto start and Disable (false) to disable auto start.

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If status is TRUE,

The Enable Auto Start VI configures the controller to automatically start an onboard program on power up. After auto start is enabled, the controller automatically executes the onboard program specified when the controller is powered on. The onboard program to execute must be saved to FLASH using the Object Memory Management VI before the controller is powered down. If the controller does not find a valid program that it can load, NIMC_autoStartFailedError is generated. If the onboard program is removed from FLASH memory, the auto start functionality is disabled.

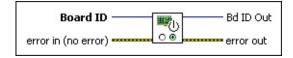
Note This VI writes to onboard FLASH memory and hence it is not safe to execute when motors are in motion. Doing so generates a NIMC_wrongModeError.

This VI may take longer than 62 ms to process, so it is not guaranteed to be compatible with real-time execution.

Enable Shutdown

Enables the shutdown (E-Stop) functionality of the controller.

Device Compatibility



<u>Details</u> <u>Remarks</u>

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Bd ID Out is provided for flow control. You can string together NI-Motion VIs by wiring the **Bd ID Out** terminal of one VI to the **Board ID**

The Enable Shutdown VI enables the controller to react to the shutdown (E-Stop) input. When the shutdown (E-Stop) input transitions from low to high, the controller goes into a shutdown state. The following actions take place in the shutdown state:

- All the axes are killed. On servo axes, the control loop is disabled and the
 output DACs are zeroed, allowing frictional forces alone to stop the
 motion. On stepper axes, the stepper pulse generation is stopped. On both
 axis types, there is no trajectory profile. If enabled, the inhibit output is
 activated to inhibit (disable) the servo amplifier or stepper driver. You can
 enable the inhibit outputs and set their polarity as active high or active low
 with the Configure Inhibit Outputs VI.
- All the axes, encoders, and ADCs are disabled.
- All the digital I/O is re-initialized to defaults. If the user has saved defaults using the Save Default Parameters VI, the digital I/O is re-initialized to the user defaults. Otherwise, the digital I/O is re-initialized to the factory defaults.
- All currently executing onboard programs are stopped.
- The controller does not accept any commands, except for the following ones:
 - Get Motion Board Information
 - Read Error Message
 - Enable Auto Start

The shutdown functionality is disabled by default on power up. This functionality has to be enabled every time the controller is powered up. You must enable this feature only after the shutdown (E-Stop) circuit is properly configured and connected to the controller. After shutdown (E-Stop) is enabled, it can be disabled only by resetting or power cycling the controller.

Note After the controller has shut down, you have to reset it or cycle its power before it can be used again.

Note The active state of the shutdown (E-Stop) line is low. To prevent the

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI SoftMotion Controller Considerations

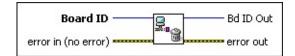
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- All read VIs are available when the NI SoftMotion Controller is in a shutdown state.
- All write VIs, including <u>Start Motion</u> and <u>Stop Motion</u>, are *unavailable* when the NI SoftMotion Controller is in a shutdown state.

Flush Return Data Buffer

Clears the Return Data Buffer (RDB) by deleting all of the buffered data.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Bd ID Out is provided for flow control. You can string together NI-

The Flush Return Data Buffer VI clears the Return Data Buffer by repeatedly reading the RDB until the buffer is empty. All return data packets in the RDB are deleted and this VI returns nothing.

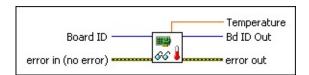
You typically use the Flush Return Data Buffer VI after an error condition when the data in the Return Data Buffer is no longer valid or relevant. This VI is also useful for flushing the RDB after a programming error has caused the buffer to become skewed. Buffer skew is when the data returned by a read VI using the default input vector of 0xFF does not return the expected data but rather returns data requested by a previous VI.

Refer to the motion controller documentation for more information about low-level communications protocols and return data packets.

Read Board Temperature

Reads the temperature from the motion controller.

Device Compatibility



Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Temperature is the temperature read from the temperature sensor on the motion controller in °C.

Bd ID Out is provided for flow control. You can string together NI-Motion VIs by wiring the **Board ID Out** terminal of one VI to the **Board ID** terminal of the next VI.

error out contains error information. If **error in** indicates an error, **error out** contains the same error information. Otherwise, it describes the error status that this VI produces

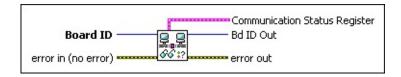
The Read Board Temperature VI reads the temperature in °C from the temperature sensor on the motion controller. You can use the current temperature sensor reading to determine whether the device requires recalibration.

Note Only 7350 controllers support the Read Board Temperature VI.

Read Communication Status

Reads the communication status register (CSR) which represents the state of the controller.

Device Compatibility



<u>Details</u> <u>Remarks</u>

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Communications Status Degister is the hitman of communication status

On power up, the controller is in a power-up reset state. Refer to <u>Clear Power</u> Up Status for information about clearing the reset condition.



Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- This VI performs a direct read of the Communication Status Register (CSR) on the 73xx motion controllers. The CSR is a hardware register containing communication handshaking and error status bits. The NI-Motion software polls this register continuously when sending and receiving packets for handshaking and error checking purposes. Refer to your motion controller documentation for more information about low-level communication protocols and return data packets.
- You also can call this VI at any time to check the communication and error status. Because the CSR is always up-to-date and directly accessible over the computer bus, executing this VI does not affect the operation of the NI motion controller.

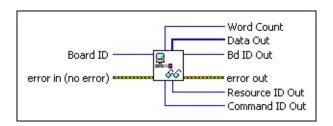
NI SoftMotion Controller Considerations

The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

- If you are using the NI SoftMotion Controller, this VI reads the NI SoftMotion Controller engine service status.
 - Ready to Receive indicates if the NI SoftMotion Controller is running.
- The NI SoftMotion Controller does not support the following CSR bits:
 - Command In Process
 - Packet Error
 - Hardware Fail

Read Return Data Buffer

Reads the Return Data Buffer.



Device Compatibility

Details

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Word Count is the number of 16-bit words in the command nacket

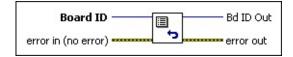
You can use the Read Return Data Buffer VI to read the RDB.

Refer to the motion controller documentation for more information about low-level communications protocols and return data packets.

Reset Default Parameters

Resets the power-up defaults to the factory-default settings.

Device Compatibility



<u>Details</u> <u>Remarks</u>

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Bd ID Out is provided for flow control. You can string together NI-Motion VIs by wiring the **Bd ID Out** terminal of one VI to the **Board ID**

The Reset Default Parameters VI resets the power-up defaults to the factorydefault settings for all important configuration, initialization, and trajectory parameters for use after subsequent power-up resets. When you execute this VI, all parameters are saved to nonvolatile flash memory and become the power-up defaults.

Note The effect of this VI is not realized until the next time the controller is powered up from a power-down state.



You only need to use this VI if you have previously modified the power-up defaults using the Save Default Parameters VI and want to revert back to the factory defaults.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

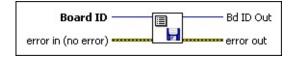
The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

• This VI may take longer than 62 ms to process on 73xx controllers. This VI is not guaranteed to execute in real time as it performs memory allocation.

Save Default Parameters

Saves the current operating parameters as defaults.

Device Compatibility



<u>Details</u> <u>Remarks</u>

Board ID is a unique number assigned by Measurement & Automation Explorer (MAX) used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

Bd ID Out is provided for flow control. You can string together NI-Motion VIs by wiring the **Bd ID Out** terminal of one VI to the **Board ID**

The Save Default Parameters VI saves all important configuration, initialization, and trajectory parameters for use after subsequent power-up resets. When you execute this VI, all parameters are saved to nonvolatile flash memory and become the power-up defaults.

If necessary, you can reinstate the factory-default parameters as the power-up defaults with the Reset Default Parameters VI.

This VI does not perform a complete state save. For proper and safe operation after power-up, certain parameters are always reset to their factory defaults to bring the controller back to a known safe state. Parameters not stored are left out by design and are typically reset to zero at power-up.

Note If you want to remember a parameter that is not included in this list, you can copy that parameter to a general-purpose variable and it is saved with this VI. You can then reset the parameter to the saved value with a program designed for this purpose.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- This VI may take longer than 62 ms to process on 73xx controllers. This VI is not guaranteed to execute in real time as it performs memory allocation.
- When the controller is powered up, the defaults are automatically applied.
 There is some time, however, between the controller powering up and the application of defaults.

The defaults on the step, direction, breakpoint, and analog output signals do not take effect until the power up reset state is cleared. Use the <u>Clear Power Up Status</u> VI to clear the power up reset state.

NI SoftMotion Controller Considerations

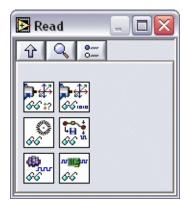
The following list includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

• User defaults are not applied until the power up reset state is cleared. Use the <u>Clear Power Up Status</u> VI to clear the power up reset state. Refer to the documentation for your drive for the initial power on state of the drive.

Advanced Read VIs

You can use the advanced read functions to view axis or coordinate execution data and status information, I/O status information, encoder execution data, and so on.

Click the icons for VI descriptions.



Read Execution Status

Read Execution Data

Read Encoder Data

Read Capture Compare Data

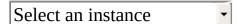
Read Motion I/O Data

Read Digital I/O Data

Read Execution Data

Reads position, velocity, and following error information on an axis or coordinate. You can use this polymorphic VI to read execution data of a single axis attribute, all execution data information from an axis, position information for all axes in a coordinate, or velocity or following error information for a coordinate.

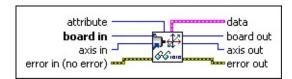
Use the pull-down menu to select an instance of this VI.



Details

Device Compatibility

Read Axis Data



attribute is the attribute you want to read. The following are valid attributes:

- position—Returns the position to the double element of the data cluster.
- **velocity**—Returns the velocity to the **double** element of the **data**
- **following error**—Returns the following error to the **double** element of the **data** cluster.

board in is a unique number assigned by Measurement & Automation
Explorer used to send and receive commands and data to or from a
specific NI motion controller.

axis in is the axis to read with this VI.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out. The error in cluster contains the following parameters:

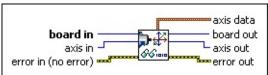
status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

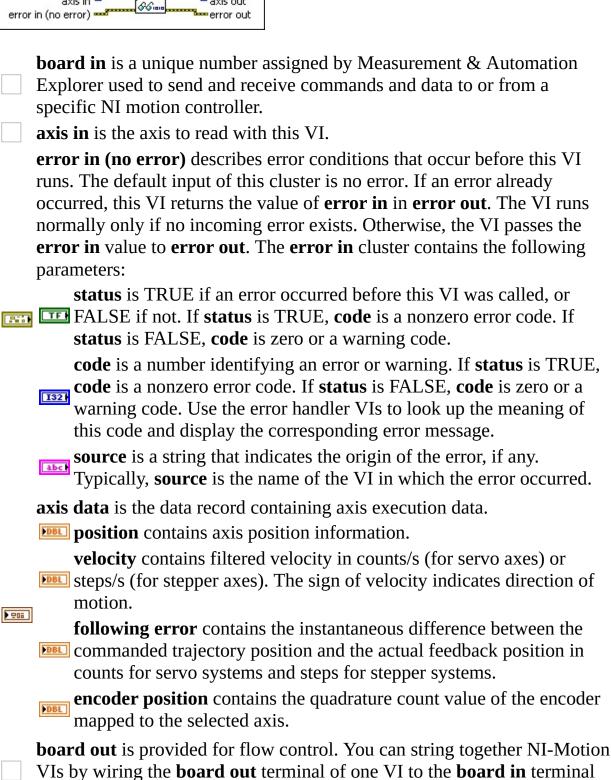
code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message

this code and display the corresponding error message.					
	source is a string that indicates the origin of the error, if any. Typically, source is the name of the VI in which the error occurred				
	data is a cluster of variables that contains the values for the selected attribute . Read the correct element of the cluster for the attribute. The values of the other elements in the cluster are undefined.				
	integer is 32-bit integer data.				
	double is 64-bit floating point data.				

Read All Axis Data

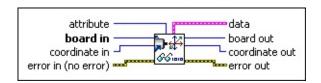
of the next VI.





axis out is provided for flow control. You can string together NI-Motion

Read Coordinate Data



attribute is the attribute you want to read. The following are valid attributes:

- **velocity**—Returns the velocity to the **double** element of the **data** cluster.
- **following error**—Returns the following error to the **double** element of the **data** cluster.

board in is a unique number assigned by Measurement & Automation
Explorer used to send and receive commands and data to or from a
specific NI motion controller.
coordinate in is the coordinate to read with this VI.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already
occurred, this VI returns the value of error in in error out . The VI runs
normally only if no incoming error exists. Otherwise, the VI passes the
error in value to error out . The error in cluster contains the following
parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

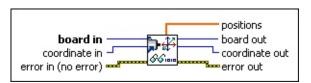
source is a string that indicates the origin of the error, if any. Typically, **source** is the name of the VI in which the error occurred.

data is a cluster of variables that contains the values for the selected **attribute**. Read the correct element of the cluster for the attribute. The values of the other elements in the cluster are undefined.

board out is provided for flow control. You can string together NI-Motion

V	alues of the other elements in the clus
	integer is 32-bit integer data.
	double is 64-bit floating point data
	Boolean is Boolean data.

Read Coordinate Position



zero or a warning code.

	board in is a unique number assigned by Measurement & Automation
	Explorer used to send and receive commands and data to or from a
	specific NI motion controller.
	coordinate in is the coordinate to read with this VI.
	error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of error in in error out . The VI runs normally only if no incoming error exists. Otherwise, the VI passes the error in value to error out . The error in cluster contains the following parameters:
P.	status is TRUE if an error occurred before this VI was called, or FALSE if not. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code.
	code is a number identifying an error or warning. If status is TRUE, code is a nonzero error code. If status is FALSE, code is zero or a warning code. Use the error handler VIs to look up the meaning of this code and display the corresponding error message. source is a string that indicates the origin of the error, if any. Typically, source is the name of the VI in which the error occurred.
[DBL	positions is the array containing the coordinate position information.
	board out is provided for flow control. You can string together NI-Motion VIs by wiring the board out terminal of one VI to the board in terminal of the next VI.
	coordinate out is provided for flow control. You can string together NI-Motion VIs by wiring the coordinate out terminal of one VI to the coordinate in terminal of the next VI.
	error out contains error information. If error in indicates an error, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
	status is TRUE if an error occurred, or FALSE if not. If status is TRUE, code is a nonzero error code. If status is FALSE, code is

code is a number identifying an error or warning. If **status** is TRUE,

Caution If this VI uses reflective memory, it should *not* be used for event checking. In this case, only use the information returned by this VI for display purposes. Refer to Reflective Memory VIs for more information.



Note Because of the way the controller updates information, the values returned are not guaranteed to all be sampled in the same time slice.

Axis Data Instances

The single axis versions of this VI return execution data on the selected axis.

For open-loop stepper axes, the VI returns position information in number of steps generated. For closed-loop stepper axes, it converts the primary feedback position from counts to steps and then returns the value in steps. Closed-loop stepper axes require you to correctly load values of steps per revolution and counts per revolution to function correctly.

Note For closed-loop axes, this VI always returns the position of the primary feedback resource. Refer to the Read Coordinate Position instance of this VI for an excession of the of this VI for an easy way to read the positions of all axes in a coordinate in one call.

For axis velocity, this VI returns filtered velocity in counts/s (for servo axes) or steps/s (for stepper axes). The sign of velocity indicates direction of motion.

For axis following error, this VI returns the instantaneous difference between the commanded trajectory position and the actual feedback position in counts for servo systems and steps for stepper systems.

Encoder position data is in quadrature count value of the encoder mapped to the selected axis.

During axis setup, you can operate the closed-loop stepper axis in open-loop mode and use this VI to directly measure the counts per revolution and steps per revolution for the axis. These values must be loaded before for subsequent closed-loop operation. Refer to the <u>Load Counts/Steps per Revolution</u> VI for more information.

You also can use **encoder position** to return a finer reading of position in cases where the encoder resolution greatly exceeds the step resolution of the closedloop stepper axis.

Read Execution Status

You can use this polymorphic VI to read the execution status of a selected axis execution status attribute, all execution status attributes on an axis, or a selected coordinate execution status attribute.

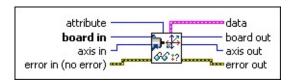
Use the pull-down menu to select an instance of this VI.

Select an instance

Details

Device Compatibility

Read Axis Status



attribute is the attribute you want to read. Refer to the <u>Details</u> section for more information. The following are valid attributes:

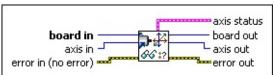
- **axis active**—Returns the status to the **Boolean** element of the **data** cluster.
- **move complete**—Returns the status to the **Boolean** element of the **data** cluster.
- profile complete—Returns the status to the Boolean element of the data cluster.
- **blend complete**—Returns the status to the **Boolean** element of the **data** cluster.
- **following error exceeded**—Returns the status to the **Boolean** element of the **data** cluster.
- **velocity threshold exceeded**—Returns the status to the **Boolean** element of the **data** cluster.
- **axis moving**—Returns the status to the **Boolean** element of the **data** cluster.
- direction forward—Returns the status to the Boolean element of the data cluster.

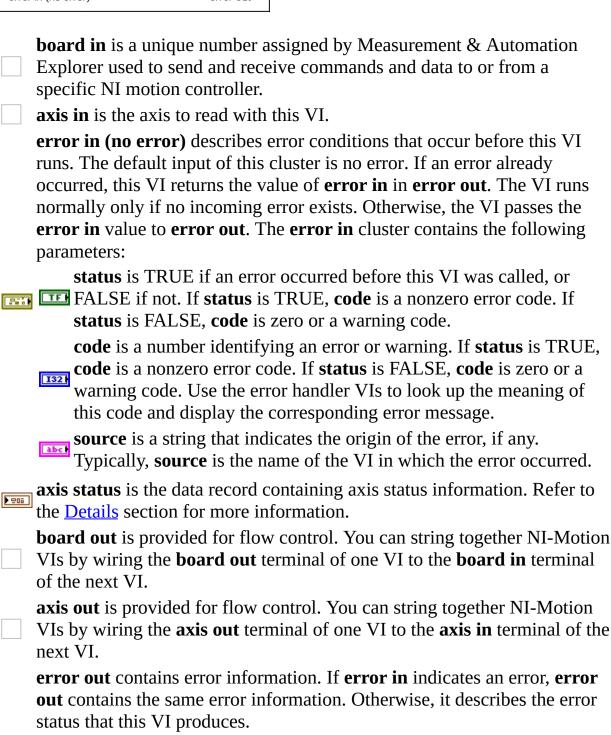
board in is a unique number assigned by Measurement & Automation
Explorer used to send and receive commands and data to or from a
specific NI motion controller.
axis in is the axis to read with this VI.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already
occurred, this VI returns the value of error in in error out. The VI runs
normally only if no incoming error exists. Otherwise, the VI passes the
error in value to error out . The error in cluster contains the following
narameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

code is a number identifying an error or warning. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a

Read All Axis Status





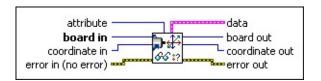
code is a number identifying an error or warning. If status is TDHE

status is TRUE if an error occurred, or FALSE if not. If **status** is

TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is

zero or a warning code.

Read Coordinate Status



attribute is the attribute you want to read. Refer to the <u>Details</u> section for more information. The following are valid attributes:

- move complete—Returns the status to the Boolean element of the data cluster.
- **profile complete**—Returns the status to the **Boolean** element of the **data** cluster.
- **blend complete**—Returns the status to the **Boolean** element of the **data** cluster.
- **following error exceeded**—Returns the status to the **Boolean** element of the **data** cluster.

board in is a unique number assigned by Measurement & Automation
Explorer used to send and receive commands and data to or from a
specific NI motion controller.

coordinate in is the coordinate to read with this VI.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

data is a cluster of variables that contains the values for the selected **attribute**. Read the correct element of the cluster for the attribute. The values of the other elements in the cluster are undefined.

Use this VI to read the execution status of an axis or coordinate.

Caution If this VI uses reflective memory, it should *not* be used for event checking. In this case, only use the information returned by this VI for display purposes. Refer to Reflective Memory VIs for more information.



Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

The VI will return **True** or **False** based on the following criteria:

Element Execution Status

axis **True** = motor is on **False** = motor is off active

move **True** = move complete

complete False = move not complete

profile **True** = profile is complete

complete False = profile generation in process

blend **True** = blend complete **complete False** = blend pending

> **True** = axis or coordinate axis exceeded the programmed following error limit

False = axis or coordinate axis following error is below the programmed following error limit

following error exceeded

Note If you are running a contoured move or slave axis move with a stepper axis, it is possible that the contour velocity or the slave axis geared velocity exceeds the maximum step rate of the Controller. In this case, the controller kills the axis and sets the following error status to true. Refer to the *Specifications* section of your controller user manual for the maximum step rate of your controller.

velocity threshold exceeded

True = axis velocity is above the programmed velocity threshold **False** = axis velocity is below the programmed velocity threshold

axis **True** = axis is moving moving **False** = axis is stopped

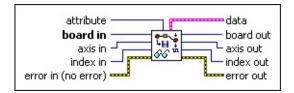
direction True = axis moving forward **forward False** = axis moving in reverse

index

Read Capture Compare Data

Reads position compare or position capture data from the selected axis.

Device Compatibility



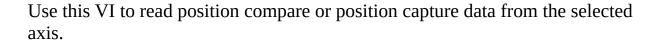
Details

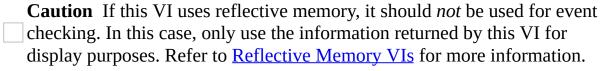
attribute is the attribute you want to read. The following are valid attributes:

- **captured position**—Returns the position to the **double** element of the **data** cluster.
- **position capture occurred**—Returns the status to the **Boolean** element of the **data** cluster. True (On) indicates that the position capture occurred and False (Off) indicates that the position capture has not occurred.
- **position compare occurred**—Returns the status to the **Boolean** element of the **data** cluster. True (On) indicates that the position compare occurred and False (Off) indicates that the position compare has not occurred.

board in is a unique number assigned by Measurement & Automation
Explorer used to send and receive commands and data to or from a
specific NI motion controller.
axis in is the axis to read with this VI.
index in is the encoder index value. Valid value is 1 for the primary
encoder.
error in (no error) describes error conditions that occur before this VI
runs. The default input of this cluster is no error. If an error already

occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** cluster contains the following



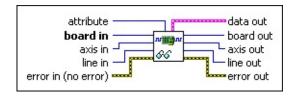


Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Read Digital I/O Data

Reads the attribute on a single digital I/O line.

Device Compatibility



Details Remarks

attribute is the attribute you want to read. The following are valid attributes:

- **output active state**—Returns the active state of the specified output line to the **Boolean** element of the **data** cluster. TRUE indicates active low/active open. FALSE indicates active high/active closed.
- **input active state**—Returns the active state of the specified input line to the **Boolean** element of the **data** cluster. TRUE indicates active low/active open. FALSE indicates active high/active closed.
- **output active**—Returns the current output state of the specified line to the **Boolean** element of the **data** cluster. TRUE indicates the output line is active. FALSE indicates the output line is inactive.
 - **Note** Reading this attribute on a line configured for input returns error -70102 (NIMC_wrongIODirectionError).
- **input active**—Returns the input state of the specified line to the **Boolean** element of the **data** cluster. TRUE indicates the input line is active. FALSE indicates the input line is inactive.
 - **Note** Reading this attribute on a line configured for output returns error -70102 (NIMC_wrongIODirectionError).

Use this VI to read the attribute on a single digital I/O line. The digital I/O lines are divided among the axes.

Caution If this VI uses reflective memory, it should *not* be used for event checking. In this case, only use the information returned by this VI for display purposes. Refer to Reflective Memory VIs for more information.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following includes considerations you must make when you are using this VI with a 73xx controller:

- The PCI-7390 does not support the **configure as input** attribute.
- The PCI-7390 has dedicated direction I/O lines. On these controllers, the pin you read (input or output) is determined by the value specified in **line in** and **attribute**.

For example, to read the input active state attribute of General-Purpose Input Bit 1 on axis 2, call the Read Digital I/O Data VI with the following parameters:

```
axis in = 2
line in = 1
attribute = input active state
```

To read the output active state of General-Purpose Output Bit 1 on axis 2, call the Read Digital I/O Data VI with the following parameters:

```
axis in = 2
line in = 1
attribute = output active state
```

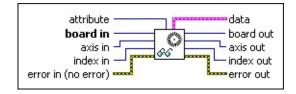
• 7330, 7340, and 7350 controllers have configurable direction I/O lines. On these controllers, the line number is translated to port and line number. For example, a call to Read Digital I/O Data on a four axis controller with eight digital I/O ports and

```
axis in = 2
```

Read Encoder Data

Reads position status or data from the selected encoder.

Device Compatibility



Details

attribute is the attribute you want to read. The following are valid attributes:

- position—Returns the quadrature count value of the encoder mapped to the selected axis to the **double** element of the **data** cluster.
- **index position**—Returns the captured index position after a <u>Find</u> Index search to the **double** element of the **data** cluster.
- **index capture occurred**—Returns the status to the **Boolean** element of the **data** cluster. True (On) indicates that the encoder index position capture occurred. False (Off) indicates that the encoder index position was not captured.

board in is a unique number assigned by Measurement & Automation
Explorer used to send and receive commands and data to or from a
specific NI motion controller.

- **axis in** is the axis to read with this VI.
- index in is the encoder index value. Valid value is 1 for the primary encoder.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or **FALSE** if not. If **status** is TRUE **code** is a popular or code. If

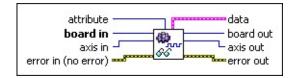
Use this VI to read status or data from the selected encoder.

	Caution If this VI uses reflective memory, it should <i>not</i> be used for event
	checking. In this case, only use the information returned by this VI for
	display purposes. Refer to <u>Reflective Memory VIs</u> for more information.
	Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.
S	about your controller.

Read Motion I/O Data

Reads status and data from the motion inputs and outputs.

Device Compatibility



Details Remarks

attribute is the attribute you want to read. The following are valid attributes:

- **forward limit active**—Returns the status to the **Boolean** element of the **data** cluster. TRUE indicates that the input is active and FALSE indicates that the input is inactive.
- **reverse limit active**—Returns the status to the **Boolean** element of the **data** cluster. TRUE indicates that the input is active and FALSE indicates that the input is inactive.
- **forward software limit active**—Returns the status to the **Boolean** element of the **data** cluster. TRUE indicates that the input is active and FALSE indicates that the input is inactive.
- **reverse software limit active**—Returns the status to the **Boolean** element of the **data** cluster. TRUE indicates that the input is active and FALSE indicates that the input is inactive.
- **home input active**—Returns the status to the **Boolean** element of the **data** cluster. TRUE indicates that the input is active and FALSE indicates that the input is inactive.
- **inhibit-in active**—Returns the status to the **Boolean** element of the **data** cluster. TRUE indicates that the input is active and FALSE

Use this VI to read the status and data from the motion inputs and outputs on the motion controller. You can read the status of these motion inputs and outputs at any time, regardless of whether they are enabled. The limit and home input status during a Find Reference move is undefined. Refer to Write Motion L/O Data for more information.

Caution If this VI uses reflective memory, it should *not* be used for event checking. In this case, only use the information returned by this VI for display purposes. Refer to Reflective Memory VIs for more information.

Note The active state of these signals determines if an On state is active high/active closed or active low/active open. Refer to Write Motion I/O Data for more information about active state and general information about the signals.

Note Refer to <u>VI Execution Times</u> for benchmark timing information about your controller.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The minimum pulse width for the inhibit-in input to be detected is approximately 1 ms.
- On the NI PCI-7390 controller, the **inhibit-in active** attribute does not return a valid value unless the inhibit-in signal is enabled.
- On the NI PCI-7390 controller, the **in-position active** attribute does not return a valid value unless the in-position signal is enabled.

NI SoftMotion Controller Considerations

The following includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

The NI SoftMotion Controller does not support the **in-position active**, **drive ready active**, or **inhibit-out active** attributes.

Advanced Write VIs

You can use the advanced write VIs for specialized I/O operations involving the digital I/O on a controller, for advanced motion I/O operations, or to configure the trajectory data for a move.

Click the icons for VI descriptions.



Write Trajectory Data

Write Capture Compare Data

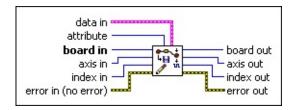
Write Motion I/O Data

Write Digital I/O Data

Write Capture Compare Data

Loads position compare or position capture data for the selected axis.

Device Compatibility



Details

attribute is the attribute to load. The following is the valid attribute:

 position compare pulse width—Sets the pulse width of the position compare signal.

board in is a unique number assigned by Measurement & Automation Explorer used to send and receive commands and data to or from a specific NI motion controller.

axis in is the axis to configure with this VI.

index in is the encoder to configure. Valid value is 1 for the primary encoder.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

Use this VI to load configuration parameters for position capture and compare or the motion controller. This function allows you to set all attributes on a per axis basis.

Use the **position compare pulse width** attribute to configure the pulse width for position compare outputs that trigger devices with specific minimum active pulse requirements.



Note A breakpoint across RTSI always generates an active high pulse of 200 ns duration.

Caution The position compare (breakpoint) frequency depends on the veloci and distance between position compare outputs and is limited by the pulse width. The position compare output frequency *must* meet the requirements specified below for correct operation.

The position compare output pulse must not overlap with the next position compare event. If the position compare output pulse overlaps with the next positi compare event, position compare generation is stopped and the position compare output is disabled. To recover from this situation change the pulse width, velocity or distance between position compare outputs. In addition, the position compare output frequency cannot be greater than the maximum value allowed for the specified position compare type. Therefore, the position compare output frequen in hertz, at the specified pulse width, in seconds, must conform to the following two conditions:

D - - '4' - --

		Position	
		Compare	Maximu
		(Breakpoint) Frequen
		Output Type	2
		Single	150 Hz
		Buffered,	
position compare frequency (hertz) $< \frac{1}{\text{pulse width (seconds)}}$	AND	PID rate of	2 kHz
pulse width (seconds)		62.5 µs to	
		250 μs	
		Buffered,	
		PID rate	1 l.II.
		greater	1 kHz
		than 250 µs	
		Periodic	4 MHz

e this VI to load configuration parameters for position capture and compare on motion controller. This function allows you to set all attributes on a per axis sis.

e the **position compare pulse width** attribute to configure the pulse width for sition compare outputs that trigger devices with specific minimum active pulse uirements.

Note A breakpoint across RTSI always generates an active high pulse of 200 ns duration.

Caution The position compare (breakpoint) frequency depends on the velocity and distance between position compare outputs and is limited by the pulse width. The position compare output frequency *must* meet the requirements specified below for correct operation.

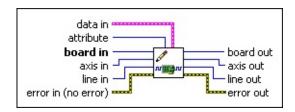
e position compare output pulse must not overlap with the next position npare event. If the position compare output pulse overlaps with the next position npare event, position compare generation is stopped and the position compare put is disabled. To recover from this situation change the pulse width, velocity, distance between position compare outputs. In addition, the position compare put frequency cannot be greater than the maximum value allowed for the ecified position compare type. Therefore, the position compare output frequency, nertz, at the specified pulse width, in seconds, must conform to the following productions:

		Position Compare (Breakpoint	Maximum) Frequency
		Output Type	2
		Single	150 Hz
tion compare frequency (hertz) $< \frac{1}{\text{pulse width (seconds)}}$	AND	Buffered, PID rate of 62.5 µs to 250 µs	2 kHz
		Buffered, PID rate greater than 250 µs	1 kHz
		Periodic	4 MHz

Write Digital I/O Data

Sets the attribute on a single digital I/O line.

Device Compatibility



Details Remarks

data in is a cluster of variables that contains the values for the selected **attribute**.

- **integer** is 32-bit integer data.
- **double** is 64-bit floating point data.
- **Boolean** is Boolean data.

Based on the **attribute**, the correct member of **data in** must be set as follows:



- For **output active state**, set the **Boolean** element of the **data in** cluster to TRUE to set the active state to active low/active open, and set it to FALSE to set the active state to active high/active closed.
- For **input active state**, set the **Boolean** element of the **data in** cluster to TRUE to set the active state to active low/active open, and set it to FALSE to set the active state to active high/active closed.
- For **output active**, set the **Boolean** element of the **data in** cluster to TRUE to configure the output line as active, and set it to FALSE to configure the output line as inactive.
- For configure as input set the Boolean element of the data in

Using This VI

Use this VI to set the attribute on a single digital I/O line. The digital I/O lines are divided among the axes.

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following includes considerations you must make when you are using this VI with a 73xx controller:

- The PCI-7390 does not support the **configure as input** attribute.
- The PCI-7390 has dedicated direction I/O lines. On these controllers, the pin you write to (input or output) is determined by the value specified in **line in** and **attribute**.

For example, to change the input active state of line 1 on axis 2, call Write Digital I/O Data with the following parameters:

```
axis in = 2
line in = 1
attribute = input active state
```

The active state is applied to axis 2 General-Purpose Input Bit 1.

To change the output active state of line 1 on axis 2, call Write Digital I/O Data with the following parameters:

```
axis in = 2
line in = 1
attribute = output active state
```

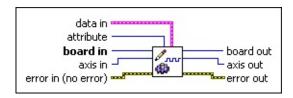
The active state is applied to axis 2 General-Purpose Output Bit 1.

• 7330, 7340, and 7350 controllers have configurable direction I/O lines. On these controllers, the line number is translated to port and line number. For example, a call to Write Digital I/O Data on a four axis controller with

Write Motion I/O Data

Loads configuration parameters for the motion inputs and outputs.

Device Compatibility



Details Remarks

data in is a cluster of variables that contains the values for the selected **attribute**.

integer is 32-bit integer data.

double is 64-bit floating point data.

Boolean is Boolean data.

Based on the **attribute**, the correct member of **data in** must be set as follows:

- For **forward limit enable** and **reverse limit enable**, set the **Boolean** element of the **data in** cluster to TRUE to enable the hardware limit, and FALSE to disable the hardware limit.
- For **forward software limit enable** and **reverse software limit enable**, set the **Boolean** element of the **data in** cluster to TRUE to enable the software limit, and FALSE to disable the software limit.
- For **home input enable**, set the **Boolean** element of the **data in** cluster to TRUE to enable the home input, and FALSE to disable home input.
- For forward limit active state, reverse limit active state, and home

Using This VI

Use this VI to load configuration parameters for the motion inputs and outputs on the motion controller. This VI allows you to set all attributes on a per axis basis.

Hardware limit inputs, home inputs, software position limits, inhibit inputs, inposition inputs, drive ready inputs, and inhibit outputs are enhancements on the NI motion controllers and are not required for basic motion control. With the exception of the Find Reference VI, you can operate all motion control VIs without enabling or using these signals. The Find Reference VI requires enabled limit and home inputs for operation. Find Reference does not require enabled software limits.

Caution National Instruments recommends using limits for personal safety, as well as to protect the motion system.

The active state for each hardware limit, home, inhibit input, in-position input, and inhibit output can be configured as either active low/active open or active high/active closed.

When configured as active low, the input or output is active when there is a low signal on the pin. Conversely, active high means that the input or output is active when there is a high signal on the pin.

Configuring an active state of active open or active closed does not correspond to the level of the signal on the input or output pin. Instead, an active open state means that the input or output is active when current is not flowing through the optocoupled input. Conversely, an active closed state means that the input or output is active when current is flowing through the optocoupled input.

Forward and Reverse Limits, and Home Inputs

The hardware limit inputs are typically connected to end-of-travel limit switches or sensors. An enabled limit input causes a halt stop on the axis when the input becomes active. Active limit inputs also prohibit attempts to start

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx controller:

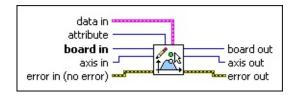
- 7330, 7340, and 7350 controllers do not support the **inhibit-in active state** and **in-position active state** attributes. Use <u>Write Digital I/O Data</u> to set the active state for these signals.
- 7330, 7340, and 7390 controllers do not support the **inhibit-out totem pole** attribute.
- 7390 controllers do not support the **inhibit-out active state** attribute. To set the active state on a 7390 use Write Digital I/O Data.
- The inhibit-in signal is software-timed latched at 1 ms. This signal must be active for approximately 1 ms to be detected by the controller.
- The following table lists the drive signals and their implementation on the controller:

Signal 1	Name	Signal		Controlle	r Support	
Controller	Drive	Direction	7330	7340	7350	7390
Inhibit-Out	Servo On	Output	Dedicated	Dedicated	Dedicated	Mappable
Inhibit-In	Servo Alarm	Input	Mappable	Mappable	Mappable	Dedicated
Drive Ready	Servo Ready	Input	Mappable	Mappable	Mappable	Mappable
In-Position	In- Position	Input	Mappable	Mappable	Mappable	Dedicated
Alarm Clear	Alarm Clear	Output	Mappable	Mappable	Mappable	Mappable

Write Trajectory Data

Configures trajectory generator parameters.

Device Compatibility



Details Remarks

data in is a cluster of variables that contains the values for the selected **attribute**.

integer is 32-bit integer data.

double is 64-bit floating point data.

Boolean is Boolean data.

Based on the **attribute**, the correct member of **data in** must be set as follows:

- For **move complete when deactivated**, set the **Boolean** element of the **data in** cluster to TRUE to add motor off status as a criteria for move complete, and set it to FALSE to remove motor off status.
- For **move complete when not moving**, set the **Boolean** element of the **data in** cluster to TRUE to add motor stop status as a criteria for move complete, and set it to FALSE to remove motor stop status.
- For **move complete after delay**, set the **Boolean** element of the **data in** cluster to TRUE to enable a user-defined delay as a criteria for move complete, and set it to FALSE to disable a user-defined delay.
- For **move complete time delay**, set **doubleData** in the TnimcData



Using This VI

Use this VI to configure trajectory generator parameters for reporting a move complete. When a move is complete on an axis, the corresponding bit in the Move Complete Status (MCS) register is set. Refer to Read Move Complete Status for information about reading the MCS register.

If **move complete when deactivated** is true, any condition that causes the axis to turn its motor off (a kill or following error trip) satisfies this requirement for Move Complete.

If the **move complete when not moving** is true, the axis must be logically stopped for the move to be considered complete.

If **move complete after delay** is true, the axis must wait a user-defined delay after the other criteria are met before the move is considered complete. The **move complete time delay** attribute is typically used to wait the mechanical settling time so that a move is not considered complete until vibrations in the mechanical system have damped out. It also can be used to compensate for PID pull-in time due to the integral term. This pull-in is typically at velocities below the Run/Stop threshold.

Note You can use the **move complete after delay** parameter to guarantee a minimum time for the False state. The status transitions from Complete to Not Complete at the start of a move and stays in the Not Complete state for at least this delay time even in the case of a zero distance move.

If **move complete when in range** is true, the axis checks its final stopping position versus its target position and only sets the Move Complete status if the absolute value of the difference is less than **move complete range distance**.

Finally, if **move complete when in position active** is true, the in-position input signal is used as a criteria for move complete, and the move is only complete if the in-position signal is active.

A non-zero value for **move complete minimum active time** guarantees the

Remarks

This section includes information about how the behavior of this VI differs among the controllers that support it.

NI 73xx Controller Considerations

The following list includes considerations you must make when you are using this VI with a 73xx motion controller:

- The NI 7344 does not support the **move complete when in position active** attribute.
- 7330, 7340, and 7350 controllers use <u>Configure Motion I/O Map</u> to map the in-position input to a general-purpose I/O line.

NI SoftMotion Controller Considerations

The following includes considerations you must make when you are using this VI with the NI SoftMotion Controller:

The NI SoftMotion Controller does not support the **move complete when in position active** attribute.

Advanced Methods

You can use the advanced methods for specialized I/O and move operations, and for specialized communications involving the controller and drive or host.

Click the icons for VI descriptions.



Axis Straight Line Move

Configure Motion I/O Map

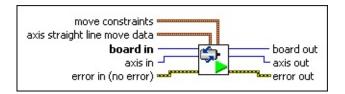
Reset Motion Controller

Clear Faults

Axis Straight Line Move

Performs a straight line move on an axis.

Device Compatibility



Details

axis straight line move data is a cluster containing straight line move information.

start mode is the start mode for the move:

- **do not start**—Do no start the straight line move at this time.
- **start**—Start the straight line move with the configured parameters.
- **position mode** selects the position mode for the move. Refer to Details for more information about the available position modes.
 - absolute—Use absolute position mode in the move.
 - **relative**—Use relative position mode in the move.
 - **velocity**—Use velocity mode in the move.

target position is the target position for the straight line move in counts or steps.

move constraints is a cluster containing move constraint information:

- **velocity** sets the velocity for the move in counts/s or steps/s.
- **acceleration** sets the acceleration for the move in counts/ s^2 or steps/ s^2 .
- **deceleration** sets the deceleration for the move in counts/ s^2 or steps/ s^2 .

acceleration jerk sets the acceleration jerk for the move in counts/s³ or steps/s³.

Using This VI

Use this VI to perform a straight line move on an axis. The position modes are described in the following sections:

Absolute Position Mode

In absolute position mode, target positions are interpreted with respect to an origin, reference, or zero position. The origin is typically set at a home switch, end of travel limit switch, or encoder index position. An absolute position move uses the specified values of acceleration, deceleration, and velocity to complete a trajectory profile with an ending position equal to the specified absolute target position.

Caution Any single move is limited to between -2^{31} and 2^{31} –1 counts or steps. An error is generated if you exceed this limit by specifying a target position too far from the current position.

The length of an absolute move depends upon the specified position and the current position when the move is started. If the target position is the same as the current position, no move occurs.

Relative Position Mode

In relative position mode, if a relative position move is started while motion is not in progress, specified target positions are interpreted with respect to the current position at the time the value is specified. A relative position move uses the specified values of acceleration, deceleration, and velocity to complete a trajectory profile with an ending position equal to the sum of the specified relative target position and the starting position.

If a relative move is started while motion is in progress, the new target position is calculated with respect to the target position of the move already in progress (considered to be the new starting position), as if that move had already completed successfully. Motion continues to the new relative position, independent of the actual position location when the new move is started.

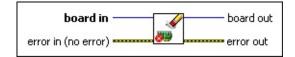
Velocity Mode

In velocity mode, the axis moves at the specified velocity until you execute a Stop Motion VI, a limit is encountered, or a new velocity is specified and you

Clear Faults

Clears all modal errors and drive faults on all axes.

Device Compatibility



Details

board in is a unique number assigned by Measurement & Automation Explorer used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

board out is provided for flow control. You can string together NI-Motion VIs by wiring the **board out** terminal of one VI to the **board in** terminal of the next VI.

error out contains error information. If error in indicates an error, error

Using This VI

Use this VI to clear all modal errors and drive faults on all axes. This VI pulses the alarm clear line on all axes that have mapped the alarm clear signal using Configure Motion I/O Map.



Note Refer to Write Motion I/O Data for more information about the alarm clear signal and to change the alarm clear pulse width.

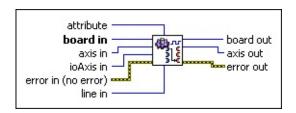
The inhibit-in signal is connected to the Alarm signal of the drive. Monitor the status of the inhibit-in signal to determine if there are any drive faults. Correct the faults, then use this VI to clear the alarm state of the drive.

The <u>Initialize Controller</u> VI automatically calls this VI if the controller is in a power-up state. If you call Initialize Controller and the controller is not in a power-up state, this VI is not called.

Configure Motion I/O Map

Maps a general purpose I/O line to an alternate function of an axis.

Device Compatibility



Details

attribute is the action to map to a digital I/O line. The following are valid attributes:

- **default input**—Maps a general purpose I/O line as the default input line.
- **default output**—Maps a general purpose I/O line as the default output line.
- **shutdown**—Maps a general purpose input line as the shutdown (E-Stop) line. Refer to <u>Enable Shutdown</u> for more information about the shutdown (E-Stop) functionality of the controller.
- **inhibit-out**—Maps a general purpose output line as an inhibit-out line. Refer to <u>Configure Inhibit Output</u> for more information about the inhibit outputs.
- **alarm clear**—Maps a general purpose output line as an alarm clear line. Refer to <u>Write Motion I/O Data</u> for more information about alarm clear.
- **inhibit-in**—Maps a general purpose input line as an inhibit-in line.

Using This VI

This VI maps a general-purpose I/O line to an alternate function of an axis such as inhibit output. For example, to map axis 3 general-purpose output 1 to the inhibit output of axis 3, call the VI with the following parameter values:

```
axis in = 3
attribute = inhibit-out
ioAxis in = 3
line in = 1
```

Notes

- For all attributes except for **shutdown**, the **axis in** input must be the same as the **ioAxis in** input.
- The action **shutdown** applies to the device level. The **axis in** input is ignored. Furthermore, only one general-purpose input can be mapped to this attribute.



- 7330, 7340, and 7350 controllers do not support the **inhibit-out** and **shutdown** attributes because these controllers have dedicated pins for these signals.
- 7390 controllers do not support the **inhibit-in** and **in-position** attributes because this controller has dedicated pins for these signals.

The following table lists the drive signals and their implementation on the controller:

Signa	l Name	Signal		Controlle	r Support	
Controller	Drive	Direction	7330	7340	7350	7390
Inhibit-Out	Servo On	Output	Dedicated	Dedicated	Dedicated	Mappable
Inhibit-In	Servo Alarm	Input	Mappable	Mappable	Mappable	Dedicated
Drive Ready	Servo Ready	Input	Mappable	Mappable	Mappable	Mappable

Reset Motion Controller

Resets the motion controller to the power-up state.

Device Compatibility



Details

board in is a unique number assigned by Measurement & Automation Explorer used to send and receive commands and data to or from a specific NI motion controller.

error in (no error) describes error conditions that occur before this VI runs. The default input of this cluster is no error. If an error already occurred, this VI returns the value of **error in** in **error out**. The VI runs normally only if no incoming error exists. Otherwise, the VI passes the **error in** value to **error out**. The **error in** cluster contains the following parameters:

status is TRUE if an error occurred before this VI was called, or FALSE if not. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is zero or a warning code.

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

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

board out is provided for flow control. You can string together NI-Motion VIs by wiring the **board out** terminal of one VI to the **board in** terminal

Using This VI

Use this VI to reset the motion controller to the power-up state. After resetting the controller, you must call <u>Initialize Controller</u> or <u>Clear Power Up Status</u> to use the controller.

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 TargetBoxTM and Target Language CompilerTM 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.

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 for technical support and professional services:

- <u>Support</u>—Online technical support resources at ni.com/support include the following:
 - Self-Help Resources—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable <u>KnowledgeBase</u>, <u>product manuals</u>, 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 <u>NI Discussion Forums</u> at ni.com/forums. National Instruments Applications Engineers make sure every question receives an answer.

For information about other <u>technical support options</u> in your area, visit ni.com/services or <u>contact</u> your local office at ni.com/contact.

- <u>Training and Certification</u>—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.
- <u>System Integration</u>—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.

NI 7330 and 7340 Timing Information

The following table lists the execution time for selected VIs, as well as for the different attributes of those VIs, if applicable. Also included is timing information for axes or vector spaces, if both are supported with a given VI.



Note The times listed are benchmark times and are not a guarantee of VI 🔊 execution times on your system. Times do not take into account other operations, onboard programs, or other factors that slow performance.

VI	VI Return Time (ms) using 1.47 GHz Processor [‡]	VI Return Time (ms) using 700 MHz Processor [‡]
Load Target Position	1.1	1.2
Read Position	1.7	1.9
Read Motion I/O Data		
Forward Limit Active		
Reverse Limit Active	1.3	1.4
Forward Software	1.3	1.4
Limit Active	1.4	1 F
Reverse Software Limit Active	1.4	1.51.5
Home Input	1.6	1.7
Active	1.6	1.7
Inhibit-In Active	1.6	1.7

In-Position Active	e 1.6	1.7
Drive Ready Active	1.6	1.7
Inhibit-Out Active	е	
Read I/O Port	1.3	1.4
Set I/O Port MOMO	1.1	1.2
Load Vector Space Position	1.1	1.2
Read Velocity		
axis	1.0	1.0
vector	1.8	1.8
	1.9	1.9
Read Velocity in RPM		
axis	2.0	2.1
vector	2.1	2.2
Load Velocity		
axis	1.2	1.3
vector		
	1.2	1.3
Load Velocity in RPM		

axis

vector	2.7	2.8
	2.8	2.9
Read Move Complete Status	0.1	0.2
Configure Vector Space	1.3	1.4
Check Move Complete Status	1.6	1.7
Check Blend Complete Status	1.4	1.5
Read Communication Status	0.1	0.2
Initialize Controller	607.1	607.1
Read Digital I/O Data		
Output Active State	1.4	1.5
Input Active State		1.5
Output Active	1.4	1.5
Input Active	1.4	1.5
Configure As Input	1.4	1.5
Read Following Error	1.8	1.9

 $^{^\}dagger\,$ An AMD Athlon 1.47 GHz PC with 256 MB of RAM was used for timing

tests. Tests run in "above normal priority" execution to achieve the most accurate results.

[‡] An Intel Pentium 3 700 MHz PC with 386 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

NI 7334, 7342, and 7344 Timing **Information**

The following table lists the execution time for selected VIs, as well as for the different attributes of those VIs, if applicable. Also included is timing information for axes or vector spaces, if both are supported with a given VI.



Note The times listed are benchmark times and are not a guarantee of VI Nexual execution times on your system. Times do not take into account other operations, onboard programs, or other factors that slow performance.

VI	VI Return Time (ms) using 1.47 GHz Processor [‡]	VI Return Time (ms) using 700 MHz Processor [‡]
Load Target Position	1.1	1.2
Read Position	1.9	1.9
Read I/O Port	1.6	1.7
Set I/O Port MOMO	1.1	1.2
Load Vector Space Position	1.3	1.4
Read Velocity		
axis	1.9	2.0
vector	2.0	2.1
Read Velocity in RPM		
axis	2.1	2.2

vector	2.1	2.2
Load Velocity		
axis	1.5	1.6
vector	1.2	1.3
Load Velocity in RPM		
axis	3.4	3.5
vector	1.9	1.9
Read Move Complete Status	0.1	0.1
Configure Vector Space	1.7	1.8
Check Move Complete Status	1.5	1.6
Check Blend Complete Status	1.6	1.7
Read Communication Status	0.1	0.1
Initialize Controller	620.4	620.4
Read Following Error	1.8	1.8

[†] An AMD Athlon 1.47 GHz PC with 256 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

[‡] An Intel Pentium 3 700 MHz PC with 386 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

NI 7350 Timing Information

The following table lists the execution time for selected VIs, as well as for the different attributes of those VIs, if applicable. Also included is timing information for axes or vector spaces, if both are supported with a given VI.



Note The times listed are benchmark times and are not a guarantee of VI execution times on your system. Times do not take into account other operations, onboard programs, or other factors that slow performance.

VI	VI Return Time (ms) using 1.47 GHz Processor [†]	VI Return Time (ms) using 700 MHz Processor [‡]
Load Target Position	1.1	1.2
Read Position	0.1	0.1
Read Axis Data		
Position	0.1	0.2
Velocity	0.1	0.2
Following Error	0.1	0.2
Read Axis Status		
Axis Active		
Move Complete	0.1	0.2
Profile Complete	0.1	0.2
Blend Complete	0.1	0.2
Following Error	0.1	0.2

Velocity Threshold	0.1	0.2
Reached	0.1	0.2
Moving	0.1	0.2
Direction Forward] 0.1	0.2
Read All Axis Data	0.2	0.3
Read All Axis Status	0.1	0.2
Read Coordinate Data		
Velocity	0.2	0.3
Following Error	0.2	0.3
Read Coordinate Position	0.3	0.3
Read Encoder Data		
Position	0.1	0.2
Index Position	1.6	1.7
Index Capture	0.1	0.2
Read Motion I/O Data		
Forward Limit Active		

Reverse Limit

Active	0.1	0.2
Forward Software	0.1	0.2
	0.1	0.2
Reverse Software Limit Active	0.1	0.2
Home Input	0.1	0.2
Active	1.5	1.6
Inhibit-In Active	1.5	1.6
In-Position Active	1.5	1.6
Drive Ready Active	1.5	1.6
Inhibit-Out Active	2	
Read Capture Compare Data		
_	1.6	1.7
Compare Data	1.6 0.1	1.7 0.2
Compare Data Captured Position	1.0	
Compare Data Captured Position Capture Status Compare Status Read I/O Port	0.1	0.2
Compare Data Captured Position Capture Status Compare Status	0.1 0.1	0.2
Compare Data Captured Position Capture Status Compare Status Read I/O Port Set I/O Port	0.1 0.1 0.2	0.2 0.2 0.3
Captured Position Capture Status Compare Status Read I/O Port Set I/O Port MOMO Load Vector	0.10.10.21.1	0.20.20.31.2

vector	0.1	0.2
	0.2	0.3
Read Velocity in RPM		
axis	0.1	0.2
vector	0.2	0.3
Load Velocity		
axis	1.3	1.4
vector	1.1	1.4
	1.1	1,2
Load Velocity in RPM		
	3.1	3.2
RPM	3.1 1.8	3.2 1.8
RPM axis		
RPM axis vector Read Move	1.8	1.8
RPM axis vector Read Move Complete Status Configure Vector	1.8 0.1	1.8 0.1
RPM axis vector Read Move Complete Status Configure Vector Space Check Move	1.80.11.8	1.8 0.1 1.8

Initialize Controller Read Coordinate Status	1218.0	1218.0
Move Complete Profile Complete Blend Complete Following Error Exceeded	0.20.20.20.2	0.30.30.30.3
Read Digital I/O Data		
Output Active State	1.3	1.3
Input Active State	1.3	1.3
Output Active	0.2	0.2
Input Active	0.2	0.2
Configure As Input	1.3	1.3
Read Following Error	2.1	2.1

 $^{^{\}dagger}$ An AMD Athlon 1.47 GHz PC with 256 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

[‡] An Intel Pentium 3 700 MHz PC with 386 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

NI 7390 Timing Information

The following table lists the execution time for selected VIs, as well as for the different attributes of those VIs, if applicable. Also included is timing information for axes or vector spaces, if both are supported with a given VI.



Note The times listed are benchmark times and are not a guarantee of VI execution times on your system. Times do not take into account other operations, onboard programs, or other factors that slow performance.

VI	VI Return Time (ms) using 1.47 GHz Processor [†]	VI Return Time (ms) using 700 MHz Processor [‡]
Load Target Position	1.1	1.2
Read Position	0.1	0.2
Read Axis Data		
Position	0.1	0.2
Velocity	0.1	0.2
Following Error	0.1	0.2
Read Axis Status		
Axis Active		
Move Complete	0.1	0.2
Profile Complete	0.1	0.2
Blend Complete	0.1	0.2
Following Error	0.1	0.2

Velocity Threshold	0.1	0.2
Reached	0.1	0.2
Moving	0.1	0.2
Direction Forward] 0.1	0.2
Read All Axis Data	0.2	0.3
Read All Axis Status	0.1	0.2
Read Coordinate Data		
Velocity	0.2	0.3
Following Error	0.3	0.4
Read Coordinate Position	0.2	0.3
Read Encoder Data		
Position	0.1	0.2
Index Position	1.5	1.6
Index Capture	0.1	0.2
Read Motion I/O Data		
Forward Limit Active		

Reverse Limit

Active	0.1	0.2
Forward Software	0.1	0.2
Limit Active	0.1	0.2
Reverse Software Limit Active	0.1	0.2
Home Input	0.1	0.2
Active	1.3	1.4
Inhibit-In Active	1.3	1.4
In-Position Active	1.3	1.4
Drive Ready Active	1.3	1.4
Inhibit-Out Active		
Read Capture		
Compare Data		
_	1.5	1.6
Compare Data	1.5 0.1	1.6 0.2
Compare Data Captured Position	1.5	
Compare Data Captured Position Capture Status	0.1	0.2
Compare Data Captured Position Capture Status Compare Status	0.10.1	0.2
Compare Data Captured Position Capture Status Compare Status Read I/O Port Set I/O Port	0.10.10.2	0.20.20.3
Captured Position Capture Status Compare Status Read I/O Port Set I/O Port MOMO Load Vector	0.10.10.21.0	0.20.20.31.1

vector	0.1	0.2
	0.2	0.3
Read Velocity in RPM		
axis	0.1	0.2
vector	0.2	0.3
Load Velocity		
axis	1.2	1.3
vector	1.1	1.2
	1.1	1.4
Load Velocity in RPM		
	2.7	2.7
RPM	2.71.5	2.7 1.6
RPM axis		
RPM axis vector Read Move	1.5	1.6
RPM axis vector Read Move Complete Status Configure Vector	1.5 0.1	1.6 0.1
RPM axis vector Read Move Complete Status Configure Vector Space Check Move	1.50.11.4	1.60.11.5

Initialize Controller	565.5	608.5
Read Coordinate Status		
Move Complete		
Profile Complete	0.2	0.3
•	0.2	0.3
Blend Complete	0.2	0.3
Following Error Exceeded	0.2	0.3
Read Digital I/O Data		
Output Active		
State	1.3	1.4
Input Active State	1.3	1.4
Output Active	0.2	0.3
Input Active	0.2	0.3
Read Following Error	2.1	2.1

 $^{^\}dagger$ An AMD Athlon 1.47 GHz PC with 256 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

[‡] An Intel Pentium 3 700 MHz PC with 386 MB of RAM was used for timing tests. Tests run in "above normal priority" execution to achieve the most accurate results.

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 6555 7838
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	385 (0) 9 725 72511
France	33 (0) 1 48 14 24 24
Germany	49 89 7413130
India	91 80 41190000
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	7 495 783 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