

NI Digital Waveform Generator/Analyzer Help

September 2007, 370520J-01

This help file explains fundamental and advanced concepts necessary for using a National Instruments digital waveform generator/analyzer. NI digital waveform generator/analyzers include the following devices:

- NI PXI/PCI-6541 (NI 6541)
- NI PXI/PCI-6542 (NI 6542)
- NI PXI/PCI-6551 (NI 6551)
- NI PXI/PCI-6552 (NI 6552)
- NI PXI/PCI-6561 (NI 6561)
- NI PXI/PCI-6562 (NI 6562)

NI digital waveform generator/analyzers include the NI-HSDIO driver with an intuitive, powerful application programming interface (API) for highspeed digital applications. NI-HSDIO is Windows-compatible and provides an API for LabVIEW, LabWindows™/CVI™, and other textbased development environments.

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

For more information about the NI digital waveform generator/analyzer or this help file, refer to the following topics and sections:

<u>Conventions</u>—formatting and typographical conventions in this help file

Related Documentation

<u>Glossary</u>

<u>Fundamentals</u>—terminology and concepts common to NI high-speed digital I/O applications

<u>Devices</u>—information specific to the different families of NI digital waveform generator/analyzers

Integration and System Considerations—information on creating a hardware system with the NI digital waveform generator/analyzer

<u>Programming</u>—programming information for the NI-HSDIO API

Important Information

Technical Support and Professional Services

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

© 2003–2007 National Instruments Corporation. All rights reserved.

Conventions

The following conventions appear in this help file:

- < > Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, DIO <0..3>.
- [] 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.
- The symbol indicates that the following text applies only to a specific product, a specific operating system, or a specific software version.
- 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.

monospace Bold text in this font denotes the messages and responses
bold that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.

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

Related Documentation

Most NI digital waveform generator/analyzer manuals also are available as PDFs. You must have Adobe Acrobat Reader with Search and Accessibility 5.0.5 or later installed to view the PDFs. Refer to the <u>Adobe</u> <u>Systems Incorporated Web site</u> to download Acrobat Reader. Refer to the <u>National Instruments Product Manuals Library</u> for updated documentation resources.

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

- <u>NI Digital Waveform Generator/Analyzer Getting Started Guide</u> (PDF)
- Specifications for the NI PXI/PCI-6541/6542 (PDF)
- NI PXI/PCI-6551/6552 Specifications (PDF)
- <u>NI PXI/PCI-6561/6562 Specifications</u> (PDF)
- *NI Script Editor Help*, available by launching the NI Script Editor (included with NI-HSDIO) and selecting **Help**»**NI Script Editor Help**.
- Maintain Forced-Air Cooling Note to Users
- <u>National Instruments High-Speed Digital ATE and Stimulus</u> <u>Response Features</u>
- <u>NI-HSDIO Instrument Driver Readme</u>
- Electrical Characteristics of Low-Voltage Differential Signaling (LVDS) Interface Circuits, Revision A, 2001 edition

Fundamentals

Expand this book for more information about concepts and terminology used in this help file.

Voltage Levels

Digital devices have voltage levels specified for normal operation of their acquisition and generation operations. Voltage levels define how a device determines a valid logic state (logic high level or logic low level).

Voltage levels are defined differently for <u>single-ended</u> and <u>differential</u> devices.

Single-Ended Voltage Levels

For single-ended devices, voltage levels are usually specified in terms of the voltage placed on the output terminal when driving a high level signal or when driving a low level signal, and by the voltage required on the input terminal for the signal to be recognized as a high or low level signal.

For the NI digital waveform generator/analyzers, the single-ended voltage levels are defined as follows:

- Generation Voltage High Level—When configured for active drive generation, this is the voltage produced at the channel electronics when the Pattern Generation Engine generates a binary one. When configured for <u>open collector</u> generation, Generation Voltage High Level is equivalent to setting the data channel to a high-impedance state.
- **Generation Voltage Low Level**—The voltage produced at the channel electronics when the Pattern Generation Engine generates a binary zero.
- Acquisition Voltage High Level—The voltage level at or above which the Pattern Acquisition Engine senses a binary one.
- Acquisition Voltage Low Level—The voltage level at or below which the Pattern Acquisition Engine senses a binary zero.

Note On the NI 655*x* devices these levels are <u>configurable</u>.

When connecting an NI digital waveform generator/analyzer to a device under test (DUT), you must ensure that the interface voltage levels are compatible. The relationship between the single-ended voltage levels and the DUT voltage levels are shown in the following figure.



M

To accurately communicate with a DUT, configure the NI module such that the following conditions are met:

Generation Voltage High Level ³ DUT ⊻_{IH}

- Generation Voltage Low Level £ DUT \underline{V}_{IL}
- Acquisition Voltage High Level £ DUT \underline{V}_{OH}
- Acquisition Voltage Low Level ³ DUT ⊻_{OL}
- Acquisition Voltage High Level > Acquisition Voltage Low Level

The extra margin between the voltage level being driven by the source and the voltage level required at the destination is known as the noise immunity margin (NIM). The NIM indicates the amount of noise tolerable on the connecting cable with a data bit being received in correctly. The total NIM is computed by the following formula:

NIM = [min (|Generation Voltage High - DUT V_{IH} |, |Generation Voltage Low - DUT V_{IL} |,

 $[DUT V_{OH} - Acquisition Voltage High], [DUT V_{OL} - Acquisition Voltage Low])]$

The NI SHC68-C68-D2 shielded cable for single-ended high-speed digital signals provides excellent protection against external noise sources. However, if your system operates in a particularly noisy environment and is having difficulty with incorrect data bits, consider increasing the NIM, if possible.

Differential Voltage Levels

Unlike single-ended signals, differential signals are transmitted in parity. That is, instead of a single conductor referenced to ground, two conductors, referenced to each other, transmit data. The digital driver still drives two voltages, as in the single-ended case. The receiver, however, interprets the signals based on the voltage difference between the pair of signals — not on a reference to ground. For the differential digital signal to be interpreted as a binary 0, the signal must be less than its complementary signal by more than a particular value, shown as V_{TH} in the figure below. V_{TH} varies and is specified by the particular logic family.

Since the conductors are referenced and transmitted together, by using differential signals you can achieve higher noise immunity in your signals. A benefit to this is that you can allow much smaller signal swings, so you can transmit data farther, faster, and at a fraction of the power

Since differential signals reference a positive signal to a complementary signal, across a specified differential impedance, voltage levels are typically specified from a differential, rather than an absolute, perspective (depending on the standard). For example, the absolute voltage levels of an LVDS transmission pair across a 100 Ω differential terminating impedance may have a V_{OH} of 1.4 V on the positive conductor and a V_{OL} of 1.1 V on the complementary conductor. The differential voltage would then be called out as the difference between the two—300 mV, shown as V_{OD} in the following figure. There is, however, a common-mode component of the signal, shown as V_{OS} in the following figure, that is also called out by most differential specifications and is referenced to common.

For differential NI digital waveform generator/analyzers, the voltage levels are defined as follows:

- **Differential Output Voltage (V_{OD})**—The difference in voltage between the positive and complementary conductors of a differential transmission. You can think of this value as the difference of the two conductors.
- Offset Voltage (V_{OS})—The common mode of the differential signal. You can think of this value as the average of the two conductors.

- **Threshold Voltage (V_{TH})**—The differential voltage threshold at which the receiver registers a valid logic state.
- Input Voltage Range (V_{RANGE})—The absolute voltage, referenced to common, allowed by the receiver.

When connecting a differential NI digital waveform generator/analyzer to a DUT, ensure that the interface voltage levels are compatible. The relationship between the differential NI device voltage levels and the DUT voltage levels are shown in the following figure.



The total <u>NIM</u> is computed by the following formula:

NIM = [min (|Generation V_{OD} - DUT V_{TH} |,|DUT V_{OD} - Acquisition V_{TH} |)]

The NI SHB12X-B12X shielded cable for differential high-speed digital signals provides excellent protection against external noise sources.

Digital Logic

Expand this book for more information about digital logic families and states.

Logic Families

Logic families are groups of logic circuits with standardized <u>voltage levels</u> that constitute a valid logic state. All circuits within a logic family are compatible with other circuits within that family, since they share the same characteristics.

Logic families can be defined by <u>single-ended voltage levels</u>, such as the 1.8 V, 2.5 V, 3.3 V, or 5.0 V logic families, or by <u>differential voltage levels</u>, such as LVPECL and LVDS.

Refer to <u>Configuring Voltage Levels</u> for more information about using NI-HSDIO to select logic families and configure your voltage levels.

Note If you use the NI 655*x* devices, you can configure custom voltage levels for your operation.

Related Topics:

- Logic Families (NI 654x)
- Logic Families (NI 655x)
- Logic Families (NI 656x)

Single-Ended Logic Families

Includes TTL, LVTTL, CMOS, and LVCMOS

Single-ended logic families use standardized <u>single-ended voltage levels</u> to interpret the voltage swing between the voltage driven by the device and ground as either a binary one or a zero.

Examples of the voltage levels for common single-ended logic families are shown in the following table.

Logic Family	Voltage Range
CMOS	0 to 5 V
TTL	0 to 5 V
LVTTL	0 to 3.3 V
LVCMOS	0 to 3.3 V

The single-ended logic families for NI digital waveform generator/analyzers are named after the voltage the NI device interprets as a binary 1 when configured for <u>active drive</u> generation. These logic families include 1.8V, 2.5V, 3.3V, and 5.0V.

Differential Logic Families

Differential logic families use <u>differential voltage levels</u> to measure the voltage difference between a pair of wires and interpret the difference as a binary one or zero.

Low-Voltage Differential Signaling (LVDS)

Low-voltage differential signaling (LVDS) is a low-noise, low-power, lowamplitude differential method for high-speed digital data transfer.

The following figure shows a diagram of a typical LVDS circuit.



A current source at the driver provides approximately 3.5 mA of current. The direction of the current depends on whether the driver drives a logic high level or low level. When the current reaches the receiver, a 100 Ω terminating resistor connects the two ends of the differential transmission line to provide a return path for the current. A voltage of approximately 350 mV (3.5 mA x 100 Ω) is established across the two input terminals of the receiver. The differential voltage at the receiver is either positive or negative, depending on the direction of the current. The receiver recognizes a positive differential voltage signal as a logic high level (binary 1) and a negative differential voltage as a logic low level (binary 0).

The electrical characteristics of an LVDS signal offers many performance improvements compared to single-ended standards. For example, since the received voltage is a differential between two signals, the voltage swing between the logic high level and low level state can be smaller, allowing for faster rise and fall times and thus faster toggle and data rates. Also, as with LVPECL circuits, the differential receiver is less susceptible to common-mode noise than single-ended transmission methods.

The LVDS standard defines the electrical aspects of this type of data transmission. The standard defines driver and receiver electrical characteristics only. The standard does not create protocol, interconnect, or connector definitions because these aspects are application-specific. For more information on LVDS, refer to the <u>Understanding LVDS for</u> <u>Digital Test Systems</u> application note on ni.com.

Note Refer to the ANSI/TIA/EIA-644-A electrical characteristics standard, *Electrical Characteristics of Low-Voltage Differential Signaling (LVDS) Interface Circuits*, Revision A, 2001 edition for more information about the LVDS standard.

Low-Voltage Positive Emitter-Coupled Logic (LVPECL)

Emitter-Coupled Logic circuits use a design with transistors that steer current through gates to compute logical functions. Because the transistors are always in the active region, they can change state very rapidly, so ECL circuits can operate at very high speeds.

LVPECL circuits are a type of ECL circuit that require a pair of signal lines for each channel. The differential transmission scheme is less susceptible to common-mode noise than single-ended transmission methods. LVPECL circuits are designed for use with supply voltages of 3 V or 3.3 V.

Digital Logic States

Test engineers can choose from a number of different digital I/O instruments with a range of features for communication and test applications. Beyond the basic capabilities of driving a digital pattern of 1's and 0's, digital instruments often support waveforms that can include some or all of the logic states shown in the following table.

	Logic State	Drive Data	Expected Response
Drive States	0	Logic Low	Don't Care
	1	Logic High	Don't Care
	Z	Disable	Don't Care
Compare States	L	Disable	Logic Low
	Н	Disable	Logic High
	Х	Disable	Don't Care

The six logic states control the voltage driver and, if supported, the compare engine of the digital tester on a per clock cycle basis. The Drive states specify what stimulus data the tester drives on a particular channel or when to disable the voltage driver (referred to as the tristate or high-impedance state). Compare states indicate the expected response from the DUT. These six logic states make it possible to perform bidirectional communication and real-time hardware comparison of acquired response data.

The <u>NI 655x</u> digital waveform generator/analyzer supports all six logic states shown in the preceding table, allowing the device to perform bidirectional stimulus/response test options with hardware comparison. Other NI digital waveform generator analyzers can perform simultaneous generation and acquisition using 1's and 0's, but they do not support bidirectional operation.

Related Topics:

- Hardware Comparison (NI 655x)
- <u>Comparing Response Data with Expected Data</u>

Hysteresis

Hysteresis refers to the difference in voltage levels between the detection of a transition from a logic low level to a logic high level and the transition from a logic high level to a logic low level. Refer to the following figure illustrating hysteresis.



All digital logic devices have some level of hysteresis on their digital inputs. The magnitude of a particular device's hysteresis can be determined by the following formula:

Hysteresis \approx V_{IH} - V_{IL}

On a rising edge of the digital signal on the input, the device detects a transition from a logic low to a logic high at $\underline{V}_{\underline{IH}}$. Conversely, the device detects a transition from a logic high to a logic low when the voltage at the input of the device crosses $\underline{V}_{\underline{IL}}$.

Hysteresis is a useful property for digital devices because it provides some amount of natural immunity to high-frequency noise in your digital system. This noise, often caused by reflections from the high edge rates of logic level transitions, could cause false transition detections by the digital device if only a single voltage threshold determined a change in logic state. This phenomenon is more clearly illustrated in the following figure.



In this figure, after applying hysteresis the first sample is acquired as a logic low level. The second sample is also a logic low level because the signal has not yet crossed the logic high level threshold. The third and

fourth samples are logic high level, and the fifth is logic low level.

For devices with fixed voltage thresholds, the NIM and hysteresis of your system are determined by your choice of system components. Some NI devices allow you to control both your system NIM and hysteresis. Both system NIM and hysteresis give your system levels of noise immunity, but for a specific logic family, there is always a trade-off between these two—the larger the hysteresis, the smaller the NIM, and vice versa. To determine how to set your voltage thresholds, carefully examine the signal quality in your system to determine whether you need more noise immunity from your valid logic levels (greater NIM) or need more noise immunity on your logic level transitions (greater hysteresis).

AC Waveform Characteristics

The following terms describe some of the common AC waveform characteristics.

Rise time and fall time—*Rise time* (t_{rise}) is the time that it takes a signal to rise from 20% to 80% of the voltage between the voltage low level and the voltage high level. *Fall time* (t_{fall}) is the time that it takes a signal to fall from 80% to 20% of the voltage between the voltage low level and the voltage high level.

The following figure illustrates rise and fall time.



• **Preshoot and Overshoot**—Preshoot and overshoot are peak distortions preceding (preshoot) or following (overshoot) an edge. The following figure shows an example of preshoot and overshoot on a signal.



Note Together, overshoot, preshoot, and undershoot are called *aberrations*.



• Settling time—Settling time (t_S) is the time required for an amplifier, relay, or other circuits to reach a stable mode of operation. In the context of signal acquisition, the settling time for a full-scale step is the amount of time required for a signal to reach a certain accuracy and stay within that accuracy range.

The following figure illustrates settling time on a digital signal.



• **Duty cycle**—For clock signals, the percentage of the waveform period that the waveform is at logic high level.

The following figure shows the difference between two waveforms with different duty cycles. Notice that the 30% duty-cycle waveform is at logic high level for less time than the 50% duty cycle waveform.

50% Duty Cycle	
30% Duty Cycle	

Termination

Unlike systems designed for lower speed applications, in high-speed digital systems, simple passive circuit elements like wires, cables, and chip PCB interconnections can significantly affect signal quality. High-speed digital edges contain frequency components that are several times the effective toggle rate of that signal. For example, a digital edge with a rise time of 1.5 ns contains significant energy in frequencies up to 333 MHz, regardless of toggle rate. When designing systems using NI digital waveform generator/analyzers, you must have a basic understanding of both transmission lines and termination so that you can maximize signal quality and minimize the effects of signal reflections.

The signals in the following figures show identical digital waveforms generated by an <u>NI PXI-6552</u>. The first figure shows a properly terminated waveform where the test system was designed with a careful understanding of both transmission lines and termination. The second figure shows a waveform from an unterminated system where transmission line effects and termination were not considered.



Consider the following key areas when designing your test system:

- Z_s —The impedance at the source of the transmission line
- Z_0 —The <u>characteristic AC impedance</u> of the transmission line
- Z_t —The impedance at the destination of the transmission line

By carefully controlling these three elements of your system, you can achieve the best possible results. Leaving these three elements

uncontrolled can result in the type of signal distortion shown in the improperly terminated signal and produce the following results:

- Signals that exceed specified high-level and low-level thresholds (overshoot and undershoot)
- Signals that have false edges (ringing)
- Signals that have reduced operating margins (degraded eye diagram caused by <u>inter-symbol interference</u>)
- Potential physical damage or overheating of driver/receiver components in extreme cases

Refer to <u>Terminating Your Module</u> for information about series and parallel resistor termination that is commonly applicable to your NI digital waveform generator/analyzer.

Transmission Lines

In lower frequency (slow edge rate) applications, you can assume that small wires interconnecting devices do not affect system performance and that every point in a wire has the same voltage as every other point for any instance in time. At lower frequencies, this "lumped" circuit model is valid. However, as frequency content increases, even the small geometries of typical wire dimensions become a significant portion of the signal wavelengths, in which case the small inductance and capacitance become electrically significant impedances.

What constitutes a significant proportion of the wavelength? While this value changes for different applications, for digital circuits, a good general rule is that if the propagation delay in a wire or interconnect is greater than one-sixth of the rise time of the digital signal, then the "lumped" circuit analysis assumption is no longer valid and you should analyze the interconnect as a *transmission line*.

For calculation purposes, you should understand the concept of *electrical length* (*I*). Electrical length is defined as the distance that a signal can travel in an electrical medium during the time that it takes for one <u>rise or</u> <u>fall time</u>, whichever is longer. Using the concept of electrical length, the general rule of the previous paragraph can be rephrased as follows: If the physical length of a wire or electrical interconnect is greater than one-sixth of the electrical length of a signal propagating on that wire, the system must be analyzed as a transmission line.

Velocity is defined as the rate at which an electrical wave propagates in the transmission medium. Using this value you can calculate electrical length in one of the following ways:

- *l(in)* = *Velocity(in/ns) t_{rise}*
- $l(in) = t_{rise}(ns)/t_{pd}(ns/in)$

where t_{rise} is the rise/fall time of the digital edge, and t_{pd} is the propagation delay of the edge in the transmission line.

For example, the NI SHC68-C68-D2 shielded cable has a t_{pd} of 165 ps/inch. On an <u>NI 655x</u> device, t_{rise} for a high-speed digital signal can be as low as 1.5 ns. Therefore, the electrical length is 9 in. ($t_{rise}/t_{pd} = 1.5/.165 = 9$ in.), and any trace lengths longer than 1.5 in. (9 in./6)

should be treated as a transmission line. Since this cable is significantly longer than 1.5 in., NI considers this cable to be a transmission line and designed the cable to have a controlled 50 Ω characteristic impedance cable.

M

Note While the propagation delay number in the previous example is specific to the NI SHC68-C68-D2 cable, if you do not know the specific propagation delays for your interconnects, when using the NI digital waveform generator/analyzer, assume that you are working with transmission lines for any wire or interconnect longer than 1 to 2 inches.

The following figure gives a simple diagram of a basic single-ended transmission line. A voltage source (V_s) generates a digital edge with an impedance of Z_s looking "into" the transmission line. The transmission line itself has some low <u>characteristic AC impedance</u> (Z₀) to ground, typically 50 Ω for most test systems. The end of the transmission line is most commonly terminated through an impedance (Z_t) to ground at the destination.



Practically, termination at only one end of the transmission line is often adequate and is more commonly used. However, for high-precision applications, termination at both the source and the load end of the transmission line yields the best results.

Characteristic Impedance

The characteristic impedance of a <u>transmission line</u> largely influences the transient response of a signal passing through it. The physical properties of the transmission line materials determine its characteristic impedance. For example, the dielectric of the insulators and the cross-sectional geometry of a cable determine its capacitance. Likewise, the inductance of the cable is a function of the length and the properties of the dielectric. The characteristic impedance is a function of both this inductance and capacitance. Manufacturers of cables provide the specification for the characteristic impedance of that cable, along with how it behaves over environmental extremes.

It is critical to match the characteristic impedance to the source impedance. If the impedances do not match, the resulting signal at the load is greatly distorted in both time and amplitude. Any time you disrupt the geometry described above, it results in impedance mismatches and signal <u>reflections</u>. For example, at the interfaces or boundaries between the cables and the devices, you should use connectors designed to maintain this characteristic impedance (coaxial connectors). Screw terminals, tees in the transmission line, or wire stubs are not recommended.

Caution Failure to use the connectors designed for these cables may result in impedance mismatches.

Signal Reflections

A digital rising or falling edge is a step function that can be modeled as a high-frequency wavefront. As the wavefront travels along the transmission line, it acts as a purely AC signal, encountering the characteristic impedance (Z_0) of the transmission line. When the wavefront reaches the end of the path, if Z_0 and the termination (Z_t) do not match, portions of the wave are reflected. As the wave reflects back along the transmission line, it eventually reaches the original source impedance (Z_s). If the transmission line characteristic impedance (Z_0) and Z_s do not match, then portions of the waves can cause significant signal degradation.

Reflection caused by an impedance mismatch at the end of a transmission line is quantified by the reflection coefficient. Reflection coefficient Γ is given by the following formula:

$$\Gamma = V_r / V_i = (Z_t - Z_0) / (Z_t + Z_0);$$

where V_r is the reflected voltage, V_i is the incident voltage, Z_t is the terminating impedance, and Z_0 is the characteristic impedance of the transmission line.

For example, by applying this formula, you can calculate that when a 3.3 V wave, traveling down a 50 Ω characteristic medium hits a 1 k Ω load impedance, the reflection coefficient Γ_t is equal to

(1 k Ω - 50 $\Omega)/(1$ k Ω + 50 $\Omega),$ or .90, and V $_r$ equals 0.9 x 3.3 V = 2.97 V.

Thus, the reflected wave V_r is almost the same magnitude as the incident wave. At the load, this condition only has the effect of giving an erroneous voltage—assuming that the circuit was originally calibrated with a 50 Ω load. While nearly the entire signal is reflected back, this reflection is eliminated at the source because the source and the transmission line are matched.

However, should the transmission line/cable be mismatched from the source and the load, the mismatch causes a scenario of multiple reflections resulting in aberrations at the load similar to what is shown in the improperly terminated signal in <u>this figure</u>.



Note NI *strongly* recommends that you take great care to ensure

that the source impedance of the system is matched as closely as possible to the characteristic impedance of the transmission line. For generation operations, the source impedance is inside the NI device and is handled by the hardware architecture. For acquisition operations, however, you control the source impedance of the system. You should create a source impedance as close to the characteristic impedance of your device as possible.

Types of Termination

There are several forms of line termination, including parallel, series, and differential.

Parallel termination matches the <u>characteristic impedance</u> of the medium at the end of the line, squelching the wavefront at the destination ($Z_t = Z_0$).

Differential termination is a variation of parallel termination used for differential transmission lines. Many electrical standards, such as emitter-coupled logic (ECL) and <u>LVDS</u>, require that traces are routed differentially. As such, parallel termination is used between the two modes of the differential trace.

Series termination places series impedance equal to the characteristic impedance at the source of the transmission line. This termination prevents the source from re-reflecting any reflections from an unterminated transmission line. It also prevents reflections from the transmission line to the source at the entry ($Z_S = Z_0$).

Practically, termination at only one end of the transmission line is often adequate and is more commonly used.

Crosstalk

Crosstalk is an expected natural phenomenon caused by the close proximity of signals and capacitance. Crosstalk results in unwanted coupling of the signals. Crosstalk is the ratio, in dB, of the level of the interference on the affected channel to the actual level of the interfering signal. Crosstalk consists of any unwanted signal on one channel that is caused by a signal on another channel.

Understanding I/O Current

The following subtopics explain important concepts that you should understand in relation to I/O current:

- Sinking and Sourcing
- AC and DC Current

Sinking and Sourcing Current

Sinking current refers to the ability of a circuit to dissipate current. Sourcing refers to the ability of a signal source to supply current.
AC and DC Current

AC current is current <u>sourced/sunk</u> during the transition between low and high level states driving a capacitive load (such as a cable) or an inductive load (such as a device interconnect). DC current is the current sourced/sunk when the generation terminals are at a static voltage driving a resistive load. The illustration below of a digital waveform shows when AC or DC current is sourced/sunk.



Use the following formulas to calculate AC current. $I = C \cdot \frac{dv}{dt}$

where

I is current in amps,

C is load capacitance in farads, and

dV/dt is the rate of change of the voltage level in volts/s,

or

 $I = \frac{1}{L} \int v_{(t)} dt$

where

I is the current in amps,

L is the inductive loading in henrys, and

 $V_{(t)}$ is the voltage in volts as a function of time.

Use the following formula to calculate DC current. $I = \frac{V}{R}$

where

I is current in amps,

V is voltage level in volts, and

R is resistance in ohms.

Digital Terminology

Expand this book for more information about some key terms used in this help file.

Timing and Triggering

Expand this book for more information about concepts and terminology related to timing and triggering.

Clocks

You can configure the following device clocks for your digital waveform generator/analyzer:

- Sample Clock
- <u>Reference Clock</u>
- <u>STROBE</u>

Sample Clock

The Sample clock is the primary timebase for the digital waveform generator/analyzer. This clock controls the rate at which samples are acquired or generated. Each period of the Sample clock is capable of initiating the acquisition or generation of one sample per channel.

Using <u>NI-HSDIO</u>, you can program the Sample clock to come from either the On Board Clock source signal or an external frequency generator. Because of the pipelined architecture of the NI digital waveform generator/analyzer, the clock source for the Sample clock must be continuous, free-running, and of a constant frequency for the duration of each generation and/or acquisition operation.

- <u>NI 654x Clock Sources Summary</u>
- <u>NI 655x Clock Sources Summary</u>
- <u>NI 656x Clock Sources Summary</u>

Reference Clock

The onboard frequency generator on the NI digital waveform generator/analyzer uses a phase-locked loop (PLL) circuit to lock the high-frequency internal timebase of the device to a known reference frequency. The most common clock to which the NI device is locked is the 10 MHz reference clock signal on the PXI backplane (PXI_CLK10). This clock signal is shared among all modules in the PXI system, so you can lock all the modules in your system to this stable clock.



Note PXI_CLK10 is only available on NI digital waveform generator/analyzers for the PXI bus. The Onboard Reference clock is a 10 MHZ clock signal that is available for use with NI digital waveform generator/analyzers for the PCI bus.

You can use the Reference clock only when On Board Clock is selected as the <u>Sample clock</u> source for a dynamic generation or acquisition session.

- <u>NI 654x Clock Sources Summary</u>
- <u>NI 655x Clock Sources Summary</u>
- <u>NI 656x Clock Sources Summary</u>

STROBE

The STROBE channel on the <u>DDC connector</u> is a dedicated channel for the STROBE signal. STROBE can be used only for acquisition sessions.

Designed for use in source-synchronous data transfer applications, the data channels are sampled precisely on a user-selected edge (programmable as rising or falling edge) of the STROBE signal, when configured as the <u>Sample clock</u>. The advantage of using STROBE as the Sample clock source signal for the acquisition operation is that the acquisition session Sample clock and the data channels now travel together through the same cable and system delays, maintaining time correlation between them.

Since NI digital waveform generator/analyzers only have one onboard clock, normally generation and acquisition sessions on the same NI device must use the same Sample clock rate. However, by choosing STROBE as the clock for acquisition operations, generation and acquisition operations can use two different timebases.

- <u>NI 654x Clock Sources Summary</u>
- <u>NI 655x Clock Sources Summary</u>
- <u>NI 656x Clock Sources Summary</u>

Triggers

Triggers are signals that cause the NI device to perform some action such as the starting, stopping, or pausing of an acquisition or generation operation. Triggers can be internal (software-generated) or external. External digital triggers can be several different <u>types</u>. External triggers can be re-exported and, along with <u>events</u>, can allow you to synchronize the hardware operation with external circuitry or other NI devices.

Refer to <u>Triggers Summary</u> and <u>Events Summary</u> for descriptions of the triggers and events you can use with your device.

Types of Triggers

You can configure the triggers supported by your NI digital waveform generator/analyzer as one of the following trigger types:

- Edge
- Level
- Pattern Match
- <u>Software</u>

Note Individual triggers may not support all the trigger types listed here. Refer to Triggers Summary for more information.

Edge Trigger

A digital signal has two discrete levels: a high level and a low level. When the signal transitions from high to low or from low to high, a *digital edge* is created. There are two types of edges: rising, which occurs when the signal transitions from low level to high level, and falling, which occurs with a transition from high level to low level. Triggers configured to act on a rising or falling edge of a digital signal are called *edge triggers*.

In the following figure, an edge trigger could be configured to occur either at the place labeled *Falling Edge of Signal* or at *Rising Edge of Signal*.



Level Trigger

You can configure certain triggers to act when a signal goes below the defined low level or above the defined high level. Triggers configured to act in this way are known as *level triggers*. Not all triggers can be configured to be level triggers. Refer to <u>Triggers Summary</u> for information about which triggers you can configure for level triggering.

Pattern-Match Trigger

A pattern-match trigger behaves like a combination lock. When the correct combination is read, the lock opens. Likewise, in the case of triggers, when the desired acquisition pattern is read, the pattern-match trigger is asserted.

The level state of a digital signal can be represented by a binary pattern, where a 1 corresponds to the high level (H) and a 0 corresponds to a low level (L). For example, if the logic levels on channels 0-3 are HLLH respectively, then this pattern could be represented in binary by replacing the H's with 1's and the L's with 0's— 1001. A pattern-match trigger allows you to configure the device to monitor the input terminals for a specific pattern (for example, 10101110). When this pattern is acquired by the device, the device asserts the pattern-match trigger.

You can also specify when you want rising (R or r) and falling edges (F or f) on any edge (E or e) to occur in the pattern to be matched.



Note Pattern-match triggers are valid only for acquisition sessions.

Software Trigger

A software trigger is generated internally by a programmatic call, such as a LabVIEW VI or C function, and can occur at any time, based upon the conditions specified in the program.

Triggers Summary

The following table describes the triggers supported by NI digital waveform generator/analyzers. The *Used In* column indicates which types of operations can use a trigger type. The *Supported Types* column denotes which trigger types are valid for a given trigger.



Note Using <u>DDR mode</u> has certain implications for using some <u>acquisition triggers</u>.

Trigger Name	Used In	Supported Types	Description
Start	Acquisition, Generation	Digital Edge, Pattern Match ¹ , Software	 The Start trigger transitions a device into a state where the device can respond to Sample clocks. For an acquisition session, this trigger transitions the device from a nonsampling state into a sampling state—the device starts sampling and storing data. For a generation session, this trigger transitions the device from a nonseling and storing data. For a generation session, this trigger transitions the device starts sampling state into a sample generation state—the device from an Idle state to a sample generation state—the device starts generating samples.
Reference	Acquisition	<u>Digital</u> <u>Edge,</u> <u>Pattern</u> <u>Match¹, Software</u>	The Reference trigger transitions a device from the Wait for Reference Trigger sampling state into the Post Reference trigger sampling state. In the Post Reference Trigger sampling state, a counter begins counting Sample clock cycles. When the configured number of samples is acquired, the device transitions into a Done state. In other words, the arrival of this trigger establishes the reference point that separates

			pretrigger and posttrigger samples.		
Advance	<u>Acquisition</u>	<u>Digital</u> <u>Edge,</u> <u>Pattern</u> <u>match¹, Software</u>	The Advance trigger initiates the acquisition of the additional records in a multirecord acquisition.		
Pause	<u>Acquisition,</u> <u>Generation</u>	<u>Digital</u> <u>Level,</u> <u>Pattern</u> <u>Match¹</u>	The Pause trigger indicates to the device that it should pause the acquisition or generation. Therefore, the Pause trigger is only effective when received during an active acquisition or generation session. For generation operations, the <u>Data</u> <u>Active event</u> indicates when the operation is paused.		
Script	Generation	<u>Digital</u> <u>Edge,</u> <u>Digital</u> <u>Level,</u> <u>Software</u>	The Script trigger is a general- purpose trigger with a role that is entirely determined by the context of the dynamic generation script. A script allows you to create sophisticated dynamic generation operations. For example, the script could configure the device to generate waveform A, then wait for the Script trigger, then generate waveform B. You can create multiple Script triggers for use in your application.		
¹ Pattern match triggers are valid only for acquisition sessions.					

Events

An event is a signal generated by the NI device at a device state. Typically, events are configured to indicate when a specific hardware condition has been met.

Refer to <u>Triggers Summary</u> and <u>Events Summary</u> for descriptions of the triggers and events you can use with your device.

Events Summary

The following table describes the event types supported by NI digital waveform generator/analyzers. The *Used In* column indicates which types of operations can use an event type.

Event Name	Used In	Description
Data Active Event	<u>Generation</u>	The Data Active event indicates when the Pattern Generation Engine is generating data. If the Pattern Generation Engine is waiting for a trigger or is paused, the Data Active event is deasserted. When the Pattern Generation Engine is generating data, the Data Active event is asserted, synchronous with the data.
Marker Event	Generation	 The Marker event is a general-purpose event that is configured within a generation script. The Marker event can be asserted synchronous to any even numbered sample within a waveform within a script. For example, the Marker event can be asserted when sample 432 of waveform A is generated. You can create multiple Marker events for use in your application. Note When using DDR mode, marker positions will have a quantization twice that of SDR mode. Refer to your device specifications for more information about marker quantization.
Ready for Start Event	Acquisition, Generation	For both acquisition and generation, the Ready For Start event indicates that the NI digital waveform/generator analyzer is configured and ready to receive a Start trigger.
Ready for Advance Event	Acquisition	An event that indicates when the device enters its Wait for Advance Trigger state, which indicates that the acquisition of the previous record is complete.
End of	<u>Acquisition</u>	An event that indicates when the device enters its

Record Event		Record Complete state, which indicates that the current record has been acquired.
Sample	<u>Hardware</u>	An event that indicates when the device detects a
Error	Comparison	sample where the actual response and the
Event		expected response do not match.

Generation

Expand this book for more information about concepts and terminology related to generation.

Drive Type

Drive type describes the behavior of the generation channels. Singleended high-speed digital waveform generator/analyzers support two drive types: <u>active drive</u> and <u>open collector</u>. The generation drive type can be configured individually for each channel and is available for dynamic and static operations.

For active drive channels, data is fed to the output driver. For opencollector channels, the data provided to the output driver is always zero, and the data is fed to the tristate control terminal of the output driver.

You can configure all NI digital waveform generator/analyzers for active drive generation and some for open collector generation. When configured for active drive generation, a channel generates the Generation Voltage High Level for logic 1. When configured for opencollector generation, a channel goes to the high-impedance state for logic 1. In both cases, a channel generates the Generation Voltage Low Level for logic 0.

Active Drive

When configured for active drive generation, a channel generates the Generation Voltage High Level for logic 1, and the channel generates the Generation Voltage Low Level for logic 0.

You can configure all NI digital waveform generator/analyzers for active drive generation and some for <u>open collector</u> generation.

Open Collector

When configured for open-collector generation, a channel goes to the high-impedance state for logic 1. For open collector operations, external pull-up resistors are typically used to force a voltage for the logic high state.

Note Differential NI digital waveform generator/analyzers, such as the <u>NI 656x</u> devices, do not support open collector generation. <u>NI 654x</u> devices support open collector generation only for <u>static</u> <u>generation</u>.

Drive type is a channel-based attribute; channels configured for opencollector output never drive the Generation Voltage High Level in the line. Logic ones found in <u>initial</u> and <u>Idle states</u> force the line to high-impedance for static and dynamic generations.

The main application for open collector generation channels is bidirectional, multidrop busses. Open-collector outputs prevent different gates from attempting to double-drive the line. By using a pull-up resistor, the bus is high if all connected devices are high, and the bus is low if any device drives a zero. This setup is generally called wired-AND for positive-true logic and wired-OR for negative-true logic. Some examples of busses implemented with open-collector outputs include I²C and SMBus.

Interfacing with a wired logic bus requires you to consider the voltages that are generated at the bus. These voltages depend on four factors: value of the pull-up resistor, value to which the line is "pulled up", configured voltage for low state, and configured input impedance. The following diagram shows how these factors interact.



Typical applications will have a Generation Voltage Low Level of 0.0 V,

and R_{in} as high impedance; R_{pull} must be carefully chosen to get the desired voltage values in the bus. Refer to your device <u>specifications</u> for the R_{in} for your device. Typical values for a pull-up resistor in I²C range between 2.4 k Ω and 3.9 k Ω .

<u>NI 655x devices</u> have <u>per cycle</u>, <u>per pin tristate</u> functionality, which allows the generation of zeros, ones, and tristate in the same channel. Open collector is limited to zero and tristate; however, these two features can operate simultaneously. Using per cycle tristate in a channel configured for open-collector generation causes the line to be tristated by both Z's and 1's found in the waveform.

Static Generation

Static generation places a single pattern on the configured channels. Static generation, like <u>static acquisition</u>, is controlled by software and does not use hardware timing.

Because a function call is required for each data point generated, static generation is generally used only for single-point or low-speed applications. Static generation can be helpful in system and cable debugging, DC-level semiconductor testing, and many other applications.

Static generation is also called immediate, unstrobed, or nonlatched generation.

Dynamic Generation

Dynamic generation is a clocked operation where binary data is sent from the NI device to the DUT across multiple digital channels. The data can be generated based on complex <u>scripts</u>, and it can react to triggers, generate markers, and be shifted in time with respect to the generating clock.

- NI 654x Dynamic Generation
- NI 655x Dynamic Generation
- NI 656x Dynamic Generation

Waveforms

All user generation data patterns are stored to the device <u>onboard</u> <u>memory</u> as *waveforms*. These waveforms are arrays of every sample that the device generates in the order they are generated.

Generated waveforms are assigned a name when they are stored in onboard memory. This name allows you to store or delete multiple waveforms from the device memory and refer to them easily for simple generation or for complex scripts. When the generation session is closed, the waveforms are removed from onboard memory.

Scripts

You can link and loop multiple waveforms together in a generation operation using a *script*. A script is a series of instructions that indicates how waveforms saved in the <u>onboard memory</u> should be sent to the DUT. The script can specify the order in which the waveforms are generated, the number of times they are generated, and the triggers and markers associated with the generation.

You can create a script to manage <u>dynamic generation</u> based on multiple waveforms and triggers. For example, you could download waveforms A, B, C, and D into device memory. You could then write a script that would wait for a trigger to initiate dynamic generation and, upon receiving this trigger, generate waveform A three times, then generate marker 0, and finally generate waveforms B, C, and D twice (BCDBCD).

A simple script example is shown below:

script myFirstScript generate countUp generate allOnes generate countDown end script

When executed, this script generates three waveforms (countUp, allOnes, and countDown) consecutively.

- Generating Multiple Waveforms/Linking & Looping
- <u>Common Scripting Use Cases</u>
- <u>Scripting Instructions</u>

Initial and Idle States

The *Initial state* configures the state of the data generation channels after a session has been configured but before the device starts generating the waveform. The Initial state is often useful while the device is waiting for a Start trigger.

The *Idle state* configures the state of the data generation channels after the waveform generation has begun and the generation has paused or stopped.

The Initial state and Idle state are per channel selectable. You can select the following values on all NI digital waveform generator/analyzers for data generation channels:

- **1**—Drive the channel to a high level.
- **0**—Drive the channel to a low level.
- X—Hold last value/Leave the channel at its current state.
- Z—(NI 654*x*/655*x* only) Put the channel in a high-impedance state.
- Note If you use an X Idle state in <u>DDR mode</u>, the last value held is the last valid state of the channel.

- Configuring Initial and Idle States
- Generation Considerations for DDR Mode

Per Cycle Tristate

NI digital waveform generator/analyzers allow you to select between driving a 0 or a 1 during every active period of the Sample clock. Some NI devices have the additional capability of selecting between driving a 0 or 1 and tristating the channel during every active clock cycle. This capability is referred to as *per cycle tristate*.

To enable this functionality, the NI digital waveform generator/analyzer allows you to create waveforms composed of 0, 1, and Z values. For each sample in the waveform, you can select which channel to tristate by inserting Z values in the waveform at that location.



Per cycle tristate is useful for communicating or testing bidirectional digital channels. For example, communicating with a memory device may require the generator to drive address and data channels during a write, but tristate the data channels during a read.

 $\overline{\mathbb{N}}$

Notes At high-speeds, care must be taken when switching a channel from driving to receiving data to ensure the signal reflections resulting from tristating the channel do not affect the signal transmission.

Per cycle tristate is currently only supported on <u>NI 655x</u> devices.

Related Topic: Per Cycle Tristate (NI 655x)

Generating Waveforms Using Streaming

Streaming is a method of generating waveforms that are too large to fit in the onboard memory of the device. Streaming can be used in dynamic generation sessions.

To stream waveform data, you allocate and identify all or a portion of the device onboard memory to act as an onboard waveform for streaming. Before initiating generation, you fill that onboard memory with the first part of your waveform. As the waveform is generated, space in the onboard memory becomes free and you fill that space with new waveform data. You repeat the process of filling the freed onboard memory in blocks of new waveform data until the waveform is complete.

The following instructions are a guide for configuring your application for streaming.

As an example, we have a 1.6 GB waveform we want to generate and a device with 256 MB of onboard memory. This 1.6 GB waveform may be in the host memory, on disk, or data that your application generates on-the-fly during generation.



1. Specify the amount of onboard memory used for streaming.

Use the <u>niHSDIO Allocate Named Waveform</u> VI or the <u>niHSDIO_AllocateNamedWaveform</u> function to specify the amount of onboard memory to reserve for streaming. This memory serves as a buffer for the streaming process. The niHSDIO Allocate Waveform VI/function returns a waveform handle.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	✓ 1.6 GB Waveform
160 MB	160 MB Allocated for Streaming
man manan ma	Remaining Onboard Memory Used for Additional Waveforms and Script Instructions

2. Identify the streaming waveform.

Set the <u>Streaming Waveform Name</u> property or the <u>NIHSDIO ATTR STREAMING WAVEFORM NAME</u> attribute to the waveform handle returned in Step 1. Setting this property ensures that no streaming data is overwritten before it is generated. NI-HSDIO monitors your progress to ensure that you write fresh data fast enough to keep up with the generation. If your application fails to keep up, or attempts to write fresh data on top of data that has not been generated, NI-HSDIO returns an error.

	Set the to Identif	Streamin ly the Wa	g Wavefor veform fo	m Har r Strea	ndle Proj aming	perty	
_	¥						
		100	MD				
		100	MD				
$\sim$	~~~~	$\sim$	~~~~~	$\sim$	~~~~		
$\sim$	~~~~~	$\sim\sim$	$\sim$	$\sim \sim$	$\sim$		

3. Fill the streaming waveform with initial data.

Call the <u>niHSDIO Write Named Waveform</u> VI or the one of the <u>Write Named Waveform functions</u> to write the first part of the waveform data to the streaming waveform in onboard memory.

Tip When transferring large blocks of waveform data, break up the data into smaller blocks and call the <u>niHSDIO</u> Write Named Waveform VI or the one of the Write Named Waveform functions multiple times—the data is appended sequentially. A computer can allocate smaller blocks of a large waveform faster than allocating a single large contiguous block in memory. Depending on the amount of RAM on the computer, transferring ten 16 MB blocks can be faster than transferring one 160 MB block.



4. Begin generating the waveform.

Call the <u>niHSDIO Init Generation Session</u> VI or the <u>niHSDIO_InitGenerationSession</u> function to begin the waveform generation. As the waveform generates, space in the streaming waveform becomes free.

······································	1.6 GB Waveform
As Waveform is Generated, Onboard Memory Becomes Free	
160 MB	

5. As the waveform generates, monitor available memory.

Use the <u>Space Available in Streaming Waveform</u> property or the <u>NIHSDIO ATTR SPACE AVAILABLE IN STREAMING WAVEFOF</u> attribute to determine how much of the onboard waveform is free for writing new data. As the waveform generates, space becomes available to write more waveform data. Once a certain amount of the onboard memory becomes available, say 10 percent of the allocated onboard memory, you can write more of the waveform to the streaming waveform in onboard memory. In general, writing in larger blocks, such as 16 MB, results in more efficient streaming and faster streaming rates.

m	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	1.6 GB Waveform
	Query the Space Available in Streaming Waveform Property to Determine Freed Onboard Memory	1
	160 MB	

6. Write a block of waveform data.

Call the <u>niHSDIO Write Named Waveform</u> VI or the one of the <u>Write Named Waveform functions</u> to write a new block of waveform data to the streaming waveform in onboard memory.

16 MB		
m	1.6 (	GB Waveform
Write	Waveform Data in Blocks	
16 MB	160 MB	

7. Repeat the process of monitoring the available memory and writing waveform data in blocks as free space becomes available.



#### **Improving Streaming Performance**

To improve your maximum sustainable transfer rate for streaming, consider the following:

- Adjust the <u>Data Transfer Block Size</u> property or the <u>NIHSDIO_ATTR_DATA_TRANSFER_BLOCK_SIZE</u> attribute to be closer to the waveform size. This change makes the data transfer is more efficient and is accomplished in a single transfer instead of four transfers.
- When streaming from hard drives, consider the hard drive speed for maximum sustainable rates. Laptop hard drives typically have a data transfer rate of 5 to 10 MB/s. Desktop hard drives often can meet or exceed 20 MB/s.

Transfer rates from hard drives can vary for a number of reasons. For example, where the data is physically stored on the hard drive and how much data is stored can affect transfer rates. Storing your waveform files on a fairly empty, defragmented hard drive may help increase performance.

- Consider using a RAID configuration to utilize striping to increase data transfer rates from disk.
- When using 18-slot PXI chassis, install the device used for streaming in the first segment, Slots 2 to 6, of the PXI chassis.
- Utilize <u>Direct DMA</u>.

### **Direct DMA**

Direct DMA can be used to transfer waveform data to and from device onboard memory at rates well beyond the typical 5 to 30 MB/sec range in a standard PC-based architecture. To achieve such high rates, Direct DMA establishes a direct connection between device onboard memory and a specialized waveform data source. Direct DMA is commonly used to <u>stream</u> waveform data from disk at data rates of higher than 100 MB/sec. Conduant's StreamStor[™] products are one example of Direct DMA-compatible data sources.

#### **Configuring Your Application for Direct DMA**

Direct DMA is available for both acquisition and generation. For acquisition operations, use the Direct DMA instance of the nniHSDIO Fetch Waveform VI; no additional attribute configurations are needed. For generation operations, complete the following steps to configure your application for Direct DMA.

- Enable the device for Direct DMA writes by setting the <u>Direct DMA</u> <u>Enable</u> property or the <u>NIHSDIO_ATTR_DIRECT_DMA_ENABLED</u> attribute to TRUE. Once enabled, NI-HSDIO monitors and reports any issues with the Direct DMA transfer.
- Identify the waveform data source and set the <u>Direct DMA</u> <u>Window Address</u> property or the <u>NIHSDIO_ATTR_DIRECT_DMA_WINDOW_ADDRESS</u> attribute to the address provided by your Direct DMA-compatible data source.
- 3. Set the <u>Direct DMA Window Size</u> property or the <u>NIHSDIO_ATTR_DIRECT_DMA_WINDOW_SIZE</u> attribute to the size of the memory window provided by your Direct DMA-compatible data source.
- 4. In LabVIEW, use the Direct DMA instance of the <u>niHSDIO Write</u> <u>Named Waveform</u> VI.

or

In C/CVI, use the instance of the Write Named Waveform function that corresponds to the data width of your device to write blocks of data to the device. For each block of data written to the device, you pass the Direct DMA window address to the **data** parameter instead of an array of samples residing in host memory. NI-HSDIO detects when the address is within the Direct DMA window and handles the transfer appropriately.
# Acquisition

Expand this book for more information about concepts and terminology related to acquisition.

# **Static Acquisition**

Static acquisition samples the configured channels once, reflecting the present state of the channels. Static acquisition, like <u>static generation</u>, is controlled by software and does not use hardware timing.

Because a function call is required for each data point acquired, static acquisition is generally used only for single-point or low-speed applications. Static acquisition can be helpful for system and cable debugging, DC-level semiconductor testing, and many other applications.

Static acquisition is also called immediate, unstrobed, or nonlatched acquisition.

# **Dynamic Acquisition**

Dynamic acquisition is a clocked event in which digital data is transferred from the DUT into <u>onboard memory</u>. The logic state of each acquisition is determined by the <u>configured data interpretation</u> method, and the acquisition <u>voltage levels</u> are defined by your dynamic acquisition operation.

At every clock edge, the Pattern Acquisition Engine stores the current state of each DIO channel configured for dynamic acquisition into onboard memory as a *sample*. Samples are stored in the order they are received to onboard memory as a <u>record</u>. You can configure the number of samples per record.

#### **Related Topics:**

- <u>NI 654x Dynamic Acquisition</u>
- NI 655x Dynamic Acquisition
- NI 656x Dynamic Acquisition

## Records

A record is a group of samples. Acquired data is stored into device <u>onboard memory</u> as a record. When <u>configuring an acquisition session</u>, you can determine how many samples are stored in a record.

You can also acquire multiple unique records in a series; this process is known as multirecord acquisition. The <u>Advance trigger</u> initiates the acquisition of the additional records in a multirecord acquisition. The <u>End</u> <u>of Record event</u> indicates when a record acquisition is complete.

### Hardware Comparison

Hardware comparison allows a device to verify in real-time at the full data rate of the device that a DUT generates the correct response data under different use cases and stimulus data. Traditionally, the comparison was done by acquiring data into the PC memory and then performing software analysis.



**Note** Real-time hardware comparison is supported with *only* the NI 655*x* devices. Other NI digital waveform generator/analyzers, such as the NI 654*x*/656*x* support acquiring the data into PC memory for analysis.

#### **Related Topics:**

- Digital Logic States
- <u>Hardware Comparison</u> (NI 655x)
- <u>Comparing Response Data with Expected Data</u>

# **Onboard Memory**

NI digital waveform generator/analyzers use multiple megabytes of onboard memory to acquire and generate data. Onboard memory allows much higher data rates than would be allowed by streaming data from system memory across the PCI bus. Acquisition Memory and Generation Memory are separate; for example, a 64 MS device has two 64 MS memory blocks—one for acquisition and one for generation.



**Note** If you are performing hardware comparison, the fault data is stored separately from the acquisition and generation data, so fault data does not consume any of your Acquisition or Generation Memory. Hardware comparison is supported only on <u>NI 655x</u> devices.

The NI 654x/655x/656x can be purchased with different memory options. Refer to <u>ni.com/catalog</u> for more information about these options.

Refer to <u>Generation Onboard Memory</u> and <u>Acquisition Onboard Memory</u> for more information about memory management for your application.

### **Generation Onboard Memory**

Waveforms and scripts are stored together in device memory. They are stored in contiguous blocks, appearing in memory in the order they were written to the device. You can delete individual waveforms from the device, freeing up the space they occupy for other waveforms to be written.

Deleting waveforms that are not at the end of the utilized space causes *memory fragmentation*. The following scenario demonstrates how memory fragmentation can occur. First, assume four waveforms are currently in memory as shown in the following figure (sizes, in MS, are shown for clarity).



In the previous figure, there is enough memory to write an additional 22 MS waveform to the device.

If Waveform C is deleted, that memory is freed, as shown in the following figure.



However, because waveforms are always stored contiguously in memory, the largest waveform that could be stored in memory is still 22 MS.

Writing Waveforms C last would have been advantageous because then deleting Waveform C would create a single block of free space, as shown in the following figure.

Waveform A	Waveform B	Waveform D	Waveform C	Free Space
(11 MS)	(8 M S)	(8 MS)	(15 MS)	(22 M S)
Waveform A	Waveform B	Waveform D	Free Space	
(11 MS)	(8 MS)	(8 M S)	(37 MS)	

In this situation, you can now write a 37 MS waveform to your device. Notice that when you create a script for your dynamic generation

# operation it consumes some space in memory, as shown in the following figure.

Waveform A	Waveform B	Waveform C	Script	Waveform D	Free Space
(11 MS)	(8MS)	(14MS)		(8 MS)	(22 M S)

## **Acquisition Onboard Memory**

When an acquisition operation is initiated, the device begins waiting for the Start trigger. Once the Start trigger is received, the device starts acquiring data and storing the samples into device memory. The first sample acquired marks the beginning of the acquired record, as shown in the following figure.

<b></b>

If no Start trigger has been configured, acquisition begins immediately after the operation is initiated.

After the device has recognized the Start trigger and has acquired the configured number of pretrigger samples, the device can now recognize a Reference trigger. While waiting for the Reference trigger, the device is still sampling data into the device memory. If the record overflows, the newest samples overwrite the oldest samples in the record. After the Reference trigger is received, the device acquires enough posttrigger samples and finishes the acquisition, as shown in the following figure.



If no Reference trigger has been configured, a single record of data is acquired.

#### **Fetching Acquired Data**

Data acquired between the Start trigger and the first pretrigger sample is also available to be fetched. The following figure shows the four common fetch positions.



In cases where no Reference trigger has been configured, *First Pretrigger Sample* and *Reference Trigger* are equivalent to *First Sample*.

#### **Multirecord Acquisitions**

In the case of multirecord acquisitions, the Advance trigger initiates the acquisition or fetch operation for the second and all subsequent records. Therefore, for multirecord acquisitions, the Start trigger shown in the previous figures would be replaced by an Advance trigger for all records after the initial acquisition.

### **Data and Clock Position**

Your NI digital waveform generator/analyzer allows you to configure the subperiod time at which each waveform sample is generated or acquired. This subperiod time selection is referred to as the <u>data position</u>. You can also configure the position of the exported Sample clock (<u>clock position</u>).

Configuring data and clock positions allows you to use your NI digital waveform generator analyzer for many common applications including, among others, measuring set-up and hold times, measuring propagation delays, and maximizing the timing margins among high-speed data transfers.

# **Data Position Settings**

You have three available data position settings for acquisition and generation channels:

- **Sample clock rising edge**—Data is generated/acquired on the rising edge of the clock driving the operation.
- **Sample clock falling edge**—Data is generated/acquired on the falling edge of the clock driving the operation.
- Delay from Sample clock rising edge—Data is generated/acquired at a specified time (specified in the <u>niHSDIO</u> <u>Configure Data Position Delay</u> VI or the <u>niHSDIO</u> <u>ConfigureDataPositionDelay</u> function) after the rising edge of the clock driving the operation. The <u>data position delay</u> <u>resolution</u> depends on your clock frequency.



**Note** NI 656*x* devices have special considerations for <u>legal</u> <u>delayed data</u> settings for Sample clock frequencies between 25 and 50 MHz.

Refer to the *Acquisition* and *Generation* books for your <u>device</u> for timing diagrams illustrating changing data position.

Refer to <u>Configuring Data Position</u> for more information about using NI-HSDIO to configure data position.

### **Clock Position Settings**

You have three available clock position settings for the position of the exported Sample clock:

- **Inverted**—The exported Sample clock is an inverted copy of the <u>Sample clock</u>.
- Noninverted—The exported Sample clock is an exact copy of the <u>Sample clock</u>.
- **Delayed**—The exported Sample clock is a delayed version of the <u>Sample clock</u>, delayed by 0 to 1 of the Sample clock periods. The <u>data position delay resolution</u> depends on your clock frequency.



**Notes** Because the data position is relative to the Sample clock, changes to the exported Sample clock position do not affect the timing of the data channel operations.

<u>NI 656x devices</u> have <u>special considerations</u> for exporting the Sample clock at frequencies between 25 and 50 MHz.

#### **Related Topics:**

- Dynamic Generation Timing Diagrams (NI 654x)
- <u>Dynamic Generation Timing Diagrams</u> (NI 655*x*)
- Dynamic Generation Timing Diagrams (NI 656x)
- <u>Advanced Attributes</u>

### **Data Position Delay Resolution**

NI digital waveform generator/analyzers have three internal independent delay mechanisms, one for <u>dynamic generation</u>, one for <u>dynamic</u> <u>acquisition</u>, and one for the <u>exported Sample clock</u>. The delay mechanisms are capable of delaying the data and clock positions by up to one full Sample clock period, in steps of 1/256 of the Sample clock period, for Sample clock frequencies of 25 MHz and above. Refer to the <u>specifications</u> document for your device for valid frequencies and ranges for delays.

Note NI 656x devices have special considerations for legal delayed data settings for Sample clock frequencies between 25 and 50 MHz.

The following table lists the resolution of the delay mechanisms for the frequencies that the NI digital waveform generator/analyzer internal Sample clock can produce. For externally supplied frequencies above 25 MHz that are not listed in this table, the delay resolution is 1/256 of the Sample clock period.

Operating Frequency*	Resolution/Step Size
200 MHz	20 ps†
100 MHz	39 ps†
66.7 MHz	59 ps†
50 MHz	78 ps
40 MHz	98 ps
33.3 MHz	117 ps
28.6 MHz	137 ps
25 MHz	156 ps

*Not all operating frequencies will be applicable to your <u>device</u>.

[†]For NI 656_X devices, refer to the <u>device specifications</u> document for more information about the supported step sizes.

## Data Width

Data width is the size, in bytes, of a raw sample from the operation. *Raw sample* refers to the native format and organization of the device, which can be an 8-, 16-, or 32-byte integer.

If you do not require all the channels in your device during a <u>dynamic</u> acquisition and would like to achieve greater memory depth, you can specify a data width other than the total number of available bits for your device for the session. For example, if you are acquiring data on eight of the 16 channels of an NI 656*x*, if you specify a data width of one byte, you can store twice as many samples than if you had specified the data width as being the entire two bytes. Data width is read-only during generation sessions.

### **Data Rate Multiplier**

NI digital waveform generator/analyzers can be configured to acquire and/or generate data once per Sample clock period (single data rate, or SDR mode) or twice per Sample clock period (double data rate, or DDR mode).



**Note** DDR is not supported by all <u>devices</u>.

You can <u>configure the data rate multiplier</u> for your acquisition and generation sessions by configuring the data rate multiplier property/attribute in NI-HSDIO.

# Single Data Rate (SDR)

When the <u>data rate multiplier</u> is configured for SDR operation, the NI digital waveform generator/analyzer generates or acquires data once per Sample clock period. The relationship of the data to the Sample clock is determined by the <u>data position</u>.

The following figure shows an example of SDR generation configured for rising edge data position.



# **Double Data Rate (DDR)**

When the data rate multiplier is configured for DDR operation, the NI digital waveform generator/analyzer generates or acquires data twice per Sample clock period. The digital waveform generator/analyzer trades channel count for data rate by generating or acquiring on half the number of channels but at twice the rate. The phase relationship of the data to the Sample clock is determined by the data position. While you can configure the data rate multiplier for acquisition and generation separately, you cannot have both SDR channels and DDR channels in the same direction.

The following figure shows an example of DDR generation configured for rising edge data position.





**Notes** DDR is currently only available on NI 656x devices.

Configuring the data rate multiplier for DDR operation impacts memory usage, data width, and other aspects of your application. Refer to the device documentation to determine if DDR mode meets your application needs.

### **Data Position with DDR**

Acquisition and generation sessions can be configured to acquire or generate the first data sample on the rising or falling edge of the Sample clock or on a delayed version of the rising edge of the Sample clock when the data rate multiplier is configured for DDR operation. The second sample is acquired or generated on each subsequent Sample clock edge. These data positions are shown in the following figure. Notice that the delay is still a fraction of the entire Sample clock period.



 $t_P = \frac{1}{f} = Period of Sample Clock$ 

 $t_{CO} = E \times ported Sample Clock Offset$ 

### **Generation Considerations for DDR**

DDR generation has some additional considerations for your application. The following diagram shows how digital data is stored in <u>Generation</u> <u>Onboard Memory</u> and how that impacts the trigger, event, and waveform quantum and the generated data.



#### Data Width

Data width is a function of your data rate multiplier. Since data width refers to how large your sample is in bytes, using DDR mode effectively halves your allowable data width. For example, on a device with 16 channels, you can generate or acquire data on all 16 channels. For the same device in DDR mode, you can generate on only eight channels and acquire on the other eight.

#### **Memory Usage**

<u>Memory</u> usage is effectively doubled per channel since the data width and channel count are halved.

#### **Marker Positions**

<u>Marker</u> positions have a quantization twice that of SDR mode. Refer to your <u>device specifications</u> for more information about quantization.

#### **Waveform Sizes**

The size of the <u>waveforms</u> you save to the <u>onboard memory</u> have a quantization twice that of SDR mode. Refer to your <u>device specifications</u> for more information about quantization.

#### Initial/Idle States

If a channel's <u>Idle state</u> is configured for "hold last value" (**X**), the last value held is the last DDR data sample.

# Acquisition Considerations for DDR

DDR acquisition has some additional considerations for your application.

#### **Edge Triggers**

In DDR acquisitions, while data channels are sampled on both clock edges, triggers are only stored once per Sample clock period. For example, refer to the following figure. Consider if PFI Trigger A was a digital edge Start trigger, and the device were configured for rising edge data position. Whether PFI Trigger A arrives before the rising edge of the clock or before the falling clock edge, the trigger has the same effect the first sample acquired is that of the rising clock edge.



#### **Pattern Match Trigger**

In DDR acquisitions, while data channels are sampled on both clock edges, triggers are only stored once per Sample clock period. In the following figure, consider if the device were configured for a pattern match Start trigger and rising edge data position. Whether the data in sample A *or* B matches the pattern configured for the pattern match trigger, sample A is the first acquired sample.



#### **Special Considerations for Using the Pause Trigger**

If a Pause trigger is asserted on either edge of the Sample clock, the acquisition is paused for the samples that occur on both edges of the clock.

#### Data Width

Data width is a function of your data rate multiplier. Since data width refers to how large your sample is in bytes, using DDR mode effectively halves your allowable data width. For example, on a device with 16 channels, you can generate or acquire data on all 16 channels. For the same device with its data rate multiplier configured for DDR, you can generate on only eight channels and acquire on the other eight.

#### **Memory Usage**

<u>Memory</u> usage is effectively doubled per channel since the data width and channel count are halved.

### **Channel-to-Channel Skew**

For <u>dynamic generation</u>, channel-to-channel skew is defined as the time difference between corresponding edges on the data channels. For example, if two data channels are each programmed to transition from low to high level on a particular sample, the time difference between the rising edges on the two channels would be the channel-to-channel skew between the two channels.

For <u>dynamic acquisition</u>, channel-to-channel skew is defined as the difference between the sampling times for each data channel. When each sample is acquired, the point in time at which each data channel is sampled with respect to every other data channel is not identical, but the difference is within some small window of time. This time window is referred to as the channel-to-channel skew.

The following figure shows the channel-to-channel skew of a group of signals.



Specified channel-to-channel skew generally refers to the skew across all data channels on a device.

### **Devices**

Expand this book for more information your NI digital waveform generator/analyzer hardware and functionality.

### NI 654*x*

The NI 654*x* is a 32-channel digital I/O device that you can use as a PC/peripheral device interface, pattern generator, pattern analyzer, or stimulus-response tester. The NI 6541 has a maximum Sample clock frequency of 50 MHz, and the NI 6542 has a maximum Sample clock frequency of 100 MHz.

The NI 654*x* also provides the following features:

- Sophisticated timing engine to maintain and measure the timing parameters of the DUT
- Selectable <u>voltage levels</u> for interfacing to devices from different logic families
- Data channels with per channel direction control
- Deep onboard memory with triggering and pattern sequencing capabilities
- Ability to use <u>NI-TClk</u> to synchronize multiple devices

You can use the internal On Board Clock or import an external clock through the front panel. You can also shift the generated data, acquired data, and exported Sample clock relative to the onboard clock for clock frequencies above 25 MHz, which is critical when accounting for propagation delays and setup-and-hold times in the DUT.

Expand this book for more information about NI 654*x* hardware-related topics.

### **Hardware Architecture**

Expand this book for more information about the NI 654x hardware architecture.
## **Block Diagram**

The following figure is a block diagram illustrating the main functional units and data flow of the NI 654x. The text that follows the figure describes the basic elements of the diagram and provides links to sections with more detailed information about some of the blocks.





The <u>Clocking</u> module selects and distributes the clocks for the dynamic generation and dynamic acquisition operations.

For dynamic generation operations, the user-supplied data is loaded from the host computer memory into the onboard Generation Memory. The Pattern Generation Engine retrieves data from Generation Memory and executes the script functionality while interacting with the associated Trigger and Event control module. The Pattern Generation Engine then sends the data to the Pattern Generation Timing and Control module, where the data is given the selected data position and data delay and is then sent to the <u>channel electronics</u> drivers. The channel electronics drivers generate the data at the voltage levels of the selected <u>logic family</u>.

For dynamic acquisition operations, signals arrive at the channel electronics, where the signal levels are compared to the voltage thresholds for the selected logic family. The Timing and Control module samples the data using the selected clock, data position, and data delay values and passes the data to the Pattern Acquisition Engine. The Pattern Acquisition Engine and the Trigger and Event Control module recognize triggers and determine when the data should be stored into Acquisition Memory. The acquired data can then be fetched by the host computer.

## **Channel Electronics**

The channel electronics of NI 654*x* devices consist of selectable voltage buffers, the appropriate termination resistors, and I/O protection diodes. Each I/O channel is capable of simultaneously driving and receiving data.

The following figure provides a basic block diagram for the channel electronics. Refer to  $\underline{NI \ 654x \ Block \ Diagram}$  for a picture of how the channel electronics circuitry fits into the overall block diagram.



#### **Dynamic Generation**

For dynamic generation operations, the data appears at the input of the selectable voltage buffer after the Pattern Generation Timing and Control module gives the data the selected data position and data delay. The selectable voltage buffer converts the data signal to the user-selected logic family before sending the data signal to the DDC connector on the NI 654*x* front panel.

The selectable voltage buffer can be set to high-impedance generation with the Tristate Control line. The Tristate Control can be set automatically by the <u>Initial and Idle States</u> or can be set programmatically with the <u>niHSDIO Tristate Channels</u> VI or <u>niHSDIO_TristateChannels</u> function. When a channel is configured for <u>open collector</u> generation (supported for static generation only), the tristate control transforms logic 1's into Z states (high-impedance).

The protection diodes are critical for guarding against overvoltage situations. Refer to <u>Input Protection</u> for more information about this portion of the channel electronics.

#### **Dynamic Acquisition**

Patterns acquired by the NI 654x are received using a selectable voltage buffer. Refer to the <u>NI 654x Specifications</u> for the input voltage thresholds for each logic family setting.

The output of the selectable voltage buffer is sampled by the Pattern Acquisition Timing and Control module before being sent to the Pattern Acquisition Engine for storage into Acquisition Memory.

The *input impedance* is high impedance, referenced to ground.

## **Voltage Ranges and Settings**

The NI 654*x* uses the following four <u>logic families</u> to control the device voltage levels:

- 1.8 V Logic
- 2.5 V Logic
- 3.3 V Logic
- 5.0 V Logic

Selecting one of these logic families determines the NI 654*x* generation levels or acquisition thresholds. For mixed-voltage systems the generation levels and acquisition thresholds can be set independently. For example, you can set the Generation Voltage Family to 1.8 V Logic while you set the Acquisition Voltage Family to 5.0 V Logic. By synchronizing and operating multiple NI 654*x* devices, you can use multiple generation or acquisition voltage configurations in the same system.

 $\overline{\mathbb{N}}$ 

**Note** The maximum generation voltage of the NI 654*x* is 3.3 V. Therefore, setting the Generation Logic Family to 5.0 V Logic produces a 3.3 V signal, which is compatible with standard 5 V TTL input thresholds. Refer to the <u>NI 654*x* specifications</u> for details.

The Generation Logic Family selection sets the voltage levels used for all data, clock, and event generations, including the exported <u>PLL reference</u> <u>clock</u>. The Acquisition Logic Family selection sets the levels used for all acquired data, clock, and trigger signals, except for the CLK IN SMB jack connector. Refer to <u>Clock Sources Summary</u> for acceptable CLK IN signal characteristics.

When programming your device using NI-HSDIO, the functions and instance VIs for <u>configuring voltage levels</u> are named according to the type of channel (data, trigger, or event) that you want to configure.

For more information about voltage level ranges and resolutions, refer to the <u>NI 654x specifications</u>.

## **Logic Families**

The NI 654*x* Acquisition and Generation Logic Families can be independently set to one of four values: 1.8 V Logic, 2.5 V Logic, 3.3 V Logic, and 5.0 V Logic. The actual <u>voltage levels</u> defined by these logic families are defined in the <u>NI 654*x* specifications</u>.

Setting the Generation Logic Family determines the generation voltage levels (Generation Voltage High and Low) for the device. The NI 654x generates these voltages for a 1 (Generation Voltage High) or a 0 (Generation Voltage Low). The following diagram clarifies the physical meaning of generation voltage level specifications.



Setting the Acquisition Logic Family determines the acquisition voltage levels (Acquisition Voltage High and Low) for the device. An acquired voltage below Acquisition Voltage Low is read as a 0, a voltage acquired above Acquisition Voltage High is read as a 1.



**Note** Any acquired voltage between Acquisition Voltage Low and Acquisition Voltage High cannot be guaranteed to be read as a 1 or a 0. In the following figure, these samples are shown as an X, indicating that the acquired value is unknown.

The following diagram illustrates the physical meaning of the acquisition voltage level specifications.



#### **Related Topics:**

Logic Families Overview

- Single-Ended Voltage Levels
- <u>Voltage Ranges and Settings</u> (NI 654*x*)

## Input Impedance

The input impedance of the NI 654*x* is high impedance. Refer to the NI 654*x* specifications for more information on the high-impedance values for your device. Selectable input impedance is only available with the NI 655*x* products.

Refer to <u>Configuring Input Impedance</u> for more information on configuring this property.

## Source Impedance

The NI 654*x* data, clock, and event generation channels have a 50  $\Omega$  source impedance. For applications where the full voltage swing is required at the DUT, a parallel termination resistance of 1 k $\Omega$  to 10 k $\Omega$  is recommended. With a system terminated by 10 k $\Omega$ , much of the signal reflections are eliminated by the source 50  $\Omega$  termination and the parallel termination. Thus the voltage seen at the termination resistor is 99.5% of the configured voltage.

Because the NI 654*x* interface cable (NI SHC68-C68-D2) is a 50  $\Omega$  transmission line, when you use 3.3 V Logic, you can build matched impedance systems with a 50  $\Omega$  parallel termination as the load. While a matched system is beneficial because all reflections are eliminated, the voltage at the termination is one-half of the generation voltage level. The change in voltage is caused by the voltage divider that is created by the source and termination impedance, as shown in the following figure.

Use the following formula to calculate the voltage sensed at the termination point,  $V_{\text{TERM}}\!.$ 

 $V_{\text{Term}} = V_0 \times \frac{R_{\text{Term}}}{R_{\text{Term}} + R_{\text{Source}}} = V_0 \times \frac{R_{\text{Term}}}{R_{\text{Term}} + 50}$ 

where  $V_O$  is the voltage driven by the NI 654*x*,

 $R_{Term}$  is the termination impedance,

R_{Source} is the source impedance

For example, if  $R_{SOURCE} = 50 \Omega$  and if the termination resistance is also set to 50  $\Omega$ , then the voltage level seen at the termination is one-half the source voltage.

Always calculate the maximum current that the NI 654x in your test system can source and sink. You can calculate the maximum current using the following formula:

Max current =  $V_{OH (max)}/(50 + R_{Term})$ 

For example, if the device is driving 1.8 V into a 50  $\Omega$  load, the maximum current would be calculated as follows:

 $I_{max} = 1.8 V/(50 \Omega + 50 \Omega) = 18 mA$ 

18 mA is higher than the NI 654x current specification at 1.8 V Logic operation; therefore, the device should not be connected to a 50  $\Omega$  load when using the 1.8 V Logic family.



**Note** Refer to the <u>NI 654x specifications</u> for details on the maximum current that the NI 654*x* can source for each generation voltage setting.

The NI 654*x* generation lines can be programmatically set to a highimpedance (tristate) state when not in use. Upon power up, <u>DDC CLK</u> <u>OUT</u>, <u>CLK OUT</u>, <u>DIO<0..31></u>, and <u>PFI <0..3></u> are set to tristate and remain in that state until configured for generation.

Refer to <u>Termination</u> and <u>Terminating Your Module</u> for more information about signal reflections and termination.

### **Input Protection**

DDC CLK OUT, CLK OUT, DIO<0..31>, PFI <0..3>, and STROBE are protected using diode clamps connected to positive and negative voltage supplies. The following figure illustrates this circuit.



The entire I/O circuit is shown in the Channel Electronics diagram.

These diodes act as open circuits unless the I/O voltage levels go above  $V_{p+}$  or below  $V_{p-}$ . When the I/O voltage exceeds  $V_{p+}/V_{p-}$ , the diodes become short circuits, clamping the input voltage to  $V_{p+}/V_{p-}$ . Therefore, these diodes prevent input voltages from going more than a diode drop, or approximately 0.5 V, beyond the positive or negative protection rails. The following diagram demonstrates the effect of the clamp.



The NI 654x is protected from instantaneous shorts to legal DUT voltages. Refer to the NI 654x specifications for details on input protection.

## **Signal Routing**

The NI 654*x* is capable of sending and receiving signals through the front panel and through the PXI trigger bus (for <u>PXI bus computers</u>) or the RTSI trigger bus (for <u>PCI bus computers</u>).

The <u>front panel connectors</u> provide connectivity for the bidirectional DIO channels as well as for control lines for sending and receiving clocks, triggers, and events.

### **Signal Routing for PXI Devices**

You can use the PXI trigger bus to send and receive events, triggers, and Sample and Reference clocks. For more information about triggers, events, and clocks, refer to <u>Dynamic Generation Triggers and Events</u>, <u>Dynamic Acquisition Triggers and Events</u>, and <u>Clocking</u>.

All signal routing operations can be characterized by a *source* and a *destination*. The following table summarizes the possible sources and destinations for NI 654x/655x/656x signals. Sources are listed in the far left column, and the possible destinations span the top of the table. Some of the internal signals are broken into separate columns for generation (Gen) and acquisition (Acq).

<b>Destinations</b> →	Front Panel			Backplane			
Sources ↓	CLK OUT	PFI 0	PFI <13>	DDC CLK OUT	PXI_TRIG <06>	PXI_TRIG7	PXI_
Front Panel							
CLK IN	$\checkmark$			$\checkmark$			
PFI 0			$\checkmark$	_	$\checkmark$	_	
PFI <13>		$\checkmark$	$\checkmark$		$\checkmark$	_	
Backplane							
PXI_TRIG<07>	_	$\checkmark$	$\checkmark$		$\checkmark$	_	
PXI_STAR	$\checkmark$			$\checkmark$			
Internal							
Sample Clock	$\checkmark$			$\checkmark$			
Reference Clock	$\checkmark$			_	—	—	
Start Trigger (Gen)		$\checkmark$	$\checkmark$		$\checkmark$		
Pause Trigger (Gen)		$\checkmark$	$\checkmark$		$\checkmark$		

ScriptTrigger<03>		$\checkmark$	$\checkmark$		$\checkmark$		
Start Trigger (Acq)		$\checkmark$	$\checkmark$		$\checkmark$	_	
Advance Trigger (Acq)		$\checkmark$	V		$\checkmark$		
Ready for Start Event (Gen)	_	$\checkmark$	$\checkmark$	_	$\checkmark$	—	
Marker<03> Event	_	$\checkmark$	$\checkmark$	_	$\checkmark$	—	
Data Active Event		$\checkmark$	$\checkmark$				
Reference Trigger (Acq)	_	$\checkmark$	$\checkmark$	—	$\checkmark$	_	
Ready for Start Event (Acq)	_	$\checkmark$	$\checkmark$	_	$\checkmark$	—	
Ready for Advance Event (Acq)		$\checkmark$	$\checkmark$		$\checkmark$	_	
End of Record Event (Acq)		$\checkmark$	$\checkmark$		$\checkmark$		

### **Signal Routing for PCI Devices**

You can use the RTSI trigger bus to send and receive events, triggers, and Sample and Reference clocks. For more information about triggers, events, and clocks, refer to <u>Dynamic Generation Triggers and Events</u>, <u>Dynamic Acquisition Triggers and Events</u>, and <u>Clocking</u>.

All signal routing operations can be characterized by a *source* and a *destination*. The following table summarizes the possible sources and destinations for NI 654x/655x/656x signals. Sources are listed in the far left column, and the possible destinations span the top of the table. Some of the internal signals are broken into separate columns for generation (Gen) and acquisition (Acq).

<b>Destinations</b> →		Front	Panel		Backp	olane		
Sources ↓	CLK OUT	PFI 0	PFI <13>	DDC CLK OUT	RTSI <06>	RTSI 7	Start Trigger (Gen)	Pause Trigger (Gen)
Front Panel								
CLK IN	$\checkmark$			$\checkmark$				
PFI 0			$\checkmark$		$\checkmark$		$\checkmark$	$\checkmark$
PFI <13>		$\checkmark$	$\checkmark$		$\checkmark$		$\checkmark$	$\checkmark$
Backplane								
RTSI <07>	—	$\checkmark$	$\checkmark$	—	$\checkmark$		$\checkmark$	$\checkmark$
Internal								
Sample Clock	$\checkmark$			$\checkmark$				
Reference Clock	V							
Onboard Ref Clock						$\checkmark$		
Start Trigger (Gen)		$\checkmark$	V		V		—	$\checkmark$
Pause Trigger (Gen)		$\checkmark$	V		V		V	

ScriptTrigger<03>		$\checkmark$	$\checkmark$	—	$\checkmark$	 $\checkmark$	$\checkmark$
Start Trigger (Acq)		$\checkmark$	V		$\checkmark$	 	
Advance Trigger (Acq)		$\checkmark$	$\checkmark$		$\checkmark$	 	
Ready for Start Event (Gen)	_	$\checkmark$	V		$\checkmark$	 $\checkmark$	V
Marker<03> Event		$\checkmark$	$\checkmark$		$\checkmark$	 $\checkmark$	$\checkmark$
Data Active Event	_	$\checkmark$	$\checkmark$			 	
Reference Trigger (Acq)		$\checkmark$	$\checkmark$		$\checkmark$	 	_
Ready for Start Event (Acq)		$\checkmark$	$\checkmark$		$\checkmark$	 	_
Ready for Advance Event (Acq)		$\checkmark$	$\checkmark$		$\checkmark$	 	
End of Record Event (Acq)		$\checkmark$	$\checkmark$		$\checkmark$		

## Clocking

The following figure shows how the <u>clock sources</u> are routed to produce the NI 654x <u>clock signals</u>.



### **Clock Sources Summary**

The following tables describe the clock sources available for the NI 654*x*. These clock sources are shown in the <u>Clocking</u> diagram. For a more general description of these clocks, refer to <u>Clocks for Digital Waveform</u> <u>Generator/Analyzers</u>.

Clock Source	Used In	Location	Description
On Board Clock	Acquisition, Generation	Internal	The NI 654 <i>x</i> provides a single high- precision 200 MHz voltage- controlled crystal oscillator (VCXO) clock source. The NI 654 <i>x</i> can generate any clock frequency of 200 MHz/ <i>n</i> , where <i>n</i> is any integer from 2 to 4,194,304 for the NI 6542, and 4 to 4,194,304 for the NI 6541. For example, for the NI 6542, the On Board Clock can run at 100 MHz, 66.67 MHz, 50 MHz, 40 MHz, 33.33 MHz, 28.57 MHz, 25 MHz, 22.22 MHz, and so on. The onboard PLL allows the On Board Clock to be phase-locked to the Reference clock, if one is provided.
CLK IN	Acquisition, Generation	Front panel SMB jack connector	The CLK IN SMB jack is intended for use as an external frequency input channel, allowing you to provide an alternate frequency as the Sample clock rate. The CLK IN signal can be any sine or square wave signal that meets the specifications provided in the <u>NI 654x specifications</u> . The CLK In signal must be free running.
PXI_STAR (NI PXI- 6541/6542 only)	Acquisition, Generation	Backplane	The PXI_STAR connector can be used as an external frequency input channel, allowing you to provide an alternate frequency as the Sample clock rate. The PXI_STAR signal specifications are provided in the <u>NI 654x specifications</u> . The PXI_STAR signal must be free

	running.

Reference	Clock		
Clock Source	Used In	Location	Description
NONE	Acquisition, Generation	Internal	When no reference clock source is selected, the PLL is not locked and the On Board Clock has no known phase relationship to any other clocks in the system.
CLK IN	Acquisition, Generation	Front panel SMB jack connector	The CLK IN SMB jack can be used to provide an external Reference clock for the PLL. The CLK IN signal can be any sine or square wave signal that meets the specifications provided in the <u>NI 654x</u> <u>specifications</u> , and must be free running.
PXI_CLK10 (NI PXI- 6541/6542 only)	Acquisition, Generation	PXI trigger bus	The PXI Clock 10 line exists on the PXI backplane and provides a 10 MHz reference clock to all slots in the chassis. The PLL can be configured to lock to this signal.
RTSI 7 (NI PCI- 6541/6542 only)	Acquisition, Generation	RTSI trigger bus	The Onboard Reference Clock can be routed to RTSI 7 to provide a 10 MHz reference clock signal to the NI 654x and other devices that share the RTSI bus. The PLL can be configured to lock to this signal.

STROBI	E		
Clock Source	Used In	Location	Description
STROBE	Acquisition	DDC connector	STROBE is intended for use as the Sample clock for dynamic acquisition sessions when source-synchronous transfers are desired (that is, when the data and clock travel together through the cable from the DUT to the NI 654 <i>x</i> ). The STROBE signal must be a free- running square wave clock. STROBE is sampled at the same voltage thresholds as the dynamic acquisition data lines.

# **Exporting a Clock**

The NI 654*x* provides several resources for exporting clocks. The <u>Clocking</u> block diagram shows how the NI 654*x* exports these clocks.



**Note** As shown in the <u>clocking block diagram</u>, it is possible to export both the Reference clock and the Sample clock at the same time if you route the Reference clock to CLK OUT and the Sample clock to DDC CLK OUT.

For information about using NI-HSDIO to export clocks, refer to the <u>niHSDIO Export Signal</u> VI or the <u>niHSDIO_ExportSignal</u> function.

Clock Destination Description DDC CLK Sample Clock DDC CLK OUT on DDC OUT connector CLK OUT CLK OUT SMB jack connector **Reference Clock** CLK OUT CLK OUT SMB jack connector **Onboard Reference Clock (NI** RTSI 7 **RTSI** trigger bus PCI-654x only) channel 7

The following table summarizes the possible exported clock options.

#### Sample Clock

The Sample clock can be exported to one of two destinations: the DDC CLK OUT pin on the DDC connector or the CLK OUT SMB jack connector.

- DDC CLK OUT—The Sample clock can be exported to the DDC CLK OUT pin on the DDC connector. The exported Sample clock is generated at the logic family voltage levels specified for dynamic generation. For dynamic generation sessions, exporting the Sample clock to this connector allows for source-synchronous clocking by routing the Sample clock through the same cable and propagation delay characteristics as the generated data.
- **CLK OUT**—The Sample clock can be exported to the CLK OUT SMB connector. The exported Sample clock is generated at the logic family voltage levels specified for dynamic generation.

You can export the Sample clock to either the DDC connector or the CLK OUT SMB jack connector, but *not* at the same time.

#### **Reference Clock**

If you configure a Reference clock for the PLL on the NI 654*x*, you can export the reference clock to the CLK OUT SMB jack connector. The exported reference clock operates at the logic family voltage levels specified for dynamic generation.

#### **Onboard Reference Clock**

If you are using an NI PCI-654x, you can export the 10 MHz onboard reference clock to RTSI 7 on the RTSI trigger bus. You can then use a RTSI cable to connect this signal to other PCI devices.

## **Channel Interface**

The NI 654*x* has 32 channels. Each channel is independently configurable for generation, acquisition, or simultaneous generation and acquisition operations. Generation and acquisition logic families on the NI 654*x* are independently selectable—you can select one logic family for all generation channels and a different logic family for all acquisition channels.

The following topics provide more information about the channel interface:

- Front Panel and Connector Pinout
- LED Indicators

#### **Front Panel and Connector Pinout**

The NI 654*x* front panel, shown below, has three SMB jack connectors and one 68-pin Digital Data & Control (DDC) VHDCI connector. The SMB jack connectors are described in the <u>SMB Jack Connector Names and</u> <u>Descriptions</u> table. The DDC connector signals are described in the <u>DDC</u> <u>Connector Names and Descriptions</u> table.



Child Guok Connector Maines and Descriptions	SMB	Jack	Connector	Names	and	<b>Descriptions</b>
----------------------------------------------	-----	------	-----------	-------	-----	---------------------

Connector	Signal Name	Signal Type	Signal Description
CLK IN	Reference/Clock Input	Control	External reference clock used for the PLL or for the external Sample clock used for pattern generation and/or acquisition.
PFI 0	Programmable Function Interface (PFI) 0	Control	Input terminal to the NI 654 <i>x</i> for external triggers or the output terminal from the NI 654 <i>x</i> for events.
CLK OUT	Reference/Clock Output	Control	Terminal for the exported PLL Reference clock or the exported Sample clock.

DDC Connector Names and Descriptions					
Pins	Signal Name	Signal Type	Signal Description		
33	DDC CLK OUT	Control	Terminal for the exported Sample clock.		
67	STROBE	Control	External Sample clock source which can be used for dynamic acquisition.		
1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65	DIO <031>	Data	Bidirectional digital I/O data channels 0 through 31.		
26, 30, 64	PFI<13>	Control	Input terminals to the NI 654 <i>x</i> for external triggers, or output terminals from the NI 654 <i>x</i> for events.		
2, 4, 6, 10, 12, 14, 16, 18, 20, 22, 24, 28, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 54, 56, 58, 62, 66	GND	Ground	Ground reference for signals.		
8, 52, 60	RESERVED	N/A	These terminals are reserved for future use. Do not connect to these pins.		

# LED Indicators (PXI Only)

The NI PXI-654*x* has two LED indicators on the front panel, labeled ACCESS and ACTIVE. The following tables describe what each LED color indicates.

#### ACTIVE LED

Color	Indications
Off	Device not armed, not triggered, or experiencing an error.
Amber	Device armed and awaiting Start trigger. If performing a dynamic acquisition operation, the device may be acquiring pretrigger samples.
Green	Device received Start trigger.
Red	Error condition.

#### ACCESS LED

Color	Indications
Off	Device not ready.
Amber	Device being accessed by software.
Green	Device ready to be programmed.
Red	Running the <u>niHSDIO Self Test</u> VI or calling the <u>niHSDIO_self_test</u> function produced a failure.

# Acquisition

Expand this book for more information about <u>static</u> and <u>dynamic</u> <u>acquisition</u> using the NI 654x.

## **Static Acquisition**

Static acquisition is a software-timed (nonclocked) operation. When performing static acquisition operations, the NI 654*x* uses the device logic family to determine and return the current logic state of the configured data channels each time a static read is requested.

You can perform static acquisition operations at any time on channels configured for static acquisition. You can also use static acquisition to read back the current value on channels configured for static generation or dynamic generation at any time.

For more information about performing static acquisition in NI-HSDIO, refer to <u>Reading and Writing Static Data</u>.
# **Dynamic Acquisition**

The NI 654*x* provides flexible acquisition capabilities for up to 32-bit wide patterns with a selectable logic family using either an internal or external clock source. External triggers can control the acquisition operation, and the Pattern Acquisition Engine can route those control signals to be shared with other devices.

Refer to <u>Dynamic Acquisition Clock Sources</u> for information about available clocks for a dynamic acquisition operation.

For information about defining acquisition resources in NI-HSDIO, refer to <u>Acquisition Configuration Functions</u> to learn which VIs and C functions are available for your application.

## **Dynamic Acquisition Clock Sources**

Dynamic acquisition is a clocked operation driven by one of several clocking resources. Refer to the main <u>Clocking</u> diagram for this device to see a block diagram for these clock resources.

#### **Dynamic Acquisition State Diagram**

The following figure illustrates the state diagram for NI 654x/NI 655x/NI 656x dynamic acquisition.



In this diagram, the device starts in an <u>Idle state</u>, where it is configured for the acquisition operation. Once configured, the device moves into a Wait for Start Trigger state, where it generates the <u>Ready for Start event</u> to indicate that the Pattern Acquisition Engine is configured and ready to receive a <u>Start trigger</u>.

After receiving the Start trigger, the device moves into a sampling state where it starts sampling pre-Reference trigger data. After the minimum number of pre-Reference trigger samples is acquired, the device continues sampling and waits for the <u>Reference trigger</u>. When the device receives the Reference trigger, a counter begins counting Sample clocks. After a predetermined number of samples are acquired, the device transitions to a Record Complete state, and generates the <u>End of Record event</u>.

At this point, if the acquisition is only configured to acquire one record, the device transitions to the Done state. If the device has more records to acquire, after the Record Complete state, the device transitions to the Wait for Advance Trigger state, and emits the <u>Ready for Advance event</u>. After receiving the <u>Advance trigger</u>, the device moves back into the

sampling state where it starts sampling pre-Reference trigger data, and starts the process of acquiring another record. This process repeats until all records are acquired, then the device moves into a Done state, and the operation stops.

## **Dynamic Acquisition Timing Diagrams**

The following diagram illustrates the <u>data positions</u> available when acquiring waveforms with the NI 654x. For simplicity, the delayed data is shown delayed by 25% of the clock period; however, this value can vary between 0% and 100%.



#### Using the Sample Clock as the Acquisition Clock

 $t_{\text{DDCSC}}$  : Time Delay from DDC Connector to Internal Sample Clock

 $0 \leq \delta_A \leq 1$  : Pattern Acquisition Data Delay (fraction of  $t_{\text{P}})$ 

 $t_{P} = \frac{1}{f}$  = Period of Sample Clock

t_{SUSC} = Set-Up Time to Sample Clock

 $t_{\text{HSC}}$  = Hold Time to Sample Clock

#### Using STROBE as the Acquisition Clock



t_{SUS} = Set-up Time to STROBE

 $t_{HS}$  = Hold Time from STROBE

 $0 \le \delta_A \le 1$ : Pattern Acquisition Data Delay (percentage of t_P)

 $t_P = \frac{1}{f}$  = Period of Sample Clock

# **Dynamic Acquisition Triggers and Events**

The following table describes the relationship of triggers and events in a dynamic acquisition operation. The sequence of triggers and events is shown in the <u>Dynamic Acquisition State Diagram</u>.

Triggers are received synchronously by the <u>Pattern Acquisition Engine</u>. The Ready for Start event and all re-exported triggers are asynchronously generated by the Pattern Acquisition Engine.

Trigger/Event	Received From	Exported To
Start Trigger	The Start trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<031>. The Start Trigger can also be sent by software.	The Start trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06>.
Reference Trigger	The Reference trigger can be received on a rising edge on PFI <03> RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<031>. The Reference trigger can also be sent by software.	The Reference trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Advance Trigger	The Advance trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<031>. The Advance trigger can also be sent by software.	The Advance trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Pause Trigger	The Pause trigger can be received from PFI<03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching	

	a pattern received on DIO<031>.	
Ready for Start Event		The Ready For Start event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Ready for Advance Event		The Ready For Advance event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
End of Record Event		The End of Record event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

## Generation

Expand this book for more information about static and dynamic generation using the NI 654x.

## **Static Generation**

Static generation is a software timed (nonclocked) operation and can be applied to any number of the available DIO<0..31> channels. When performing a static generation operation, the NI 654*x* sets the current state of the configured data channels to the requested logic state of the configured generation logic family. NI 654*x* devices support the <u>open</u> collector drive type for static generation operations, which means that they can set the configured data channels to a high-impedance (Z) state when the channels are configured to drive the Generation Voltage High Level.

Refer to <u>Voltage Ranges and Settings</u> for more information about voltage levels for each logic state.

Static generation can be done on any number of channels, provided that those channels are not configured for a dynamic operation. To statically set the state of dynamic generation channels, refer to <u>Initial and Idle</u> <u>States</u>.

For more information about performing static generation with NI-HSDIO, refer to <u>Reading and Writing Static Data</u>.

# **Dynamic Generation**

Dynamic generation is a clocked operation where binary data is sent from the NI digital waveform generator/analyzer to the DUT across multiple digital channels.

The NI digital waveform generator/analyzer can generate complex digital patterns at a variety of voltage levels synchronous to any of several clock sources. The data can be generated as simple waveforms or based on complex <u>scripts</u>. External triggers can control the data generation, and the <u>Pattern Generation Engine</u> can export several types of <u>events</u> to indicate the progress of the generation.

In addition, the NI digital waveform generator/analyzer allows for precise <u>subperiod timing control</u> between the generated data and the exported Sample clock, making the device a versatile digital pattern generator.

NI 654*x* devices support only <u>active drive</u> generation during dynamic generation operations.

## **Dynamic Generation Clock Sources**

Dynamic generation is a clocked operation. The dynamic generation operation is clocked by one of several clocking resources. Your application needs may determine which source you should use. Refer to the main <u>Clocking</u> diagram to see a block diagram for these clock resources.

The following section discusses additional considerations for using these clocking resources for dynamic generation:

On Board Clock

The default clock source for dynamic generation sessions is the On Board Clock. This clock can be locked to a reference clock to synchronize operations across multiple devices or can be used without a reference clock when multidevice synchronization is not required. The On Board Clock is derived from integer divisors of the 200 MHz VCXO. Refer to the <u>NI 654x specifications</u> for information about the possible On Board Clock frequencies.

You can configure the On Board Clock source in the following ways:

- Free-running, nonphase-locked—In this mode, the VCXO is used at its fundamental frequency, allowing for a stable and accurate 200 MHz clock. This configuration is the default setting, and it is most useful when only one NI 654x is in the system or when multidevice synchronization is not required.
- Phase-Locked—The On Board Clock source can be locked to a reference clock using the PLL circuit to ensure that Sample clock alignment across devices is achieved. In this operation mode, the PLL circuit must be provided a precision source to which it can lock. The On Board Clock source can be locked to one of the following reference clock sources:
  - PXI_CLK10 (NI PXI-6541/6542 only)/RTSI 7 (NI PCI-6541/6542)—The PXI standard defines a precision 10 MHz reference (PXI_CLK10) to be distributed across the backplane to each device in the PXI chassis. If you are using PXI, this 10 MHz backplane clock is used as the reference for the PLL in this mode

of PLL operation. If you are using PCI, drive the 10 MHz On Board Reference Clock onto RTSI 7, and configure RTSI 7 as the reference clock source.

- **CLK IN**—If you want to provide your own reference, you can provide an external source on the CLK IN SMB connector to which the PLL can lock. Using an external reference allows you to easily synchronize clocks across instruments within and outside of the system. Refer to the <u>NI 654x specifications</u> for information about the possible reference clock frequencies.
- External Source (CLK IN)

Alternatively, your dynamic generation operation can be driven from an external Sample clock source. Using an external frequency generator, you can drive dynamic generation operations at any frequency within the NI 654*x* specifications. Frequency limitations and acquisition levels are listed in <u>NI 654*x*</u> <u>specifications</u>.

• PXI_STAR (NI PXI-6541/6542 only)

The PXI specification allocates resources for high-speed precision clock and trigger routing across the PXI backplane. The NI PXI-654*x* can use this resource to clock your dynamic generation task. An external source can drive this resource at any suitable frequency, allowing the NI PXI-654*x* to operate at noninteger divisors of 200 MHz, similar to how it operates using an external clock source (CLK IN).

For a summary of these and other clock sources, refer to <u>Clock Sources</u> <u>Summary</u>.

#### **Dynamic Generation State Diagram**

The following figure illustrates the state diagram for the NI 654x/655x/NI 656x dynamic generation.



The device starts in an <u>Idle state</u> where it is configured for a generation operation. Once initiated, the device moves into a Wait for Start Trigger state, where it generates the <u>Ready for Start event</u> to indicate that the <u>Pattern Generation Engine</u> is configured and ready to receive a <u>Start trigger</u>. After receiving the Start trigger, the dynamic generation operation begins, and at this point, the generation can be acted upon by <u>Script</u> and <u>Pause</u> triggers, and it can generate the <u>Data Active</u> and <u>Marker</u> events. When the generation is complete, the device moves into the Done state.

# **Dynamic Generation Timing Diagrams**

The following figure illustrates the <u>data</u> and <u>clock</u> positions available when generating waveforms with the NI 654x. For simplicity, the data is shown delayed by 25% of the clock period; however, this value can vary between 0% and 100%.

Note Data generation on the rising clock edge, falling clock edge, or delayed, is per channel selectable. However, in the delayed case, the delay value is constant across all delayed channels.



 $t_{\text{SCDDC}}$  : Time Delay from Sample Clock (Internal) to DDC Connector

 $0 \leq \delta_C \leq 1$  : Exported Sample Clock Delay (Fraction of  $t_p)$ 

 $0 \leq \delta_G \leq 1$  : Pattern Generation Data Delay (Fraction of  $t_p)$ 

 $t_p = \frac{1}{f} = Period of Sample Clock$ 

 $t_{CO}$  = Exported Sample Clock Offset; 0 or 2.5 ns, Software-Selectable

For more information about using NI-HSDIO to adjust the data position, refer to <u>Configuring Data Position</u>.



# Generation Provided Setup and Hold Times Timing Diagram

 $t_{PC0}$  = Time from Rising Clock Edge to Data Transition (Provided Clock to Out Time)

 $t_{CO}$  = Exported Sample Clock Offset

#### **Dynamic Generation Triggers and Events**

The following table describes the relationship of triggers and events in a dynamic acquisition operation. The sequence of triggers and events is shown in the <u>Dynamic Generation State Diagram</u>.

Start, Pause, and Script triggers are received asynchronously by the <u>Pattern Generation Engine</u>. Cable propagation delays and pipeline delays can cause the Pattern Generation Engine to take multiple clock cycles to respond to a trigger. Refer to <u>NI 654x specifications</u> for more information. The <u>Ready For Start</u>, <u>Data Active</u>, and <u>Marker 0 events</u> are generated by the Pattern Generation Engine synchronous to the data generation.

The following table provides information about how these triggers and events can arrive at and be exported from the NI 654x.

Trigger/Event	Received From	Exported To
<u>Start Trigger</u>	The Start trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices). The Start trigger can also be sent by software.	The Start trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Pause Trigger	<ul> <li>The Pause trigger is level-based and can be received on PFI&lt;03&gt;, RTSI&lt;07&gt; (PCI devices), or PXI_TRIG&lt;07&gt; (PXI devices). The Pause trigger can also be sent by software.</li> <li>Note When the Pause trigger is asserted, the NI 654x Pattern Generation Engine may take several clock cycles to respond because of cable propagation delay and the pipelining in the system. Refer to the NI 654x specifications for more</li> </ul>	The Pause trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

	information.	
<u>Script Trigger</u> <03>	Four Script triggers can be edge- or level-based and can be received on PFI <03>, RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices). The Script trigger can also be sent by software.	The Script trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
<u>Ready for</u> <u>Start Event</u>		The Ready For Start event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
<u>Data Active</u> <u>Event</u>		The Data Active event can be exported to PFI <03>.
<u>Marker Event</u> <03>		A Marker event can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

## NI 655*x*

The NI 655*x* is a 20-channel digital I/O device that you can use as a PC/peripheral device interface, pattern generator, pattern analyzer, or stimulus-response tester. The NI 6551 has a maximum Sample clock frequency of 50 MHz, and the NI 6552 has a maximum Sample clock frequency of 100 MHz.

The NI 655*x* also provides the following features:

- Sophisticated timing engine to maintain and measure the timing parameters of the DUT
- <u>Programmable voltage levels</u> for testing multiple devices or characterizing a single device under varying conditions
- Data channels with per channel direction control
- Deep onboard memory with triggering and pattern sequencing capabilities
- Ability to use <u>NI-TClk</u> to synchronize multiple devices
- Per pin, <u>per cycle tristate</u> capabilities
- Support for real-time hardware comparison of actual response data with expected response data. For more information on this feature, refer to the <u>National Instruments High-Speed Digital ATE and</u> <u>Stimulus Response Features</u> white paper on ni.com.

You can use the internal On Board Clock or import an external clock through the front panel. You can also shift the generated data, acquired data, and exported Sample clock relative to the onboard clock for clock frequencies above 25 MHz, which is critical when accounting for propagation delays and setup-and-hold times in the DUT.

Expand this book for more information about NI 655*x* hardware-related topics.

## **Hardware Architecture**

Expand this book for more information about the NI 655*x* hardware architecture.

# **Block Diagram**

The following figure is a block diagram illustrating the main functional units and data flow of the NI 655x. The text that follows the figure describes the basic elements of the diagram and provides links to sections with more detailed information about some of the blocks.





The <u>Clocking</u> module selects and distributes the clocks for the dynamic generation and dynamic acquisition operations.

For dynamic generation operations, the user-supplied data is loaded from the host computer memory into the onboard Generation Memory. The Pattern Generation Engine retrieves data from <u>Generation Memory</u> and executes the script functionality while interacting with the associated Trigger and Event control module. The Pattern Generation Engine then sends the data to the Pattern Generation Timing and Control module, where the data is given the selected data position and data delay and is then sent to the <u>Channel Electronics</u> drivers. The Channel Electronics drivers generate the data at the user-defined voltage levels.

For dynamic acquisition operations, signals arrive at the Channel Electronics comparators, where the signal levels are compared to the user-defined voltage thresholds. The Timing and Control module samples the data using the selected clock, data position, and data delay values and passes the data to the Pattern Acquisition Engine. The Pattern Acquisition Engine and the Trigger and Event Control module recognize triggers and determine when the data should be stored into Acquisition Memory. The acquired data can then be fetched by the host computer.

## **Channel Electronics**

The channel electronics of the NI 655*x* devices consist of a variable voltage driver, dual comparators, the appropriate termination resistors, and I/O protection diodes. Each I/O channel is capable of simultaneously driving and receiving data.

The following figure provides a basic block diagram for the channel electronics. Refer to  $\underline{NI \ 655x \ Block \ Diagram}$  for a picture of how the channel electronics fits into the overall block diagram.



#### **Dynamic Generation**

For dynamic generation operations, the data appears at the input of the variable voltage driver after the Pattern Generation Timing and Control module gives the data the selected data position and data delay. The variable voltage driver converts the data signal to the user-defined voltage levels (Generation Voltage High and Generation Voltage Low) before sending the data signal to the DDC connector on the NI 655*x* front panel.

The variable voltage driver can be set to high-impedance generation with the Tristate control line. The Tristate control can be set automatically by the Initial and Idle States or can be set programmatically with the niHSDIO Tristate Channels VI or niHSDIO TristateChannels function. When a channel is configured for open collector generation, the tristate control transforms logic 1's into Z states (high-impedance).

The protection diodes are critical for guarding against overvoltage situations.

#### **Dynamic Acquisition**

Patterns acquired by the NI 655*x* are received using a dual-comparator architecture. One comparator is assigned to each acquisition threshold (Acquisition Voltage High and Acquisition Voltage Low).

The output of the dual comparators are combined into a single bit using the selected <u>data interpretation</u> method. The data is sampled by the Pattern Acquisition Timing and Control module before being sent to the Pattern Acquisition Engine for storage into <u>Acquisition Memory</u>.

You can programmatically set the input impedance to high impedance or 50  $\Omega,$  referenced to ground.

#### **Voltage Ranges and Settings**

For testing functional limits, you can set custom voltage levels in 10 mV increments from –2 V to +5.5 V. This range allows compatibility with common logic families such as CMOS, TTL, and LVTTL, as well as custom logic levels. The <u>Channel Electronics</u> diagram shows how these voltage levels are used with the dual comparators and the variable voltage driver.

You can set four voltage levels on the NI 655*x*: Generation Voltage High and Low Levels and Acquisition Voltage High and Low Level. Generation voltage levels are the voltage levels used for all data, clock, and event generations, with the exception of the exported PLL reference clock, which is fixed at 3.3 V Logic. Acquisition voltage levels are used for all acquired data, clock, and trigger signals, except for the CLK IN SMB jack connector. Refer to <u>Clock Sources Summary</u> for acceptable CLK IN signal characteristics.

When programming your device using NI-HSDIO, the functions and instance VIs for configuring voltage levels are named according to the type of channel (data, trigger, or event) that you want to configure. For more information about configuring voltage levels with NI-HSDIO, refer to <u>Configuring Voltage Levels</u>.

To stay within PXI and PCI power and cooling requirements, the allowable generation voltage levels are reduced at higher Sample clock frequencies. The following table shows the relationship between Sample clock frequency and allowable generation voltage levels.

Typical Sample Clock Application Frequency		Minimum Level	Maximum Level
Negative logic	Up to 50 MHz	–2.0 V	3.7 V
5 V Logic	Up to 50 MHz	–0.5 V	5.5 V
3.3 V Logic Up to 50 MHz (NI 6551) Up to 100 MHz (NI 6552)		–0.5 V	3.7 V

#### NI 655*x* Generation Voltage Level Limits

You can independently set <u>Generation Voltage High</u> and <u>Generation</u> Voltage Low anywhere from the minimum level to the maximum level shown in the table, depending on your application and Sample clock frequency. <u>Acquisition Voltage High</u> and <u>Acquisition Voltage Low</u> can always be independently set anywhere between –2.0 and 5.5 V, regardless of the Sample clock frequency. By synchronizing and operating multiple NI 655*x* devices, you can use different voltage configurations in the same system.

For more information about voltage level ranges and resolutions, refer to <u>NI 655x specifications</u>.

## **Data Interpretation**

For any NI 655*x* acquisition, the outputs of the <u>dual comparators</u> must be combined into a single bit before being sampled and returned to the software. One comparator uses the Acquisition Voltage Low threshold for comparison, and the other uses the Acquisition Voltage High threshold. You can set these thresholds independently and then configure the data interpretation method to determine the signal behavior, based on the context of these thresholds. The following sections describe the data interpretation in more detail.

#### **High or Low**

High or Low is the default data interpretation method, and this method mimics the data-sheet performance of most digital semiconductors. Using this setting, the NI 655*x* provides some <u>hysteresis</u> for your acquired signal. When the input signal is sampled below Acquisition Voltage Low, a 0 is received. A 1 is not recognized until the acquired signal passes above Acquisition Voltage Low threshold *and* above Acquisition Voltage High threshold. Conversely, if the acquired signal was last sampled above Acquisition Voltage High (as a 1), the signal is not be sampled as a 0 until the signal is sampled below Acquisition Voltage High *and* below Acquisition Voltage Low. In short, signals with voltage in the mid-band (between Acquisition Voltage High and Acquisition Voltage Low) are recognized at the last valid logic level (either above Acquisition Voltage High or Acquisition Voltage Low) at which they were sampled.



#### Valid or Invalid

This data interpretation method returns an indication of whether the acquired signal is between Acquisition Voltage High and Acquisition Voltage Low. Signals sampled between Acquisition Voltage High and Acquisition Voltage Low (in the mid-band, or Invalid) are returned as a 1, while signals sampled either above Acquisition Voltage High or below Acquisition Voltage Low (Valid) are returned as a 0.



For information about configuring data interpretation with NI-HSDIO, refer to <u>Configuring Data Interpretation</u>.

# Logic Families

NI 655x use logic families with the voltage levels shown in the following table:

NI-HSDIO Logic Family	Acquisition Voltage Low	Acquisition Voltage High	Generation Voltage Low	Generation Voltage High
5.0 V Logic	1.8 V	2.0 V	0.0 V	5.0 V
3.3 V Logic	1.6 V	1.7 V	0.0 V	3.3 V
2.5 V Logic	1.2 V	1.3 V	0.0 V	2.5 V
1.8 V Logic	0.85 V	0.95 V	0.0 V	1.8 V

You can also configure custom voltage levels for your operation.

#### **Related Topics:**

- <u>Configuring Voltage Levels</u>
- Logic Families Overview
- Single-Ended Voltage Levels
- Voltage Ranges and Settings (NI 655x)

## Input Impedance

You can programmatically set the input impedance of DIO<0..19> and STROBE to be 50  $\Omega$  or high-impedance. Refer to the NI 655*x* specifications for more information on the high-impedance values for your device. The main application of 50  $\Omega$  is in matched-impedance systems. In these systems, the DUT source impedance is 50  $\Omega$  and the NI 655*x* input impedance is 50  $\Omega$ . This system has the benefit of no signal reflections, at the cost of one-half the signal amplitude. For applications where minor reflections can be tolerated, 50 k $\Omega$  input impedance is appropriate. Refer to Transmission Lines and Terminating Your Module for more information about signal reflections and termination.

When using an input impedance of 50  $\Omega$ , keep in mind that the inherent voltage divider, shown in the following graphic, can cause the voltage levels sensed on the data lines to be affected. Using the 50  $\Omega$  impedance setting can cause the measured voltage level to be lower than the voltage at the source, depending on source impedance. To compensate for this attenuation, set the acquisition voltage thresholds accordingly.



You can calculate the voltages measured by the NI 655x with the following formula:

 $V_{I/O} = V_{Source}(R_{Term}/(R_{Term} + R_{Source}))$ 

where  $V_{I/O}$  is the voltage seen at the NI 655*x* connector

 $V_{\rm S}$  is the voltage driven by the source

R_{Term} is the input impedance

 $R_{\rm S}$  is the source impedance

For example, if the NI 655x input impedance is set to 50  $\Omega$ , a 5 V acquisition is sensed as 2.5 V if driven from a 50  $\Omega$  source.

Always calculate the maximum current that you may be causing the NI 655x to sink. You can calculate the maximum current by using the following formula:

Max current =  $(max(V_{source}))/(R_{source} + R_{term})$ 

Verify that the resulting max current is within the NI 655x specifications.

 $\overline{\mathbb{N}}$ 

**Note** Channels configured for simultaneous generation and acquisition operations have an input impedance of 50  $\Omega$  to the voltage being generated by the variable voltage driver because the NI 655*x* variable voltage driver is configured to have a 50  $\Omega$  source impedance.

Related Topic: Configuring Input Impedance

# Source Impedance

The NI 655*x* data, clock, and event generation channels have a 50  $\Omega$  source impedance. For applications where the full voltage swing is required at the DUT, a parallel termination resistance of 1 k $\Omega$  to 10 k $\Omega$  is recommended. With a system terminated by 10 k $\Omega$ , the majority of the signal reflections are eliminated by the source 50  $\Omega$  termination and the parallel termination, and the voltage seen at the termination resistor is 99.5% of the configured voltage.

Because the NI 655*x* interface cable (NI SHC68-C68-D2) is a 50  $\Omega$  transmission line, it is possible to build matched impedance systems with a 50  $\Omega$  parallel termination as the load. While a matched system is beneficial because all reflections are eliminated, the voltage at the termination is a voltage division of the voltage generation of the NI 655*x*, as illustrated in the following figure.

Use the following formula to calculate the voltage sensed at the termination point,  $V_{\text{TERM}}\!.$ 

$$V_{\text{Term}} = V_0 \times \frac{R_{\text{Term}}}{R_{\text{Term}} + R_{\text{Source}}} = V_0 \times \frac{R_{\text{Term}}}{R_{\text{Term}} + 50}$$

where  $V_O$  is the voltage driven by the NI 655*x*,

 $R_{Term}$  is the termination impedance

*R*_{Source} is the source impedance

For example, if  $R_{SOURCE} = 50 \Omega$  and if the termination resistance is also set to 50  $\Omega$ , then the voltage level seen at the termination is one-half the source voltage.

Always calculate the maximum current that the NI 655x in your test system can source and sink. You can calculate the maximum current using the following formula:

```
Max current = max{|V_{OH}|, |V_{OL}|}/(50 + R<sub>Term</sub>)
```



**Note** Refer to the <u>NI 655x specifications</u> for details on the maximum current that the NI 655x can source.

The NI 655*x* generation lines can be programmatically set to a highimpedance (tristate) state when not in use. Upon power up, DIO<0..19> and PFI <0..3> are set to high-impedance and remain in that state until configured for generation.

Refer to <u>Termination</u> and <u>Terminating Your Module</u> for more information about signal reflections and termination.
### **Input Protection**

DIO<0..19>, PFI <0..3>, DDC CLK OUT, and STROBE are protected using diode clamps connected to the positive and negative voltage supplies. The following figure illustrates this circuit.



The entire I/O circuit is shown in the <u>Channel Electronics</u> diagram.

These diodes act as open circuits unless the I/O voltage levels go above  $V_{p+}$  or below  $V_{p-}$ . When the I/O voltage exceeds  $V_{p+}/V_{p-}$ , the diodes become short circuits, clamping the input voltage to  $V_{p+}/V_{p-}$ . Therefore, these diodes prevent input voltages from going more than a diode drop, or approximately 0.5 V, beyond the positive or negative protection rails. The following diagram demonstrates the effect of the clamp.



The NI 655x is protected from instantaneous shorts to legal DUT voltages. Refer to the <u>NI 655x specifications</u> for details on device input protection.

# **Signal Routing**

NI digital waveform generator/analyzers are capable of sending and receiving signals through the front panel and through the PXI trigger bus (for <u>PXI bus computers</u>) or the RTSI trigger bus (for <u>PCI bus computers</u>).

The front panel connectors provide connectivity for the bidirectional DIO channels as well as for control lines for sending and receiving clocks, triggers, and events. Refer to the Front Panel and Connector Pinout topic for your <u>device</u> for more information about the front panel connectivity.

# Clocking

The following figure shows how the <u>clock sources</u> are routed to produce the NI 655x <u>clock signals</u>.



### **Clock Sources Summary**

The following tables describe the clock sources available for the NI 655*x*. These clock sources are shown in the <u>Clocking</u> diagram. For a more general description of these clocks, refer to <u>Clocks for Digital Waveform</u> <u>Generator/Analyzers</u>.

Clock Source	Used In	Location	Description
On Board Clock	Acquisition, Generation	Internal	The NI 655 <i>x</i> provides a single high- precision 200 MHz voltage- controlled crystal oscillator (VCXO) clock source. The NI 655 <i>x</i> can generate any clock frequency of 200 MHz/ <i>n</i> , where <i>n</i> is any integer from 2 to 4,194,304 for the NI 6552, and 4 to 4,194,304 for the NI 6551. For example, for the NI 6552, the On Board Clock can run at 100 MHz, 66.67 MHz, 50 MHz, 40 MHz, 33.33 MHz, 28.57 MHz, 25 MHz, 22.22 MHz, and so on. The onboard PLL allows the On Board Clock to be phase-locked to the Reference clock, if one is provided.
CLK IN	Acquisition, Generation	Front panel SMB jack connector	The CLK IN SMB jack is intended for use as an external frequency input channel, allowing you to provide an alternate frequency as the Sample clock rate. The CLK IN signal can be any sine or square wave signal that meets the specifications provided in the <u>NI 655x specifications</u> . The CLK IN signal must be free running.
PXI_STAR (NI PXI- 6551/6552 only)	Acquisition, Generation	Backplane	The PXI_STAR connector can be used as an external frequency input channel, allowing you to provide an alternate frequency as the Sample clock rate. The PXI_STAR signal specifications are provided in the <u>NI 655x specifications</u> . The PXI_STAR signal must be free

	running.

Reference Clock					
Clock Source	Used In	Location	Description		
NONE	Acquisition, Generation	Internal	When no reference clock source is selected, the PLL is not locked and the On Board Clock has no known phase relationship to any other clocks in the system.		
CLK IN	Acquisition, Generation	Front panel SMB jack connector	The CLK IN SMB jack can be used to provide an external reference clock for the PLL. The CLK IN signal can be any sine or square wave signal that meets the specifications provided in the <u>NI 655x</u> <u>specifications</u> . The CLK IN signal must be free running.		
PXI_CLK10 (NI PXI- 6551/6552 only)	Acquisition, Generation	PXI trigger bus	The PXI Clock 10 line exists on the PXI backplane and provides a 10 MHz reference clock to all slots in the chassis. The PLL can be configured to lock to this signal.		
RTSI 7 (NI PCI- 6551/6552 only)	Acquisition, Generation	RTSI trigger bus	The Onboard Reference Clock can be routed to RTSI 7 to provide a 10 MHz reference clock signal to the NI 655x and other devices that share the RTSI bus. The PLL can be configured to lock to this signal.		

STROBE						
Clock Source	Used In	Location	Description			
STROBE	Acquisition	DDC connector	STROBE is intended for use as the Sample clock for dynamic acquisition sessions when source-synchronous transfers are desired (that is, when the data and clock travel together through the cable from the DUT to the NI 655 <i>x</i> ). The STROBE signal must be a free- running square wave clock. STROBE is sampled at the same voltage thresholds as the dynamic acquisition data lines.			

# **Exporting a Clock**

The NI 655*x* provides several resources for exporting clocks. The <u>Clocking</u> block diagram shows how the NI 655*x* exports these clocks.



**Note** As shown in the Clocking block diagram, it is possible to export both the reference clock and the Sample clock at the same time if you route the Reference clock to CLK OUT and the Sample clock to DDC CLK OUT.

For information about using NI-HSDIO to export clocks, refer to the <u>niHSDIO Export Signal</u> VI or the <u>niHSDIO_ExportSignal</u> function.

Clock	Destination	Description
Sample Clock	DDC CLK OUT	DDC CLK OUT on DDC connector
	CLK OUT	CLK OUT SMB jack connector
Reference Clock	CLK OUT	CLK OUT SMB jack connector
Onboard Reference Clock (NI PCI-655 <i>x</i> only)	RTSI 7	RTSI trigger bus channel 7

The following table summarizes the possible exported clock options.

#### Sample Clock

The Sample clock can be exported to one of two destinations: the DDC CLK OUT pin on the DDC connector or the CLK OUT SMB jack connector.

- **DDC CLK OUT**—The Sample clock can be exported to the CLK OUT pin on the DDC connector. The exported Sample clock is generated at the same voltage levels specified for dynamic generation. For dynamic generation sessions, exporting the Sample clock to this connector allows for source synchronous clocking by routing the Sample clock through the same cable and propagation delay characteristics as the generated data.
- **CLK OUT**—The Sample clock can be exported to the CLK OUT SMB connector. The exported Sample clock is generated at the same voltage levels specified for dynamic generation.

You can export the Sample clock to either the DDC connector or the CLK OUT SMB jack connector, but *not* at the same time.

#### **Reference Clock**

If you configure a Reference clock for the PLL on the NI 655*x*, you can export the reference clock to the CLK OUT SMB jack connector. The exported reference clock operates at 3.3 V logic, independent of the rest of your programmed channels, DIO<0..19>.

#### **Onboard Reference Clock**

If you are using an NI PCI-655*x*, you can export the 10 MHz onboard reference clock to RTSI 7 on the RTSI trigger bus. You can then use a RTSI cable to connect this signal to other PCI devices.

# **Channel Interface**

The NI 655*x* has 20 channels. Each channel is independently configurable for generation, acquisition, or simultaneous generation and acquisition operations. Generation and acquisition voltage levels on the NI 655*x* are independently programmable—you can select one set of voltage levels for all generation drivers and a different set of voltage levels for all acquisition comparators.

The following topics provide more information about the channel interface:

- Front Panel and Connector Pinout
- LED Indicators

#### **Front Panel and Connector Pinout**

The NI 655*x* front panel, shown below, has three SMB jack connectors and one 68-pin Digital Data & Control (DDC) VHDCI connector. The SMB jack connectors are described in the <u>SMB Jack Connector Names and</u> <u>Descriptions</u> table. The DDC connector signals are described in the <u>DDC</u> <u>Connector Names and Descriptions</u> table.



SMB	Jack	Connector	Names	and	<b>Descriptions</b>
-----	------	-----------	-------	-----	---------------------

Connector	Signal Name	Signal Type	Signal Description
CLK IN	Reference/Clock Input	Control	External reference clock used for the PLL or for the external Sample clock used for pattern generation and/or acquisition.
PFI 0	Programmable Function Interface (PFI) 0	Control	Input terminal to the NI 655 <i>x</i> for external triggers or the output terminal from the NI 655 <i>x</i> for events.
CLK OUT	Reference/Clock Output	Control	Terminal for the exported PLL Reference clock or the exported Sample clock.

DDC Connector Names and Descriptions					
Pins	Signal Name	Signal Type	Signal Description		
33	DDC CLK OUT	Control	Terminal for the exported Sample clock.		
67	STROBE	Control	External Sample clock source which can be used for dynamic acquisition.		
13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65	DIO <019>	Data	Bidirectional digital I/O data channels 0 through 19.		
26, 30, 64	PFI<13>	Control	Input terminals to the NI 655 <i>x</i> for external triggers or output terminals from the NI 655 <i>x</i> for events.		
2, 4, 6, 10, 12, 14, 16, 18, 20, 22, 24, 28, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 54, 56, 58, 62, 66	GND	Ground	Ground reference for signals.		
1, 3, 5, 7, 8, 9, 11, 35, 37, 39, 41, 43, 45, 52, 60	RESERVED	N/A	These terminals are reserved for future use. Do not connect to these pins.		

# LED Indicators (PXI Only)

The NI PXI-655*x* has two LED indicators on the front panel, labeled ACCESS and ACTIVE. The following tables describe what each LED color indicates.

#### ACTIVE LED

Color	Indications
Off	Device not armed, not triggered, or experiencing an error.
Amber	Device armed and awaiting Start trigger. If performing a dynamic acquisition operation, the device may be acquiring pretrigger samples.
Green	Device received Start trigger.
Red	Error condition.

#### ACCESS LED

Color	Indications
Off	Device not ready.
Amber	Device being accessed by software.
Green	Device ready to be programmed.
Red	Running the <u>niHSDIO Self Test</u> VI or calling the <u>niHSDIO_self_test</u> function produced a failure.

# Acquisition

Expand this book for more information about <u>static</u> and <u>dynamic</u> <u>acquisition</u> using the NI 655x.

# **Static Acquisition**

Static acquisition is a software-timed (nonclocked) operation. When performing static acquisition operations, the NI 655*x* returns the current logic state of the configured data channels each time a static read is requested.

You can perform static acquisition operations at any time on channels configured for static acquisition. You can also use static acquisition to read back the current value on channels configured for static generation or dynamic generation at any time.

For more information about performing static acquisition in NI-HSDIO, refer to <u>Reading and Writing Static Data</u>.

# **Dynamic Acquisition**

The NI 655x provides flexible acquisition capabilities for up to 20-bit wide patterns with programmable voltage thresholds using either an internal or external clock source. The NI 655x dual comparator architecture allows for <u>data interpretation</u> based on the acquired <u>voltage level</u> relationship to both the Acquisition Voltage Low and Acquisition Voltage High thresholds. External triggers can control the acquisition operation, and the Pattern Acquisition Engine can route those control signals to be shared with other devices.

Refer to <u>Dynamic Acquisition Clock Sources</u> for information about available clocks for a dynamic acquisition operation.

For information about defining acquisition resources in NI-HSDIO, refer to <u>Acquisition Configuration Functions</u> to learn which VIs and C functions are available for your application.

# **Dynamic Acquisition Clock Sources**

Dynamic acquisition is a clocked operation driven by one of several clocking resources. Refer to the main <u>Clocking</u> diagram for this device to see a block diagram for these clock resources.

# **Dynamic Acquisition Timing Diagrams**

The following diagram illustrates the <u>data positions</u> available when acquiring waveforms with the NI 655x. For simplicity, the delayed data is shown delayed by 25% of the clock period; however, this value can vary between 0% and 100%.



#### Using the Sample Clock as the Acquisition Clock

 $t_{\text{DDCSC}}$  : Time Delay from DDC Connector to Internal Sample Clock

 $0 \leq \delta_A \leq 1$  : Pattern Acquisition Data Delay (fraction of  $t_{\text{P}})$ 

 $t_{P} = \frac{1}{f}$  = Period of Sample Clock

t_{SUSC} = Set-Up Time to Sample Clock

 $t_{\text{HSC}}$  = Hold Time to Sample Clock

#### Using STROBE as the Acquisition Clock



t_{SUS} = Set-up Time to STROBE

 $t_{HS}$  = Hold Time from STROBE

 $0 \le \delta_A \le 1$ : Pattern Acquisition Data Delay (percentage of t_P)

 $t_P = \frac{1}{f}$  = Period of Sample Clock

# **Dynamic Acquisition Triggers and Events**

The following table describes the relationship of triggers and events in a dynamic acquisition operation. The sequence of triggers and events is shown in the <u>Dynamic Acquisition State Diagram</u>.

Triggers are received synchronously by the <u>Pattern Acquisition Engine</u>. The Ready for Start event and all re-exported triggers are asynchronously generated by the Pattern Acquisition Engine.

Trigger/Event	Received From	Exported To
Start Trigger	The Start trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<019>. The Start trigger can also be sent by software.	The Start trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Reference Trigger	The Reference trigger can be received on a rising edge on PFI <03> RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<019>. The Reference trigger can also be sent by software.	The Reference trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Advance Trigger	The Advance trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not- matching a pattern received on DIO<019>. The Start trigger can also be sent by software.	The Advance trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Pause Trigger	The Pause trigger can be received from PFI<03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-	

	matching a pattern received on DIO<019>.	
Ready for Start Event		The Ready For Start event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Ready for Advance Event		The Ready For Advance event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
End of Record Event		The End of Record event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

# Generation

Expand this book for more information about <u>static</u> and <u>dynamic</u> <u>generation</u> using the NI 655x.

## **Static Generation**

Static generation is a software timed (nonclocked) operation and can be applied to any number of the available DIO<0..19> channels. When performing a static generation operation, the NI 655*x* sets the current state of the configured data channels to the requested logic state—high level or low level. NI 655*x* devices support the <u>open collector drive type</u>, which means that they can set the configured data channels to a high-impedance (Z) state when the channels are configured to drive the Generation Voltage High Level.

Refer to <u>Voltage Ranges and Settings</u> for more information about voltage levels for each logic state.

Static generation can be done on any number of channels, provided that those channels are not configured for a dynamic operation. To statically set the state of dynamic generation channels, refer to <u>Initial and Idle</u> <u>States</u>.

For more information about performing static generation with NI-HSDIO, refer to <u>Reading and Writing Static Data</u>.

# **Dynamic Generation**

Dynamic generation is a clocked operation where binary data is sent from the NI digital waveform generator/analyzer to the DUT across multiple digital channels.

The NI 655*x* can generate complex digital patterns up to 20 bits wide at a variety of <u>voltage levels</u> synchronous to any of several clock sources. The data can be generated as simple waveforms or based on complex scripts. External triggers can control the data generation, and the <u>Pattern</u> <u>Generation Engine</u> can export several types of <u>events</u> to indicate the progress of the generation.

In addition, the NI 655x allows for precise <u>subperiod timing control</u> between the generated data and the exported Sample clock. All these options make the NI 655x a versatile digital pattern generator.

NI 655*x* devices support the <u>open collector drive type</u>, which means that they can set the configured data channels to a high-impedance (Z) state when the channels are configured to drive the Generation Voltage High Level.

# **Dynamic Generation Clock Sources**

Dynamic generation is a clocked operation. The dynamic generation operation is clocked by one of several clocking resources. Your application needs may determine which source you should use. Refer to the main <u>Clocking</u> diagram to see a block diagram for these clock resources.

The following information discusses additional considerations for using these clocking resources for dynamic generation:

On Board Clock

The default clock source for dynamic generation sessions is the On Board Clock. This clock can be locked to a reference clock to synchronize operations across multiple devices or can be used without a reference clock when multidevice synchronization is not required. The On Board Clock is derived from integer divisors of the 200 MHz VCXO. Refer to the <u>NI 655x specifications</u> for information about the possible On Board Clock frequencies.

You can configure the On Board Clock source in the following ways:

- Free-running, nonphase-locked—In this mode, the VCXO is used at its fundamental frequency, allowing for a stable and accurate 200 MHz clock. This configuration is the default setting, and it is most useful when only one NI 655x is in the system or when multidevice synchronization is not required.
- Phase-Locked—The On Board Clock source can be locked to a reference clock using the PLL circuit to ensure that Sample clock alignment across devices is achieved. In this operation mode, the PLL circuit must be provided a precision source to which it can lock. The On Board Clock source can be locked to one of the following reference clock sources:
  - PXI_CLK10 (NI PXI-6551/6552 only)/RTSI 7 (NI PCI-6551/6552)—The PXI standard defines a precision 10 MHz reference (PXI_CLK10) to be distributed across the backplane to each device in the PXI chassis. If you are using PXI, this 10 MHz backplane clock is used as the reference for the PLL in this mode

of PLL operation. If you are using PCI, drive the 10 MHz On Board Reference Clock onto RTSI 7, and configure RTSI 7 as the reference clock source.

- **CLK IN**—If you want to provide your own reference, you can provide an external source on the CLK IN SMB connector to which the PLL can lock. Using an external reference allows you to easily synchronize clocks across instruments within and outside of the system. Refer to the <u>NI 655x specifications</u> for information about the possible reference clock frequencies.
- External Source (CLK IN)

Alternatively, your dynamic generation operation can be driven from an external Sample clock source. Using an external frequency generator, you can drive dynamic generation operations at any frequency within the NI 655*x* specifications. Frequency limitations and acquisition levels are listed in <u>NI 655*x*</u> <u>specifications</u>.

• PXI_STAR (NI PXI-6551/6552 only)

The PXI specification allocates resources for high-speed precision clock and trigger routing across the PXI backplane. The NI PXI-655*x* can use this resource to clock your dynamic generation task. An external source can drive this resource at any suitable frequency, allowing the NI PXI-655*x* to operate at noninteger divisors of 200 MHz, similar to how it operates using an external clock source (CLK IN).

For a summary of these and other clock sources, refer to <u>Clock Sources</u> <u>Summary</u>.

# **Dynamic Generation Timing Diagrams**

The following figure illustrates the data and clock positions available when generating waveforms with the NI 655x. For simplicity, the data is shown delayed by 25% of the clock period; however, this value can vary between 0% and 100%.

Note Data generation on the rising clock edge, falling clock edge, or delayed, is per channel selectable. However, in the delayed case, the delay value is constant across all delayed channels.



 $t_{\text{SCDDC}}$  : Time D elay from Sample Clock (Internal) to DDC Connector

 $0 \leq \delta_C \leq 1$  : Exported Sample Clock Delay (Fraction of  $t_p)$ 

 $0 \leq \delta_G \leq 1$  : Pattern Generation Data Delay (Fraction of  $t_p)$ 

 $t_p = \frac{1}{f} = P \text{ eriod of Sam ple Clock}$ 

 $t_{\rm CO}$  = Exported Sample Clock Offset; 0 or 2.5 ns, Software-Selectable

For more information about using NI-HSDIO to adjust the data position, refer to <u>Configuring Data Position</u>.

### **Dynamic Generation Triggers and Events**

The following table describes the relationship of triggers and events in a dynamic acquisition operation. The sequence of triggers and events is shown in the <u>Dynamic Generation State Diagram</u>.

Start, Pause, and Script triggers are received asynchronously by the <u>Pattern Generation Engine</u>. Cable propagation delays and pipeline delays can cause the Pattern Generation Engine to take multiple clock cycles to respond to a trigger. Refer to <u>NI 655x specifications</u> for more information. The <u>Ready For Start</u>, <u>Data Active</u>, and <u>Marker 0 events</u> are generated by the Pattern Generation Engine synchronous to the data generation.

The following table provides information about how these triggers and events can arrive at and be exported from the NI 655x.

Trigger/Event	Received From	Exported To
<u>Start Trigger</u>	The Start trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices). The Start trigger can also be sent by software.	The Start trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Pause Trigger	<ul> <li>The Pause trigger is level-based and can be received on PFI&lt;03&gt;, RTSI&lt;07&gt; (PCI devices), or PXI_TRIG&lt;07&gt; (PXI devices). The Pause trigger can also be sent by software.</li> <li>Note When the Pause trigger is asserted, the NI 655x Pattern Generation Engine may take several clock cycles to respond because of cable propagation delay and the pipelining in the system. Refer to the NI 655x specifications for more</li> </ul>	The Pause trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

	information.	
<u>Script Trigger</u> <03>	Four Script triggers can be edge- or level-based and can be received on PFI <03>, RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices). The Script trigger can also be sent by software.	The Script trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
<u>Ready for</u> <u>Start Event</u>		The Ready For Start event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
<u>Data Active</u> <u>Event</u>		The Data Active event can be exported to PFI <03>.
<u>Marker Event</u> <03>		A Marker event can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
### **Per Cycle Tristate**

The NI 655*x* is capable of <u>tristating channels on a per pin, per cycle basis</u> while <u>generating waveforms</u>. In addition to the standard digital states of 0 and 1, the NI 655*x* digital waveforms also support the Z state, so during each clock cycle (sample) a channel can drive low, drive high, or go to a high-impedance state.

To use this feature, write your waveforms using the digital waveform data type (WDT). For more information about writing digital data using the digital waveform data type, refer to <u>Digital Waveform Data</u> <u>Representation</u>.

The NI 655*x* stores information about which channels to tristate in a section of onboard memory that is separate from the waveform data. The NI 655*x* implementation of per pin, per cycle tristate is optimized for memory usage so that each distinct combination of channels to tristate is stored only once in memory. Thus, if a waveform tristates the same combination of channels at various samples throughout the waveform, only one memory location of the tristate memory is used. The tristate memory can store up to 4,095 distinct combinations of channels to tristate.



**Note** If the input impedance for a channel is configured for  $50 \Omega$ , the channel has a  $50 \Omega$  impedance connection to ground. Thus, when the line is set to tristate, this connection pulls the channel to ground instead of placing the channel in a high-impedance state.

# Hardware Comparison

Hardware comparison allows the NI 655x to verify that a DUT returns the correct response data under different use cases and stimulus data.

Note Real-time hardware comparison is supported with *only* the NI 655*x* devices. Other NI digital waveform generator/analyzers, such as the NI 654*x*/656*x* support acquiring the data into PC memory for analysis.

There are two primary methods for comparing acquired response data with expected data. With the first method, the NI 655*x* captures the actual response data into PC memory and uses software to post-process the results. The software uses only the two basic logic states, 0 and 1, to configure the testers stimulus data. With the second method, you can preload the NI 655*x* with both stimulus and expected response data and make real-time comparisons as data is acquired. Whenever a waveform contains a comparison logic state (H or L), then the acquired response data is compared to the expected response. You can choose whether this real-time hardware comparison operation drives and compares data (Stimulus and Expected Response Mode) or whether it only acquires and compares (Expected Response Only mode).

Data comparison logic in the onboard FPGA connects the generation and acquisition circuitry. The data decoder receives data from onboard memory and enables/disables the driver based on the logic state of each sample. The decoder transfers the expected response to the acquisition engine. A FIFO allows the alignment of the actual response with the expected response. If an error is detected during the comparison, then information on the fault is stored separately from the acquired data so the application software can retrieve both types of information for further analysis.

The device stores the following information for each fault detected:

- Sample number of the fault
- Channel(s) at fault
- Total number of repeated errors (useful if the <u>Filter Repeated</u> <u>Sample Errors</u> property or the <u>NIHSDIO ATTR HWC FILTER REPEATED SAMPLE ERRORS</u> attribute is enabled)

#### **Related Topics:**

- Digital Logic States
- <u>Comparing Response Data with Expected Data</u>

### Hardware Comparison Triggers and Events

Hardware comparison supports the triggers and events used in NI 655x <u>acquisition</u> and <u>generation</u> sessions.

In addition, the <u>Sample Error event</u> is also used in hardware comparison to indicate when the device detects a sample that is in error.

#### **Related Topics:**

- Dynamic Acquisition Triggers and Events
- Dynamic Generation Triggers and Events
- Eliminating Round Trip Delay

## NI 656*x*

The NI 656*x* is a 16-channel digital I/O device that you can use as a PC/peripheral device interface, pattern generator, pattern analyzer, or stimulus-response tester. The NI 6561 has a maximum Sample clock frequency of 100 MHz, and the NI 6562 has a maximum Sample clock frequency of 200 MHz. Using the data rate multiplier of these devices, you can generate and acquire data at 200 Mb/s or 400 Mb/s, respectively.

The NI 656*x* also provides the following features:

- Sophisticated timing engine to maintain and measure the timing parameters of a DUT
- <u>LVDS</u> voltage levels for high-speed interfacing to devices
- Data channels with per channel software direction control
- Selectable single-ended LVCMOS or differential LVDS configuration for PFI channel 3 on the <u>DDC connector</u>
- <u>Double-data rate (DDR) or single-data rate (SDR)</u> data rate multiplier options
- Deep onboard memory with triggering and pattern sequencing capabilities
- Ability to use <u>NI-TClk</u> to synchronize multiple devices

You can use the internal On Board Clock or import an external clock through the front panel. You can also shift the generated data, acquired data, and exported Sample clock relative to the onboard clock for clock frequencies above 25 MHz, which is critical when accounting for propagation delays and setup-and-hold times in the DUT.

Expand this book for more information about NI 656*x* hardware-related topics.

### **Hardware Architecture**

Expand this book for more information about the NI 656*x* hardware architecture.

## **Block Diagram**

The following figure is a block diagram illustrating the main functional units and data flow of the NI 656*x*. The text that follows the figure describes the basic elements of the diagram and provides links to sections with more detailed information about some of the blocks.





The <u>Clocking</u> module selects and distributes the clocks for the dynamic generation and dynamic acquisition operations.

For dynamic generation operations, the user-supplied data is loaded from the host computer memory into the onboard Generation Memory. The Pattern Generation Engine retrieves data from the Generation Memory and executes the script functionality while interacting with the associated Trigger and Event control module. The Pattern Generation Engine then sends the data to the Pattern Generation Timing and Control module, where the data is given the selected data position and data delay and is then sent to the <u>Channel Electronics</u> drivers. The Channel Electronics drivers generate the data at <u>LVDS voltage levels</u>.

For dynamic acquisition operations, signals arrive at the Channel Electronics circuitry, where the signal levels are interpreted appropriately for LVDS. The Timing and Control module samples the data using the selected clock, data position, and data delay values and passes the data to the Pattern Acquisition Engine. The Pattern Acquisition Engine and the Trigger and Event Control module recognize triggers and determine when the data should be stored into Acquisition Memory. The acquired data can then be fetched by the host computer.

## **Channel Electronics**

The channel electronics of NI 656*x* devices consist of LVDM buffers and the appropriate termination resistors. LVDM is an LVDS-compatible standard that allows for a 100  $\Omega$  parallel termination at the source and destination, which provides for the software-selectable direction control feature of the NI 656*x*. Each I/O channel is capable of simultaneously driving and receiving data.

The following figure provides a basic block diagram for the channel electronics. Refer to <u>NI 656x Block Diagram</u> for a picture of how the channel electronics circuitry fits into the overall block diagram.



### **Dynamic Generation**

For dynamic generation operations, the data signal appears at the buffer input after the Pattern Generation Timing and Control module gives the data the selected data position and data delay. The buffer converts the data signal to LVDS voltage levels before sending the data signal to the DDC connector on the NI 656*x* front panel.

The buffer can be set to high-impedance generation with the tristate control line. The tristate control cannot be set automatically by the <u>Initial</u> and <u>Idle States</u>. Set tristate programmatically with the <u>niHSDIO Tristate</u> <u>Channels</u> VI or <u>niHSDIO TristateChannels</u> function.

Protection for the channel electronics is critical for guarding against overvoltage situations and is built into the LVDM buffers. Refer to <u>Input</u> <u>Protection</u> for more information about this portion of the channel electronics.



**Note** LVDM buffers drive LVDS logic levels across 100  $\Omega$  source and 100  $\Omega$  destination termination loads (50  $\Omega$  total DC load).

#### **Dynamic Acquisition**

Patterns acquired by the NI 656*x* are received using a differential receiver. Refer to the <u>NI 656*x* Specifications</u> for the input voltage thresholds and ranges for LVDS.

The output of the receiver is sampled by the Pattern Acquisition Timing and Control module before being sent to the Pattern Acquisition Engine for storage into Acquisition Memory.

The input impedance is differential 100  $\Omega$ .

## **Voltage Ranges and Settings**

The NI 656*x* uses the following three voltage <u>logic families</u> on various pins:

- <u>LVDS</u>—Data channels, PFI<1..3>, clock inputs/outputs
- <u>LVPECL</u>—Exported Sample clock
- <u>3.3V Logic/TTL/CMOS</u>—PFI 0, PFI 3

All data channels on NI 656*x* devices are LVDS compliant. The following table describes which terminal configurations are supported by the PFI channels on the device.

PFI Channel	Location	Terminal Configuration
0	NI 656 <i>x</i> front panel	TTL/CMOS only
1, 2	DDC connector	LVDS only
3	DDC connector	LVDS or TTL/CMOS, software- selectable

Triggers and events must be individually <u>configured to use LVDS or</u> <u>single-ended terminal configurations</u> using NI-HSDIO.

When the Sample clock is exported to the DDC connector, both an LVDS and an LVPECL version of the clock signal are exported.

You do not need to configure voltage levels to use the NI 656*x*. Using the NI-HSDIO <u>Configure Voltage</u> functions or VIs with the NI 656*x* returns an error.

For more information about voltage level ranges and resolutions, refer to <u>NI 656x specifications</u>.

# **Logic Families**

The NI 656*x* PFI 3 channel can be independently set to one of two modes: LVDS or single-ended. The actual <u>voltage levels</u> defined by these logic families are defined in the <u>NI 656*x* specifications</u>. The <u>PFI terminal</u> <u>configuration</u> is software-selectable using NI-HSDIO.

Refer to <u>Voltage Ranges and Settings</u> (NI 656*x*) for more information about the logic families supported by the other PFI channels.

#### **Related Topics:**

- Logic Families Overview
- Differential Voltage Levels
- Voltage Ranges and Settings (NI 656x)

## Input Impedance

The input impedance of the NI 656*x* is 100  $\Omega$  differential. <u>Selectable input</u> <u>impedance</u> is only available with the <u>NI 655*x*</u> products.

Refer to <u>Configuring Input Impedance</u> for more information on configuring this property.

### Source Impedance

The NI 656*x* LVDS data and event generation channels have a 100  $\Omega$  differential source impedance. For generation operations, terminate the transmission line with a 100  $\Omega$  parallel resistance.

Because the NI 656x interface cable (NI SHB12X-B12X) is a 100  $\Omega$  differential transmission line, when you use LVDS, you can build matched impedance systems with a 100  $\Omega$  differential termination as the load. LVDS specifications of the NI 656x are expected across 100  $\Omega$  source in parallel with 100  $\Omega$  destination termination (50  $\Omega$  DC load).

## **Data Rate Multiplier**

The NI 656*x* can be configured for single data rate (SDR) or double data rate (DDR) operation by setting the data rate multiplier for your acquisition and/or generation session in NI-HSDIO. The data rate multiplier can be independently configured for the generation and acquisition sessions.

### Single Data Rate (SDR)

When the data rate multiplier is configured for SDR operation, generation and acquisition sessions can run on all <u>16 channels</u> of the NI 656x. Direction control is software-specified.

### Double Data Rate (DDR)

When the data rate multiplier is configured for DDR operation, the digital waveform generator/analyzer trades channel count for data rate by generating or acquiring on half the number of channels but at twice the rate. This rate is achieved by generating data on both the rising and falling edge of the Sample clock. For generation sessions, the device generates data on the lower eight channels of the NI 656*x* (DIO <0..7>), and for acquisition sessions the device acquires data on the upper eight channels (DIO <8..15>).

DDR mode has some important implications for some NI 656*x* functionality.

- **Memory Usage**—<u>Memory</u> size in samples is effectively doubled since the data width and channel count are halved.
- **Marker Positions**—<u>Marker</u> positions have a quantization twice that of SDR mode. Refer to the <u>NI 656x specifications</u> for more information about quantization.
- Waveform Sizes—The size of the <u>waveforms</u> you save to the <u>onboard memory</u> have a quantization twice that of SDR mode. Refer to the <u>NI 656x specifications</u> for more information about quantization.
- Initial/Idle States—If a channel's Idle state is configured for "hold last value," the last value held is the last DDR data sample.
- Data Width—Data width is a function of your data rate multiplier. Since data width refers to how large your sample is in bytes, using DDR mode effectively halves your allowable data width. For NI 656x devices, SDR operation would have a data width of 2 bytes per sample while DDR mode would have a data width of 1 byte per sample.

#### **Related Topics:**

- Data Rate Multiplier Overview
- Single Data Rate (SDR)
- Double Data Rate (DDR)
- Data Position with DDR
- Generation Considerations for DDR
- <u>Acquisition Considerations for DDR</u>

### **Input Protection**

PFI 0 and 3 are protected using diode clamps connected to positive and negative voltage supplies. PFI 0 and 3 are clamped to 5 V.

The following figure illustrates this circuit.

These diodes act as open circuits unless the I/O voltage levels go above  $V_{p+}$  or below  $V_{p-}$ . When the I/O voltage exceeds  $V_{p+}/V_{p-}$ , the diodes become short circuits, clamping the input voltage to  $V_{p+}/V_{p-}$ . Therefore, these diodes prevent input voltages from going more than a diode drop, or approximately 0.5 V, beyond the positive or negative protection rails. The following diagram demonstrates the effect of the clamp.



DIO <0..15>, PFI <1..2>, DDC CLK OUT LVDS, DDC CLK OUT LVPECL, and STROBE are protected using robust pin electronics. Each channel is independently buffered using a robust pin electronic transceiver. These devices can protect against instantaneous shock and overvoltage cases.

In addition to the I/O circuit shown in the <u>Channel Electronics</u> diagram, data channels have a weak pull-up resistor (300 k $\Omega$ ), internal to the I/O buffer, to 3.3 V. This internal pull-up resistor is a fail-safe mechanism intended to set a known state when the receiver circuit is not being driven (tristate).

In a tristate condition, there is an equivalent RC circuit created from 3.3 V to ground. This RC equivalent is the interaction between these weak pullup resistors and the capacitive transmission media (a cable or PCB, for example). As such, when the output terminals were previously tristated, some finite amount of time passes between the time when output terminals are driven to when the cable voltage reaches a steady common-mode value. This time varies with cable length and can take as many as several Sample clock periods.



**Note** For this reason the NI 656x device does *not* support highimpedance <u>initial and Idle states</u>. Use the <u>niHSDIO Tristate</u> <u>Channels</u> VI or the <u>niHSDIO_TristateChannels</u> function.

Refer to the <u>NI 656x specifications</u> for details on input protection.

# **Signal Routing**

The NI 656*x* is capable of sending and receiving signals through the front panel and through the PXI trigger bus.

The <u>front panel connectors</u> provide connectivity for the bidirectional DIO channels as well as for control lines for sending and receiving clocks, triggers, and events.

# Clocking

The following figure shows how the <u>clock sources</u> are routed to produce the NI 656x <u>clock signals</u>.



### **Clock Sources Summary**

The following tables describe the clock sources available for the NI 656*x*. These clock sources are shown in the <u>Clocking</u> diagram. For a more general description of these clocks, refer to <u>Clocks for Digital Waveform</u> <u>Generator/Analyzers</u>.

Clock Source	Used In	Location	Description
On Board Clock	Acquisition, Generation	Internal	The NI 656 <i>x</i> provides a single high- precision 200 MHz voltage- controlled crystal oscillator (VCXO) clock source. The NI 656 <i>x</i> can generate any clock frequency of 200 MHz/ <i>n</i> , where <i>n</i> is any integer from 1 to 4,194,304 for the NI 6562, and 2 to 4,194,304 for the NI 6561. For example, for the NI 6562, the On Board Clock can run at 200 MHz, 100 MHz, 66.67 MHz, 50 MHz, 40 MHz, 33.33 MHz, 28.57 MHz, 25 MHz, 22.22 MHz, and so on. The onboard PLL allows the On Board Clock to be phase-locked to the Reference clock, if one is provided.
CLK IN	Acquisition, Generation	Front panel SMB jack connector	The CLK IN SMB jack is intended for use as an external frequency input channel, allowing you to provide an alternate frequency as the Sample clock rate. The CLK IN signal can be any sine or square wave signal that meets the specifications provided in the <u>NI 656x specifications</u> . The CLK IN signal must be free running.
PXI_STAR (NI PXI- 6561/6562 only)	Acquisition, Generation	Backplane	The PXI_STAR connector can be used as an external frequency input channel, allowing you to provide an alternate frequency as the Sample clock rate. The PXI_STAR signal specifications are provided in the <u>NI 656x specifications</u> . The PXI_STAR signal must be free

	running.	

Reference Clock			
Clock Source	Used In	Location	Description
NONE	Acquisition, Generation	Internal	When no reference clock source is selected, the PLL is not locked and the On Board Clock has no known phase relationship to any other clocks in the system.
CLK IN	Acquisition, Generation	Front panel SMB jack connector	The CLK IN SMB jack can be used to provide an external Reference clock for the PLL. The CLK IN signal can be any sine or square wave signal that meets the specifications provided in the <u>NI 656x</u> <u>specifications</u> . The CLK IN signal must be free running.
PXI_CLK10	Acquisition, Generation	PXI trigger bus	The PXI Clock 10 line exists on the PXI backplane and provides a 10 MHz reference clock to all slots in the chassis. The PLL can be configured to lock to this signal.
RTSI 7 (NI PCI- 6561/6562 only)	Acquisition, Generation	RTSI trigger bus	The Onboard Reference Clock can be routed to RTSI 7 to provide a 10 MHz reference clock signal to the NI 654x and other devices that share the RTSI bus. The PLL can be configured to lock to this signal.

STROBE				
Clock Source	Used In	Location	Description	
STROBE	Acquisition	DDC connector	STROBE is intended for use as the Sample clock for dynamic acquisition sessions when source-synchronous transfers are desired (that is, when the data and clock travel together through the cable from the DUT to the NI 656 <i>x</i> ). The STROBE signal must be a free- running LVDS signal.	

# **Exporting a Clock**

The NI 656*x* provides several resources for exporting clocks. The <u>Clocking</u> block diagram shows how the NI 656*x* exports these clocks.



**Note** As shown in the <u>clocking block diagram</u>, it is possible to export both the <u>Reference clock</u> and the <u>Sample clock</u> at the same time if you route the Reference clock to the <u>CLK OUT SMB jack</u> <u>connector</u> and the Sample clock to DDC CLK OUT channel on the <u>DDC connector</u>. You cannot route the Sample clock to both connectors.

For information about using NI-HSDIO to export clocks, refer to the <u>niHSDIO Export Signal</u> VI or the <u>niHSDIO_ExportSignal</u> function.

The following table summarizes the possible exported clock options.

Clock	Destination	Description
Sample Clock	DDC CLK OUT	DDC CLK OUT LVDS and DDC CLK OUT LVPECL on DDC connector
	CLK OUT	CLK OUT SMB jack connector
Reference Clock	CLK OUT	CLK OUT SMB jack connector
Onboard Reference Clock (NI PXI-656 <i>x</i> only)	RTSI 7	RTSI trigger bus channel 7

### Sample Clock

The Sample clock can be exported to one of two destinations: the DDC connector or the CLK OUT SMB jack connector.

• **DDC CLK OUT**—The Sample clock can be exported to the DDC CLK OUT on the DDC connector. The exported Sample clock is generated at the logic family voltage levels specified for dynamic generation. For dynamic generation sessions, exporting the Sample clock to this connector allows for source-synchronous clocking by routing the Sample clock through the same cable and propagation delay characteristics as the generated data.



• **CLK OUT**—The Sample clock can be exported to the CLK OUT SMB jack connector. Refer to the <u>NI 656x specifications</u> for more information about the voltage levels for the exported Sample clock.

You can export the Sample clock to either the DDC connector or the CLK OUT SMB jack connector, but *not* at the same time.

#### **Reference Clock**

If you configure a reference clock for the PLL on the NI 656*x*, you can export the reference clock to the CLK OUT SMB jack connector. Refer to the <u>NI 656*x* specifications</u> for more information about the voltage levels for the exported Reference clock.

#### **Onboard Reference Clock**

If you are using an NI PCI-656*x*, you can export the 10 MHz onboard reference clock to RTSI 7 on the RTSI trigger bus. You can then use a RTSI cable to connect this signal to other PCI devices.

## Valid Data Delay Ranges

At frequencies higher than 50 MHz, you can legally configure your data delay as any fractional value from 0 to 1 clock period.

At the 25 to 50 MHz frequency range, however, portions of the Sample clock period do not support the data delay. For frequencies between 25 and 50 MHz, you can legally configure data delay as any value from 0 to 1 Sample clock periods except:

$$\left[0.25 \pm \left(0.25 - \frac{5ns}{t_{\rho}}\right)\right]$$
 and  $\left[0.75 \pm \left(0.25 - \frac{5ns}{t_{\rho}}\right)\right]$ 

where  $t_p$  represents the period of the Sample clock.

The following figure compares the legal and illegal settings for delayed data position.



### **Channel Interface**

The NI 656*x* has 16 channels. Each channel is independently configurable for generation, acquisition, or simultaneous generation and acquisition operations. Generation and acquisition logic families on the NI 656*x* are automatically configured. Using the NI-HSDIO Configure Voltage functions or VIs with the NI 656*x* returns an error.

The following topics provide more information about the channel interface:

- Front Panel and Connector Pinout
- LED Indicators

### **Front Panel and Connector Pinout**

The NI 656*x* front panel, shown below, has three SMB jack connectors and one 73-pin Digital Data & Control (DDC) 12x Infiniband connector. The SMB jack connectors are described in the <u>SMB Jack Connector</u> <u>Names and Descriptions</u> table. The DDC connector signals are described in the <u>DDC Connector Names and Descriptions</u> table.





**Note** If you design a custom cabling solution with connector (779157-01) and cable (192744-01), the NI 656*x* pinout is reversed at the end connector. For example, the signal shown on pin 1 shown in the previous figure would map to pin 73 at the end connector.

Child Guok Connector Maines and Descriptions
----------------------------------------------

Connector	Signal Name	Signal Type	Signal Description
CLK IN	Reference/Clock Input	Control	External reference clock used for the PLL or for the external Sample clock used for pattern generation and/or acquisition.
PFI 0	Programmable Function Interface (PFI) 0	Control	Input terminal to the NI 656 <i>x</i> for external triggers or the output terminal from the NI 656 <i>x</i> for events.
CLK OUT	Reference/Clock Output	Control	Terminal for the exported PLL Reference clock or the exported Sample clock.
DDC Connector Names and Descriptions			
----------------------------------------------------------------------	---------------------------	----------------	--------------------------------------------------------------------------------------------------------------------------------------------
Pins	Signal Name	Signal Type	Signal Description
65	DDC CLK OUT LVDS	Control	Positive terminal for the LVDS exported Sample clock.
66	DDC CLK OUT LVDS*	Control	Complementary terminal for the LVDS exported Sample clock.
71	DDC CLK OUT LVPECL	Control	Positive terminal for the LVPECL exported Sample clock.
72	DDC CLK OUT LVPECL*	Control	Complementary terminal for the LVPECL exported Sample clock.
62	STROBE	Control	Positive external Sample clock source that can be used for dynamic acquisition.
63	STROBE*	Control	Complementary external Sample clock source that can be used for dynamic acquisition.
14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, 53, 56, 59	DIO <015>	Data	Positive bidirectional digital I/O data channels 0 through 15.
15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60	DIO <015>*	Data	Complementary bidirectional digital I/O data channels 0 through 15.
2, 5, 8	PFI<13>	Control	Positive input terminals to the NI 656 <i>x</i> for external triggers or output terminals from the NI 656 <i>x</i> for events.

3, 6, 9	PFI<13>*	Control	Complementary input terminals to the NI 656 <i>x</i> for external triggers or output terminals for the NI 656 <i>x</i> for events. Note Pin 9 is grounded when the PFI channel is configured for the single-ended terminal configuration.
1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58	GND	Ground	Ground reference for signals.
11, 12, 68, 69	RESERVED	N/A	These terminals are reserved for future use. Do not connect to these pins.

# LED Indicators (PXI Only)

The NI PXI-656*x* has two LED indicators on the front panel, labeled ACCESS and ACTIVE. The following tables describe what each LED color indicates.

#### ACTIVE LED

Color	Indications
Off	Device not armed, not triggered, or experiencing an error.
Amber	Device armed and awaiting Start trigger. If performing a dynamic acquisition operation, the device may be acquiring pretrigger samples.
Green	Device received Start trigger.
Red	Error condition.

#### ACCESS LED

Color	Indications
Off	Device not ready.
Amber	Device being accessed by software.
Green	Device ready to be programmed.
Red	Running the <u>niHSDIO Self Test</u> VI or calling the <u>niHSDIO_self_test</u> function produced a failure.

# Acquisition

Expand this book for more information about <u>static</u> and <u>dynamic</u> <u>acquisition</u> using the NI 656x.

## **Static Acquisition**

Static acquisition is a software-timed (nonclocked) operation. When performing static acquisition operations, the NI 656*x* returns the current logic state of the configured data channels each time a static read is requested.

You can perform static acquisition operations at any time on channels configured for static acquisition. You can also use static acquisition to read back the current value on channels configured for static generation or dynamic generation at any time.

For more information about performing static acquisition in NI-HSDIO, refer to <u>Reading and Writing Static Data</u>.

# **Dynamic Acquisition**

The NI 656*x* provides flexible acquisition capabilities for up to 16-bit wide patterns with <u>LVDS</u> voltage levels using either an internal or external clock source. External triggers can control the acquisition operation, and the Pattern Acquisition Engine can route those control signals to be shared with other devices.

Refer to <u>Dynamic Acquisition Clock Sources</u> for information about available clocks for a dynamic acquisition operation.

For information about defining acquisition resources in NI-HSDIO, refer to <u>Acquisition Configuration Functions</u> to learn which VIs and C functions are available for your application.

## **Dynamic Acquisition Clock Sources**

Dynamic acquisition is a clocked operation driven by one of several clocking resources. Refer to the main <u>Clocking</u> diagram for this device to see a block diagram for these clock resources.

## **Dynamic Acquisition Timing Diagrams**

The following diagram illustrates the <u>data positions</u> available when acquiring waveforms with the NI 656*x* in <u>SDR mode</u>. For simplicity, the delayed data is shown delayed by 0.25 clock periods; however, this value can vary between zero and one, with some <u>exceptions</u>.



#### Using the Sample Clock as the Acquisition Clock

 $t_{\text{DDCSC}}$  : Time Delay from DDC Connector to Internal Sample Clock

 $0 \leq \delta_A \leq 1$  : Pattern Acquisition Data Delay (fraction of  $t_{\text{P}})$ 

 $t_{P} = \frac{1}{f}$  = Period of Sample Clock

t_{SUSC} = Set-Up Time to Sample Clock

 $t_{\text{HSC}}$  = Hold Time to Sample Clock

#### Using STROBE as the Acquisition Clock



t_{SUS} = Set-up Time to STROBE

 $t_{HS}$  = Hold Time from STROBE

 $0 \le \delta_A \le 1$  : Pattern Acquisition Data Delay (fraction of tp)

 $t_{P} = \frac{1}{f} = Period of Sample Clock$ 

Note: At 25 MHz and higher, STR OBE duty cycle is corrected to 50% while maintaining rising edge placement.

# **Dynamic Acquisition Triggers and Events**

The following table describes the relationship of triggers and events in a dynamic acquisition operation. The sequence of triggers and events is shown in the <u>Dynamic Acquisition State Diagram</u>.

Triggers are received synchronously by the <u>Pattern Acquisition Engine</u>. The Ready for Start event and all re-exported triggers are asynchronously generated by the Pattern Acquisition Engine.

Trigger/Event	Received From	Exported To
Start Trigger	The Start trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<015>. The Start Trigger can also be sent by software.	The Start trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Reference Trigger	The Reference trigger can be received on a rising edge on PFI <03> RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<015>. The Reference trigger can also be sent by software.	The Reference trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Advance Trigger	The Advance trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching a pattern received on DIO<015>. The Advance trigger can also be sent by software.	The Advance trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Pause Trigger	The Pause trigger can be received from PFI<03>, RTSI<07> (PCI devices), PXI_TRIG<07> (PXI devices), or by matching/not-matching	

	a pattern received on DIO<015>.	
Ready for Start Event		The Ready For Start event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Ready for Advance Event		The Ready For Advance event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
End of Record Event		The End of Record event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

## Generation

Expand this book for more information about <u>static</u> and <u>dynamic</u> <u>generation</u> using the NI 656x.

## **Static Generation**

Static generation is a software timed (nonclocked) operation and can be applied to any number of the available DIO<0..15> channels. When performing a static generation operation, the NI 656*x* sets the current state of the configured data channels to the requested logic state.

Static generation can be done on any number of channels, provided that those channels are not configured for a dynamic operation. To statically set the state of dynamic generation channels, refer to <u>Initial and Idle</u> <u>States</u>.

For more information about performing static generation with NI-HSDIO, refer to <u>Reading and Writing Static Data</u>.

# **Dynamic Generation**

Dynamic generation is a clocked operation where binary data is sent from the NI digital waveform generator/analyzer to the DUT across multiple digital channels.

The NI digital waveform generator/analyzer can generate complex digital patterns synchronous to any of several clock sources. The data can be generated as simple waveforms or based on complex <u>scripts</u>. External triggers can control the data generation, and the <u>Pattern Generation</u> <u>Engine</u> can export several types of <u>events</u> to indicate the progress of the generation.

In addition, the NI digital waveform generator/analyzer allows for precise <u>subperiod timing control</u> between the generated data and the exported Sample clock, making the device a versatile digital pattern generator.

## **Dynamic Generation Clock Sources**

Dynamic generation is a clocked operation. The dynamic generation operation is clocked by one of several clocking resources. Your application needs may determine which source you should use. Refer to the main <u>Clocking</u> diagram to see a block diagram for these clock resources.

The following information discusses additional considerations for using these clocking resources for dynamic generation:

On Board Clock

The default clock source for dynamic generation sessions is the On Board Clock. This clock can be locked to a reference clock to synchronize operations across multiple devices or can be used without a reference clock when multidevice synchronization is not required. The On Board Clock is derived from integer divisors of the 200 MHz VCXO. Refer to the <u>NI 656x specifications</u> for information about the possible On Board Clock frequencies.

You can configure the On Board Clock source in the following ways:

- Free-running, nonphase-locked—In this mode, the VCXO is used at its fundamental frequency, allowing for a stable and accurate 200 MHz clock. This configuration is the default setting, and it is most useful when multidevice synchronization is not required.
- Phase-Locked—The On Board Clock source can be locked to a reference clock using the PLL circuit to ensure that Sample clock alignment across devices is achieved. In this operation mode, the PLL circuit must be provided a precision source to which it can lock. The On Board Clock source can be locked to one of the following reference clock sources:
  - PXI_CLK10 (NI PXI-6561/6562 only)—The PXI standard defines a precision 10 MHz reference (PXI_CLK10) to be distributed across the backplane to each device in the PXI chassis. If you are using PXI, this 10 MHz backplane clock is used as the reference for the PLL in this mode of PLL operation.

- **CLK IN**—If you want to provide your own reference, you can provide an external source on the CLK IN SMB connector to which the PLL can lock. Using an external reference allows you to easily synchronize clocks across instruments within and outside of the system. Refer to the <u>NI 656x specifications</u> for information about the possible reference clock frequencies.
- External Source (CLK IN)

Alternatively, your dynamic generation operation can be driven from an external Sample clock source. Using an external frequency generator, you can drive dynamic generation operations at any frequency within the NI 656*x* specifications. Frequency limitations and acquisition levels are listed in <u>NI 656*x*</u> <u>specifications</u>.

• PXI_STAR (NI PXI-6561/6562 only)

The PXI specification allocates resources for high-speed precision clock and trigger routing across the PXI backplane. The NI PXI-656x can use this resource to clock your dynamic generation task. An external source can drive this resource at any suitable frequency, allowing the NI PXI-656x to operate at noninteger divisors of 200 MHz, similar to how it operates using an external clock source (CLK IN).

For a summary of these and other clock sources, refer to <u>Clock Sources</u> <u>Summary</u>.

## **Dynamic Generation Timing Diagrams**

The following figure illustrates the <u>data</u> and <u>clock</u> positions available when generating waveforms with the NI 656*x* in <u>SDR mode</u>. For simplicity, the data is shown delayed by 25% of the clock period; however, this value can vary between 0% and 100%, with some <u>exceptions</u>. Refer to the <u>NI 656*x* specifications</u> for more information about valid ranges.

Note Data generation on the rising or falling clock edge is per channel selectable. However, if you use the delayed position, *all* the data and PFI channels must be delayed, and the delay value must be constant across all channels.



 $t_{SCDDC}$  : Time Delay from Sample Clock (Internal) to DDC Connector Exported Sample Clock

 $0 \leq \delta_C \leq 1$  : Exported Sample Clock Delay (Fraction of  $t_p)$ 

 $0 \le \delta_G \le 1$  : Plattern Generation Data Delay (Fraction of  $t_p)$ 

 $t_p = \frac{1}{f}$  = Period of Sample Clock

t_{co} = Exported Sample Clock Offset

 $t_{CPD}$  = Exported Sample Clock to Selectable PFI Offset (LVDS)

t_{CPS} = Exported Sample Clock to Selectable PFI Offset (LVCMOS)

For more information about using NI-HSDIO to adjust the data position, refer to <u>Configuring Data Position</u>.

# Generation Provided Setup and Hold Times Timing Diagram



t_{PSU} = Minimum Provided Setup Time; SDR = Single Data Rate, DDR = Double Data Rate

 $t_{CO}$  = Exported Sample Clock Offset

NOTE: At 25 MHz and higher, STROBE duty cycle is corrected to 50%.

#### **Dynamic Generation Triggers and Events**

The following table describes the relationship of triggers and events in a dynamic acquisition operation. The sequence of triggers and events is shown in the <u>Dynamic Generation State Diagram</u>.

Start, Pause, and Script triggers are received asynchronously by the <u>Pattern Generation Engine</u>. Cable propagation delays and pipeline delays can cause the Pattern Generation Engine to take multiple clock cycles to respond to a trigger. Refer to <u>NI 656x specifications</u> for more information. The <u>Ready For Start</u>, <u>Data Active</u>, and <u>Marker 0 events</u> are generated by the Pattern Generation Engine synchronous to the data generation.

The following table provides information about how these triggers and events can arrive at and be exported from the NI 656x.

Trigger/Event	Received From	Exported To
<u>Start Trigger</u>	The Start trigger can be received from a rising or falling edge on PFI <03>, RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices). The Start trigger can also be sent by software.	The Start trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
Pause Trigger	<ul> <li>The Pause trigger is level-based and can be received on PFI&lt;03&gt;, RTSI&lt;07&gt; (PCI devices), or PXI_TRIG&lt;07&gt; (PXI devices). The Pause trigger can also be sent by software.</li> <li>Note When the Pause trigger is asserted, the NI 656x Pattern Generation Engine may take several clock cycles to respond because of cable propagation delay and the pipelining in the system. Refer to the NI 656x specifications for more</li> </ul>	The Pause trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

	information.	
<u>Script Trigger</u> <03>	Four Script triggers can be edge- or level-based and can be received on PFI <03>, RTSI<07> (PCI devices), or PXI_TRIG<07> (PXI devices). The Script trigger can also be sent by software.	The Script trigger can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
<u>Ready for</u> <u>Start Event</u>		The Ready For Start event can be exported to PFI<03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).
<u>Data Active</u> <u>Event</u>		The Data Active event can be exported to PFI <03>.
<u>Marker Event</u> <03>		A Marker event can be exported to PFI <03>, RTSI<06> (PCI devices), or PXI_TRIG<06> (PXI devices).

## Integration and System Considerations

Expand this book for more information about system considerations and hardware integration.

# **Terminating Your Module**

Expand this section to learn about terminated and unterminated configurations for acquisition and generation operations with the following device families:

- <u>NI 654</u>*x*
- <u>NI 655</u>*x*
- <u>NI 656</u>*x*

Refer to <u>Termination</u> for more information about the theory and recommendations for system termination.

# Terminating Your NI 654x

Expand this section to learn about the termination configurations for <u>acquisition</u> and <u>generation</u> with the NI 654x.

Refer to <u>Termination</u> for more information about the theory and recommendations for system termination.

## NI 654x Generation Termination

# Generation Termination: High-Impedance Load Configuration

A common configuration for your NI 654*x* is to connect the output terminals of your NI device directly to your device under test (DUT). Most digital logic inputs have an input impedance of at least 1–10 K $\Omega$ . Therefore, connecting the NI 654*x* output terminals directly to the input of your DUT creates a source-terminated configuration, because the generation channels of the NI 654*x* have a 50  $\Omega$  source impedance.

While this source termination configuration does not provide the absolute highest level of signal quality, there are many advantages to a source-terminated configuration. First, very good signal levels are possible if you ensure that you have the cleanest possible 50  $\Omega$  characteristic impedance transmission line. Second, this source-terminated configuration allows you to directly wire to your DUT without the need for additional termination resistors. Lastly, given that at DC there is effectively a voltage divider between the 50  $\Omega$  Z_s resistance and the high-impedance Z_t of your DUT, having an source-terminated system preserves the largest possible voltage swings at the DUT according to the following formula: V_{DUT} = V_{source}*(Z_t/(Z_s + Z_t)

So, for a DUT with an input impedance of  $1 \text{ k}\Omega$ , programming a generation voltage level of 3.3 V at the NI 654x source produces a 3.3 V * (1000/1050) = 3.14 V swing at the DUT.

A source-terminated configuration results in reflections in the transmission line. These reflections, however, are absorbed at the source and not re-reflected back to the load, thus preserving the signal integrity. Practically, the source impedance does not perfectly match the transmission line impedance; therefore, a small fraction of the reflected wave is re-reflected back toward the load. This second reflection creates small signal aberrations and a low level of inter-symbol interference.

For example, a 5% mismatch at the source results in a 2.5% re-reflection back at the load:

 $\Gamma_{\rm s} = (1.05 \ \text{--}1)/(1.05 \ \text{+-}1) \approx 2.5\%$ 

#### **Generation Termination: Terminated Load Configuration**

The source-terminated load configuration is easy to use with a terminated source, such as the NI 654*x*, and is recommended for all applications except the most demanding in regard to timing precision or signal integrity. For applications demanding the highest levels of signal quality and timing precision, NI recommends that you seriously consider following the recommendations of the terminated load configuration.

For applications requiring the highest levels of signal integrity and timing accuracy, NI strongly recommends carefully controlling the termination impedance at the end of the <u>transmission line</u>. To control the termination impedance, add a parallel termination resistor to ground as close as possible to the digital input pin of the device under test (DUT). In this configuration, the transmission line is terminated at both ends, which produces the highest possible signal integrity.

Ideally, the source impedance,  $Z_S$ , and the <u>characteristic impedance</u> of the transmission line,  $Z_0$ , should be kept as close as possible to 50  $\Omega$  as this will give you the best possible signal quality.

However, depending on your NI device, having all the lines terminated into 50  $\Omega$  may violate the maximum current specifications. Refer to the NI <u>654x specifications</u> for more information about the maximum current for your device to determine how many lines you can simultaneously terminate into 50  $\Omega$ .

While a  $\underline{Z}_t$  of 50  $\Omega$  is ideal, you can also use values as high as 300  $\Omega$  without significantly affecting signal quality. Using this higher resistance value enables you to increase the voltage swing across the DUT and decrease the drive current requirements on your NI 654*x*.

Given that at DC there is effectively a voltage divider between the 50  $\Omega$  Z_S resistance and the termination resistance, having a terminated load reduces the largest possible voltage swings at the DUT according to the following formula: V_{DUT} = V_{source}*(Z_t/(Z_s + Z_t)

So, for a 50  $\Omega$  termination, programming a generation voltage level of 3.3 V at the NI 654*x* source produces a 3.3 V x (50/100) = 1.65 V at the DUT. This reduced voltage swing at the DUT should be considered when you create your system.

Depending on voltage swing requirements, you have several generation termination options. The following table lists some of the options for the different voltage swings.



**Tip** Using a parallel termination resistor is only necessary for applications requiring the highest signal integrity.

Required Voltage Swing at DUT	Termination Options
3.3 V	Generation voltage family = 3.3 V Logic, no termination resistance
2.5 V	<ul> <li>Generation voltage family = 2.5 V Logic, no termination resistance</li> <li>Generation voltage family = 3.3 V Logic, Z_t = 156 Ω (I_{max} = 16 mA)</li> </ul>
1.8 V	<ul> <li>Generation voltage family = 1.8 V Logic, no termination resistance</li> <li>Generation voltage family = 3.3 V Logic, Z_t = 60 Ω (I_{max} = 30 mA)</li> <li>Generation voltage family = 2.5 V Logic, Z_t = 129 Ω (I_{max} = 14 mA)</li> </ul>

## **NI 654x Acquisition Termination**

#### **High-Impedance Load Acquisition Configuration**

For acquisition operations, the NI 654*x* can only be used in a highimpedance load configuration because the input impedance of the NI 654*x* is set to 10 k $\Omega$ . The high-impedance load configuration is easy to drive since it does not present a significant DC load to the source and preserves the signal amplitude. It is very important that you follow the recommendations in this section to achieve the highest level of AC signal quality.

The same transmission line considerations discussed for the generation case are applicable to acquired signals. The input impedance of 10 k $\Omega$  implies a reflection coefficient  $\Gamma_t$  of 0.99, or nearly full reflection.

With all high-impedance load transmission lines, it is essential that you take care to match the source impedance of the transmission line to the characteristic impedance of the transmission line. The source matching in this configuration is particularly important, since there are significant reflections from the high-impedance load (input of NI 654*x* in this case).

You should take care to ensure that your transmission line has a <u>characteristic impedance</u> of as close to 50  $\Omega$  as possible.

Therefore,  $Z_s$  (external source output impedance) should match  $Z_0 = 50 \Omega$  (the cable impedance) for 10 k $\Omega$  input configuration. To achieve this  $Z_s = 50 \Omega$ , you must determine the output impedance of your digital driver and add a series resistor as close as possible to the driver pin such that the output impedance of your buffer plus the value of the series resistor equal 50  $\Omega$ .

If you require 50  $\Omega$  termination for your acquisition application, consider using one of the <u>NI 655x</u> products.

#### NI 654x Termination Summary

The following table provides a high-level summary of the termination considerations for your NI 654x. Click a title for more information about that type of termination.



because of the mismatch on both ends. The recommended setup is to increase $Z_s$ to 50 $\Omega$ by adding series resistance at the source.	
<ul> <li>Z_s &gt; 50 Ω</li> <li>This configuration is not recommended because of the mismatch on both ends.</li> <li>Note You must reduce the operating frequency significantly to allow the signal reflections to settle down. You must reduce the frequency proportional to cable length and impedance mismatch.</li> </ul>	

# **Terminating Your NI 655x**

Expand this section to learn about the termination configurations for <u>acquisition</u> and <u>generation</u> with the NI 655x.

Refer to <u>Termination</u> for more information about the theory and recommendations for system termination.

# **NI 655x Generation Termination**

#### **Generation Termination: Unterminated Load Configuration**

A common configuration for your NI 655*x* is to connect the output terminals of your NI device directly to your device under test (DUT). Most digital logic inputs have an input impedance of 1–10 K $\Omega$ . Since your NI 655*x* was designed to be used in a 50  $\Omega$  environment, connecting the NI device output terminals directly to the input of your DUT effectively creates an unterminated load configuration.

While this unterminated configuration does not provide the absolute highest level of signal quality, there are many advantages to an unterminated configuration. First, very good signal levels are possible if you ensure that you have the cleanest possible 50  $\Omega$  <u>characteristic</u> impedance transmission line. Second, this unterminated configuration allows you to directly wire to your DUT without the need for additional termination resistors. Lastly, given that at DC there is effectively a voltage divider between the 50  $\Omega$  Z_s resistance and the high-impedance Z_t of your DUT, having an unterminated load preserves the largest possible voltage swings at the DUT according to the following formula: V_t = V_s* (Z_t/(Z_s + Z_t)

So, for a DUT with an input impedance of  $1 \text{ k}\Omega$ , programming a generation voltage level of 3.3 V at the NI 655x source produces a 3.3 V * (1000/1050) = 3.14 V swing.

The unterminated load generates reflections in the transmission line. The load reflections, however, are absorbed at the source and not re-reflected back to the load, thus preserving the signal integrity. Practically, the source impedance does not perfectly match the transmission line impedance; therefore, a small fraction of the reflected wave is re-reflected back toward the load. This second reflection creates small signal aberrations and a low level of inter-symbol interference.

For example, a 5% mismatch at the source results in a 2.5% re-reflection back at the load:

 $\Gamma_{\rm s} = (1.05 - 1)/(1.05 + 1) \approx 2.5\%$ 

#### **Generation Termination: Terminated Load Configuration**

The unterminated load configuration is easy to use with a terminated source, such as the NI 655*x*, and is recommended for all applications except the most demanding in regard to timing precision or signal integrity. For applications demanding the highest levels of signal quality and timing precision, NI recommends that you seriously consider following the recommendations of the terminated load configuration.

For applications requiring the highest levels of signal integrity and timing accuracy, NI strongly recommends carefully controlling the termination impedance at the end of the <u>transmission line</u>. To control the termination impedance, add a parallel termination resistor to ground as close as possible to the digital input pin of the device under test (DUT). In this configuration, the transmission line is terminated at both ends of the transmission line, which produces the highest possible signal integrity.

Ideally, the source impedance,  $Z_S$ , and the <u>characteristic impedance</u> of the transmission line,  $Z_0$ , should be kept as close as possible to 50  $\Omega$  as this will give you the best possible signal quality.

However, depending on your NI device, having all the lines terminated into 50  $\Omega$  may violate the maximum current specifications for your NI device. Refer to the NI 655*x* specifications for more information about the maximum current for your device to determine how many lines you can simultaneously terminate into 50  $\Omega$ .

While a  $Z_t$  of 50  $\Omega$  is ideal, you can also use values as high as 150  $\Omega$  without significantly affecting signal quality. Using this higher resistance value enables you to increase the voltage swing across the DUT and decrease the drive current requirements on your NI 655x.

Given that at DC there is effectively a voltage divider between the 50  $\Omega$  Z_S resistance and the termination resistance, having a terminated load reduces the largest possible voltage swings at the DUT according to the following formula: V_t = V_s*(Z_t/(Z_s + Z_t)

So, for a 50  $\Omega$  termination, programming a generation voltage level of 3.3 V at the NI 655*x* source produces a 3.3 V x (50/100) = 1.65 V. This reduced voltage swing at the DUT should be considered when you create your system.
### **NI 655x Acquisition Termination**

### **Unterminated Acquisition Configuration**

For acquisition operations, an unterminated configuration essentially implies that you set the input impedance of the NI device to 10 k $\Omega$  using NI-HSDIO. The unterminated configuration is easy to drive since it does not present a significant DC load to the source and preserves the signal amplitude. As a result, this is the recommended configuration for most applications. When you use this mode, however, it is very important that you follow the recommendations in this section to achieve the highest level of AC signal quality.

The same transmission line considerations discussed for the generation case are applicable to acquired signals. Programming an input impedance of 10 k $\Omega$  implies a reflection coefficient  $\Gamma_t$  of 0.99, or nearly full reflection and is effectively an unterminated input.

With all unterminated transmission lines, it is essential that you take care to match the source impedance of the transmission line to the characteristic impedance of the transmission line. The source matching in this configuration is particularly important, since there are significant reflections from the unterminated load (input of NI 655*x* in the case).

First, you should take care to ensure that your transmission line has a <u>characteristic impedance</u> of as close to 50  $\Omega$  as possible.

Therefore,  $Z_s$  (external source output impedance) should match  $Z_0 = 50 \Omega$  (the cable impedance) for 10 k $\Omega$  input configuration. To achieve this  $Z_s = 50 \Omega$ , you must determine the output impedance of your digital driver and add a series resistor as close as possible to the driver pin such that the output impedance of your buffer plus the value of the series resistor equal 50  $\Omega$ .

A 10 k $\Omega$  input configuration is *not* recommended for use with drivers that are not matched to the cable impedance at 50  $\Omega$ .

#### **Terminated Acquisition Configuration**

Input termination of 50  $\Omega$  is recommended for best signal integrity since there are no reflections back to the signal source, provided that the signal source can drive this load.

The 50  $\Omega$  input configuration reduces the signal swing seen by the NI 655*x* input comparators by half, assuming the signal source is 50  $\Omega$ . You should consider this amplitude reduction when configuring the input thresholds.

For example, a 50  $\Omega$  source with a 0 to 5 V step generation (into high-impedance) is seen as a 0 to 2.5 step source at the NI 655*x* acquisition comparators when you configure the NI 655*x* for 50  $\Omega$  input impedance.

### **NI 655x Termination Summary**

The following table provides a high-level summary of the termination considerations for your NI 655x. Click a title for more information about that type of termination.



	because of the mismatch on both ends. The recommended setup is to increase $Z_s$ to 50 $\Omega$ by adding series resistance at the source. $Z_s > 50 \Omega$ This configuration is not recommended because of the mismatch on both ends. <b>Note</b> You must reduce the operating frequency significantly to allow the signal reflections to sottle down. You
	must reduce the frequency proportional to cable length and impedance mismatch.
Z _t = 50 Ω	$Z_s = 50 \Omega$ This configuration is recommended for best signal integrity. $V_i = V_0/2$ Source must be able to drive 50 Ω
	$Z_{s} < 50 \Omega$ Source-side mismatch degrades signal integrity, but this configuration is still usable in many applications. $V_{i} = 50/(50 + Z_{s}) * V_{o}$
	$Z_s > 50 \Omega$ Source-side mismatch degrades signal integrity, but this configuration is still usable in many applications. $V_i = 50/(50 + Z_s) * V_0$

### Terminating Your NI 656x

Expand this section to learn about the termination configurations for <u>acquisition</u> and <u>generation</u> with the NI 656x.

Refer to <u>Termination</u> for more information about the theory and recommendations for system termination.

### NI 656x Generation Termination

## Generation Termination: Terminated Load Configuration DIO, DDC CLK OUT, and LVDS PFI Channels

The NI 656x requires a differential termination at the destination of 100  $\Omega$  to properly drive the LVDS logic levels and to maintain signal integrity.

LVDS is a current-driven technology. That is, a logic state is derived from the differential voltage generated by forcing a current through a known impedance. Forcing current one direction signifies a logic high level, and forcing current the alternate direction signifies a logic low level. As such, it requires that the current path be completed, through a 100  $\Omega$  resistor, at the destination. The differential voltage then seen at the receiver is a function of this resistor. LVDS levels are generated using a 100  $\Omega$  resistor at the receiver.

For applications requiring the highest levels of signal integrity and timing accuracy, NI strongly recommends carefully controlling the termination impedance at the end of the transmission line. In a differential environment, there is an effective virtual ground at the midpoint of there terminating resistor. In a 50  $\Omega$  single-ended environment, the transmission line is effectively matched with the 100  $\Omega$  differential impedance caused by this virtual ground effect.

#### **Unloaded Single-Ended PFI Channels**

A common configuration for your NI 656*x* is to configure the terminals for single-ended mode and connect them directly to your device under test (DUT). Most digital logic inputs have an input impedance of 1–10 K $\Omega$ . Therefore, connecting the NI 656*x* output terminals directly to the input of your DUT effectively creates a source-terminated configuration because the PFI channels of the NI 656*x* have a 50  $\Omega$  source impedance when in single-ended mode.

While this source configuration does not provide the absolute highest level of signal quality, there are many advantages to a source-terminated configuration. First, very good signal levels are possible if you ensure that you have the cleanest possible 50  $\Omega$  characteristic impedance transmission line. Second, this source-terminated configuration allows you to directly wire to your DUT without the need for additional termination resistors. Lastly, given that at DC there is effectively a voltage

divider between the 50  $\Omega$  Z_s resistance and the high-impedance Z_t of your DUT, having a source-terminated load preserves the largest possible voltage swings at the DUT according to the following formula: V_t = V_s*(Z_t/(Z_s + Z_t)

So, for a DUT with an input impedance of  $1 \text{ k}\Omega$ , programming a generation voltage level of 3.3 V at the NI 655*x* source produces a 3.3 V * (1000/1050) = 3.14 V swing.

The source-terminated load generates reflections in the transmission line. These reflections, however, are absorbed at the source and not rereflected back to the load, thus preserving the signal integrity. Practically, the source impedance does not perfectly match the transmission line impedance; therefore, a small fraction of the reflected wave is rereflected back toward the load. This second reflection creates small signal aberrations and a low level of inter-symbol interference.

For example, a 5% mismatch at the source results in a 2.5% re-reflection back at the load:

 $\Gamma_{\rm s} = (1.05 - 1)/(1.05 + 1) \approx 2.5\%$ 

#### Loaded Single-Ended PFI Channels

The source-terminated load configuration is easy to use with a terminated source, such as the NI 656*x*, and is recommended for all applications except the most demanding in regard to timing precision or signal integrity. For applications demanding the highest levels of signal quality and timing precision, NI recommends that you seriously consider following the recommendations of the terminated load configuration.

For applications requiring the highest levels of signal integrity and timing accuracy, NI strongly recommends carefully controlling the termination impedance at the end of the <u>transmission line</u>. To control the termination impedance, add a parallel termination resistor to ground as close as possible to the digital input pin of the device under test (DUT). In this configuration, the transmission line is terminated at both ends of the transmission line, which produces the highest possible signal integrity.

Ideally, the source impedance,  $Z_S$ , and the <u>characteristic impedance</u> of the transmission line,  $Z_0$ , should be kept as close as possible to 50  $\Omega$  as this will give you the best possible signal quality.

However, depending on your NI device, having all the lines terminated

into 50  $\Omega$  may violate the maximum current specifications. Refer to the <u>NI</u> <u>656x specifications</u> for more information about the maximum current for your device to determine how many lines you can simultaneously terminate into 50  $\Omega$ .

While a  $Z_t$  of 50  $\Omega$  is ideal, you can also use values as high as 300  $\Omega$  without significantly affecting signal quality. Using this higher resistance value enables you to increase the voltage swing across the DUT and decrease the drive current requirements on your NI 656x.

Given that at DC there is effectively a voltage divider between the 50  $\Omega$  Z_S resistance and the termination resistance, having a terminated load reduces the largest possible voltage swings at the DUT according to the following formula: V_t = V_s*(Z_t/(Z_s + Z_t)

For a 50  $\Omega$  termination, programming a generation voltage level of 3.3 V at the NI 656x PFI source produces a 3.3 V x (50/100) = 1.65 V at the DUT. This reduced voltage swing at the DUT should be considered when you create your system.

### NI 656x Acquisition Termination

### **DIO, STROBE, LVDS PFI Channels**

The NI 656x devices employ a single differential 100  $\Omega$  terminating resistor located at the differential receiver. This 100  $\Omega$  resistor guarantees signal quality in a 100  $\Omega$  differential environment and induces the correct voltage levels required by the LVDS standard.

This impedance is always present and is not software selectable.

### **Single-Ended PFI Channels**

For single-ended trigger operations, the NI 656*x* can only be used in a high-impedance load configuration because the input impedance of the NI 656*x* is set to 10 k $\Omega$ , when in single-ended mode. The high-impedance load configuration is easy to drive since it does not present a significant DC load to the source and preserves the signal amplitude. It is very important that you follow the recommendations in this section to achieve the highest level of AC signal quality.

The same transmission line considerations discussed for the generation case are applicable to acquired signals. The input impedance of 10 k $\Omega$  implies a reflection coefficient  $\Gamma_t$  of 0.99, or nearly full reflection.

With all high-impedance load transmission lines, it is essential that you take care to match the source impedance of the transmission line to the characteristic impedance of the transmission line. The source matching in this configuration is particularly important, since there are significant reflections from the high-impedance load (input of NI 656*x* in this case).

You should take care to ensure that your transmission line has a <u>characteristic impedance</u> of as close to 50  $\Omega$  as possible.

Therefore,  $Z_s$  (external source output impedance) should match  $Z_0 = 50 \Omega$  (the cable impedance) for 10 k $\Omega$  input configuration. To achieve this  $Z_s = 50 \Omega$ , you must determine the output impedance of your digital driver and add a series resistor as close as possible to the driver pin such that the output impedance of your buffer plus the value of the series resistor equal 50  $\Omega$ .

If you require 50  $\Omega$  termination for your acquisition application, consider using one of the <u>NI 655x</u> products.

### **Thermal Shutdown**

NI-HSDIO 1.1 and later support thermal shutdown capabilities with NI digital waveform generator/analyzers. These capabilities allow your device to detect when the device temperature has risen above its optimal operating temperature and to then power down, preventing damage to your device or improper performance.

Air circulation paths, fan settings, and space allowances are several factors that can influence device temperature. To prevent thermal shutdown, follow the guidelines in the *5. Installing the Hardware* section of the <u>NI Digital Waveform Generator/Analyzer Getting Started Guide</u>, which ships with your device.

In the event that your device powers down, you will be notified with an error message in one of the following ways:

- **NI-HSDIO**—NI-HSDIO will return an error when you use any of the functions that program the hardware or check hardware status, for example, the static acquisition and generation functions, commit functions, and self calibration function.
  - Note This particular error code is not returned by <u>niHSDIO_self_test</u> (or the <u>niHSDIO_Self_Test</u> VI) because it can only return a **0** for pass or a nonzero value for fail.
- MAX—Measurement & Automation Explorer will return an error message if you run a self-test on your device after it exceeds the thermal shutdown temperature. The thermal shutdown error continues to be reported until the device is successfully reset.

To re-enable your device after thermal shutdown, complete the following steps:

- 1. Power down the computer or chassis that contains the module.
- 2. Review the guidelines in the *Installing the Hardware* section of the <u>NI Digital Waveform Generator/Analyzer Getting Started Guide</u> that shipped with your module and make any necessary adjustments to ensure your module can cool itself effectively.
- Reset the device by either calling <u>niHSDIO_ResetDevice</u> (or using the <u>niHSDIO Reset Device</u> VI) or performing a device reset in MAX. For more information on performing a device reset in MAX, refer to the *Configuring in MAX* section of the <u>NI Digital Waveform</u>

Generator/Analyzer Getting Started Guide. The thermal shutdown error continues to be reported until the device is successfully reset.

### ΡΧΙ

This section contains information about integrating NI digital waveform generator/analyzers into a PXI-based measurement system.

The PXI architecture has built-in timing and triggering features that can synchronize multiple devices using the PXI trigger bus lines on the PXI backplane. Multiple devices in a modular instrumentation system can share a common Reference clock and synchronize to triggers that are distributed over controlled signal paths that ensure matched propagation. PC plug-ins with RTSI also provide an internal bus that can be accessed by multiple devices. Internal routing of these timing signals in PXI (and PC plug-ins with RTSI) eliminate complicated external wiring and the need to calculate propagation delays. Standardized timing protocols eliminate incompatibilities, giving you the best performance when synchronizing any kind of analog, digital, or timing measurements.

### **Chassis Considerations**

NI PXI modules are designed to operate in a PXI/Compact PCI chassis at specified environmental conditions. Device performance and reliability may be limited at temperatures above the specified operating range. For best performance, take the following precautions:

- Ensure that the ambient temperature is within the temperature listed in the specifications document for your module, and that the temperature is stable (±5 °C).
- Follow standard metrology practices.
- Use a PXI chassis with a well designed cooling system.

Operating the module outside the specified operating temperatures can increase bias currents in the electronic components, increase noise, accelerate drifts, and decrease product life. Beyond the maximum specified operating temperatures, the circuits perform differently than during the factory calibration, resulting in additional measurement errors which may not be accounted for by the Tempco specifications.

To minimize the temperature rise above ambient, position the chassis away from heat sources and clean the PXI/Compact PCI chassis air filter at regular intervals. Clean air filters are essential to ensuring that the devices operate at peak performance.

Operating under high humidity or dusty conditions can cause leakages between circuit components to increase and result in additional measurement errors.

### **Chassis Cooling Guidelines**

Follow these guidelines to optimize cooling and ensure best performance and reliability:

- Always run chassis with fans set on high. In newer NI chassis the settings are "HIGH" and "AUTO". On some older NI chassis the fan settings may be "HI" and "LO".
- Cover all empty slots in the chassis with a blank EMC slot filler panel.
- Remove and clean the inlet filters often to prevent buildup of dust and other foreign material that may restrict airflow.
- Locate the chassis such that the fan inlets and outlet vents are not obstructed. Keep other objects and equipment a minimum of 3 inches sway from the fan inlets.

For more information regarding cooling considerations, refer to your chassis documentation.

### PCI

Peripheral Component Interconnect (PCI) is a high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA. It has achieved widespread acceptance as a standard for PCs and workstations, and offers a theoretical maximum transfer rate of 132 Mbytes/s.

### RTSI

The Real-Time System Integration Bus (RTSI) is a timing bus developed by National Instruments. PCI devices can use the RTSI bus to share and exchange timing and control signals between multiple devices. The RTSI bus consists of the RTSI bus interface connector (located at the back of the device) and a RTSI ribbon cable.

When there are multiple SMC-based products in the PCI computer that need to be phase-locked to a common 10 MHz Reference clock, they can be connected with a RTSI cable. One of these connected devices should drive its Onboard Reference Clock to RTSI 7. All of the devices should receive their Reference clock from RTSI 7.

**Tip** The device on the end of the RTSI cable should drive the Onboard Reference clock, instead of a device located at the center of the cable. This configuration results in optimal signal integrity for the Reference clock.

### **Synchronizing Multiple Devices**

Synchronizing multiple devices can occur to varying degrees. On the simplest level, the devices may be essentially operating independently with triggers and events passed between them to regulate operations. However, in more tightly integrated systems, the Sample clocks are phase aligned between all the devices, and all devices are triggered at the same time. For the NI digital waveform generator/analyzers, this alignment is done using the Sample clock source, Reference clock source, and by using NI-TClk for clock alignment and trigger routing.

### Sample Clock Phase Alignment

#### • PXI Devices

For PXI digital waveform generator/analyzers using an internal clock source, the internal clock source can be phase aligned to the PXI_CLK10 signal on the backplane by selecting PXI_CLK10 as the Reference clock source. NI-TClk ensures that the Sample clock dividers on each PXI device are in phase for Sample clock alignment.

For PXI digital waveform generator/analyzers using an external clock source (CLK IN, STROBE, or PXI_STAR), ensure that the Sample clocks are aligned when presented to the devices. If you are using PXI_STAR as the external clock source, the matched length traces on the PXI backplane assist in keeping the distributed Sample clocks aligned. You can use a device like the NI PXI-5404 or NI PXI-6653 to distribute clocks on PXI_STAR.

#### PCI Devices

For PCI digital waveform generator/analyzers using an internal clock source, the internal clock source can be phase aligned to a 10 MHz reference signal on the RTSI 7 line of the RTSI connector. Configure the PCI device at one end of the RTSI cable to drive the Onboard Reference Clock onto RTSI 7, and configure all of the PCI devices to receive their Reference clock from RTSI 7. NI-TClk ensures that the Sample clock dividers on each device are in phase for Sample clock alignment.

For PCI digital waveform generator/analyzers using an external clock source (CLK IN or STROBE), ensure that the Sample clocks are aligned when presented to the devices.

### **Trigger Routing**

The NI-TClk software uses the PXI trigger bus/RTSI bus lines to deterministically pass triggers between multiple NI digital waveform generator/analyzers. Refer to the multidevice NI-TClk examples for more information. Deterministic trigger routing ensures that all NI digital waveform generator/analyzers in the system start on the same sample.

### Programming

Expand this book to view the topics that provide information about programming your NI waveform generator/analyzer with NI-HSDIO.

### **Getting Started with NI-HSDIO**

This topic describes how to begin using NI-HSDIO with your application development environment (ADE), lists any files to include in your application, and mentions special considerations to make with each ADE.

To successfully build your application, install NI-HSDIO. You also must install one of the following ADEs:

- LabVIEW
- LabWindows/CVI
- <u>Microsoft Visual C++</u>

### Using NI-HSDIO in LabVIEW

This topic assumes that you are using the National Instruments LabVIEW ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-HSDIO application in LabVIEW, follow these general steps:

- 1. Open an existing or new LabVIEW VI.
- 2. From the Function Palette, locate the NI-HSDIO VIs at **NI Measurements»NI-HSDIO**.
- 3. Select the VIs that you want to use and drop them on the block diagram to build your application.

#### **Example Programs for LabVIEW**

If you are using LabVIEW 7.0 or later, you can use the NI Example Finder to search or browse examples. NI-HSDIO examples are classified by keyword, so you can search for a particular device or measurement function.

To browse the NI-HSDIO examples available in LabVIEW, launch LabVIEW, click **Open»Examples**, and navigate to **Hardware Input and Output»Modular Instruments»NI-HSDIO**. You can also acess the examples using the Start menu, by selecting **Start»All Programs»National Instruments»NI-HSDIO»Examples**.

### Considerations for using the LabVIEW Real-Time Module

To develop an NI-HSDIO application in the LabVIEW Real-Time Module, follow the same steps used for developing any application in LabVIEW, using the NI-HSDIO LabVIEW VIs.



**Note** Applications running NI-HSDIO in the LabVIEW Real-Time Module on a real-time target may be compromised and/or slow at 64 MB.

#### **Hardware Support**

NI-HSDIO supports operating all NI digital waveform generator/analyzers on RT targets.

#### **Unsupported Features**

When using NI digital waveform generator/analyzers with the LabVIEW Real-Time Module, the following features are *not* supported:

- External calibration
- Express VIs

#### **Related Documentation**

- For configuration instructions for remote systems, refer to the *Remote Systems Help* in Measurement & Automation Explorer (MAX) by selecting Help»Help Topics»Remote Systems in MAX.
- For more information on the LabVIEW Real-Time Module, refer to the *LabVIEW Real-Time Module User Manual* at <u>ni.com/manuals</u>.
- For additional troubleshooting and support information, refer to the LabVIEW Real-Time Support main page.

### Using NI-HSDIO in LabWindows/CVI

This topic assumes that you are using the LabWindows™/CVI™ ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-HSDIO application in LabWindows/CVI, follow these general steps:

- 1. Open an existing or new project file.
- 2. Load the NI-HSDIO function panel from the location specified in the <u>NI-HSDIO Instrument Driver Readme</u>.
- 3. Use the function panel to navigate the function hierarchy and generate function calls with the proper syntax and variable values.

#### **Example Programs for LabWindows/CVI**

If you are using LabWindows/CVI 7.0 or later, you can use the NI Example Finder to search or browse examples. NI-HSDIO examples are classified by keyword, so you can search for a particular device or measurement function.

To browse the NI-HSDIO examples available in LabWindows/CVI, launch LabWindows/CVI, select Help»NI Example Finder, and navigate to Hardware Input and Output»Modular Instruments»NI-HSDIO. You can also acess the examples using the Start menu, by selecting Start»All Programs»National Instruments»NI-HSDIO»Examples.

### Using NI-HSDIO in Visual C++

This topic assumes that you are using the Microsoft Visual C++ ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-HSDIO application in Visual C++, follow these general steps:

- 1. Open an existing or new Visual C++ project.
- Create source files of type .c (C source code) or .cpp (C++ source code) and add them to the project. Make sure that you include the NI-HSDIO header file, niHSDIO.h, as follows in your source code files: #include "niHSDIO.h"
- Specify the directory that contains the NI-HSDIO header file under the Preprocessor»Additional include directories settings in your compiler—for Visual C++ 6.0 these files are under Project»Settings»C/C++. The NI-HSDIO header files are located in the .\Include directory within your NI-HSDIO directory.
- 4. Add the NI-HSDIO import library niHSDIO.lib to the project under Link»General»Object/Library Modules. The NI-HSDIO import library files are located in the .\Lib directory within your NI-HSDIO directory.
- 5. Add NI-HSDIO function calls to your application.
- 6. Build your application.

### Example Programs for Visual C++

You can find example programs at the location specified in the <u>NI-HSDIO</u> <u>Instrument Driver Readme</u> or from the Start menu by selecting **Programs**»National Instruments»NI-HSDIO»Examples»c.

All C examples are installed with support makefiles that are executable from a command prompt. To build examples, run vcvars32 to set up your build environment. From the example directory, type nmake /f makefilename.



**Note** You might get a compiler error if the example uses NI-HWS because some operating systems may not be able to resolve the \$(PROGRAMFILES) variable referred to in the makefile. Replace HWSPATH=\$(PROGRAMFILES)\National Instruments\NI-HWS in the makefile with the absolute install path for NI-HWS. The default absolute path is specified in the <u>NI-HSDIO Instrument Driver</u> <u>Readme</u>.

#### **Special Considerations**

#### **String Passing**

To pass strings, pass a pointer to the first element of the character array. Be sure that the string is null-terminated.

#### **Parameter Passing**

By default, C passes parameters by value. Remember to pass pointers to variables when you need to pass by address.

### **Digital Waveform Data Representation**

NI-HSDIO supports two data types to represent digital waveform data. The first data type is as a one-dimensional array of integer data. The other data type is the digital waveform data type (WDT). VIs and functions in NI-HSDIO that write or read/fetch digital waveform data can accept either data type.

The WDT is required when you use more than the 0 and 1 states in a waveform, for example, if your waveform includes Z's, X's, H's, or L's. Binary data uses only 1 bit per channel per sample, and this cannot represent more than two states. DWDT uses 8 bits per channel per sample and can represent extended digital states.

One difference between U32 array data and WDT data is memory usage. Each sample of U32 data occupies four bytes of PC memory, independent of the number of channels being used. Each sample of a digital WDT data occupies 1 byte for each channel used, but unused channels do not occupy memory. Thus, a 1,000-sample waveform of 16 channels represented in a raw U32 array would occupy (4 Bytes/sample) x (1,000 samples) = 4 kBytes.

The same 1,000 sample waveform represented in a WDT would occupy approximately

(1 byte/channel) x (16 channels/sample) x (1,000 samples) = 16 kBytes.

#### **Related Topics:**

- Digital Waveform Data Representation in LabVIEW
- Digital Waveform Data Representation in C

# Digital Waveform Data Representation in LabVIEW

NI-HSDIO supports two data types to represent digital waveform data. The first data type is as a one-dimensional array of integer data. The following figures show the LabVIEW control for this data type and an example of a VI wired to the data type.



The following figures show the control for the LabVIEW digital waveform data type (WDT) and an example of a VI wired to the data type.



VIs in NI-HSDIO that write or read/fetch digital waveform data can accept either data type.

The digital WDT includes not only the digital values but also can contain additional attribute information, such as time stamps (shown in the figure of the WDT control). Raw data, such as the U32 array data, consist only of the digital values.

If you intend to graph data using the LabVIEW digital waveform graph, NI recommends that you use the WDT, as it can be directly wired, as shown in the following figure.





#### **Creating Waveforms with per Cycle Tristate Capabilities**

NI-HSDIO provides the <u>niHSDIO Convert Binary to DWDT</u> VI to convert digital waveforms containing Z values from a U32 array of binary data values and a U32 array of masks of which channels to tristate.

### **Digital Waveform Data Representation in C**

As in <u>LabVIEW</u>, the NI-HSDIO C functions represent digital waveform data in two formats. Data can be represented as a one-dimensional array of binary data or in an expanded waveform data type (WDT) format.

Each integer value in the array corresponds to the state of one channel during one sample of the waveform. The following example shows writing a waveform in the waveform data type format.

```
error = niHSDIO_WriteNamedWaveformWDT( vi, waveformName,
NUM_SAMPLES, NIHSDIO_VAL_GROUP_BY_SAMPLE, data);
```

If your device supports <u>per cycle tristate</u>, use the waveform data type format to use this feature. The digital state values for 0, 1, and Z are defined in the niHSDIO.h header file under Digital Channel States, and are defined as follows:

#define NI_DIO_0 0
#define NI_DIO_1 1
#define NI_DIO_Z 2
#### File I/O and Digital Waveform Data

You will often store digital waveform data on a disk and use file I/O in your program to retrieve the data. A convenient file format for storing digital waveform data is the .hws file format. NI Hierarchical Waveform Storage (NI-HWS) provides a set of functions for easily and efficiently storing and retrieving digital waveform data files. NI-HWS works with binary data (U8, U16, and U32) and <u>Waveform Data Type (WDT) data</u>, as shown in the LabVIEW code snippets in the following figure:



NI-HWS can convert some waveform data from the data type in which it was stored to a different data type when the data is read or retrieved. The table below shows the supported type conversions. A "Yes" in a box means that data stored as one data type can be retrieved as the other data type; a dash means the conversion is not supported.

	Retrieved Type						
Stored Type	Digital WDT	Digital 1D U8	Digital 2D U8	Digital 1D U16	Digital 2D U16	Digital 1D U32	Digital 2D U32
Digital WDT	Yes						
Digital 1D U8	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Digital 2D U8	Yes		Yes		Yes		Yes
Digital 1D U16	Yes	Yes*	Yes*	Yes	Yes	Yes	Yes
Digital	Yes		Yes*		Yes		Yes

2D U16							
Digital 1D U32	Yes	Yes*	Yes*	Yes*	Yes*	Yes	Yes
Digital 2D U32	Yes		Yes*		Yes*	_	Yes

*For digital waveforms, smaller data types can read larger data types with the following restrictions: the data must be mapped and all mapped bits must be within the smaller data types range.

When digital data is retrieved as WDT, the Dynamic Channel List attribute is used to determine the number of channels of data that the digital WDT contains. When data is initially stored as WDT, this attribute is automatically set.

The NI Digital Waveform Editor (DWE) uses digital .hws files as its native file format for easy transfer of data between the DWE and your programming environment. The following figure illustrates the relationship between your programming environment, the DWE, and .hws files.



# **Programming Flow**

The diagrams in the following sections show the basic programming flow of applications using NI-HSDIO for waveform generation, waveform acquisition, simultaneous generation and acquisition, and static generation and acquisition.

The functions and VIs are categorized under these main topics to assist you in understanding where you should call a function or VI in your applications. Functions and VIs that do not fall into the programming flow are considered Utility functions that perform various tasks, such as resetting the device.

## **Initialize Your Session**

For any application you write, open a session to establish communication with the instrument by using one of the following two functions:

LabVIEW VIs	C Functions
niHSDIO Init Acquisition Session	niHSDIO_InitAcquisitionSession
niHSDIO Init Generation Session	niHSDIO_InitGenerationSession

Use these functions for both dynamic and static data operations. Use the ViSession handle or instrument handle returned by these functions to identify the NI device in all subsequent NI-HSDIO function calls.

**Tip** Do not confuse these two functions with <u>niHSDIO Initiate</u> or <u>niHSDIO Initiate</u>, which are used to start a dynamic data operation.

In addition to establishing a session with the device, these two functions also send initialization commands to set the device to the state necessary for the operation of the instrument driver. These two functions can also perform a number of additional tasks, such as verifying that the instrument driver is valid for the device and resetting the device to a known state.

# **Select Channels**

Your digital device contains multiple channels. Use the following functions to specify which channels you want to use in your data operations.

LabVIEW VIs	C Functions
niHSDIO Assign Dynamic Channels	niHSDIO_AssignDynamicChannels
niHSDIO Assign Static Channels	niHSDIO_AssignStaticChannels

You can configure a channel for more than one simultaneous data operation. A channel can be simultaneously configured for the following operations:

- Dynamic Generation and any (Static and/or Dynamic) Acquisition
- Static Generation and any (Static and/or Dynamic) Acquisition
- Both Static and Dynamic Acquisition

Note You *cannot* configure a particular channel for simultaneous dynamic *and* static generation.

#### **Configure the Hardware**

Use Configuration VIs and functions to set up the triggers, voltage levels, and other settings and features needed for your data operation.

Acquisition Configuration Functions and Generation Configuration Functions have tables that list VIs and functions for configuring the NI digital waveform generator/analyzer. Click the VI/function for more information.

# **Acquisition Configuration Functions**

LabVIEW VIs	C Functions
niHSDIO Configure Sample Clock	niHSDIO_ConfigureSampleClock
niHSDIO Configure Acquisition Size	niHSDIO_ConfigureAcquisitionSize
niHSDIO Configure Data Interpretation	niHSDIO_ConfigureDataInterpretation
niHSDIO Configure	niHSDIO_ConfigureDataVoltageLogicFamily
Voltage (Polymorphic	niHSDIO_ConfigureDataVoltageCustomLevels
VI)	niHSDIO_ConfigureTriggerVoltageLogicFamily
	niHSDIO_ConfigureTriggerVoltageCustomLevels
	niHSDIO_ConfigureEventVoltageLogicFamily
	niHSDIO_ConfigureEventVoltageCustomLevels
niHSDIO Configure	niHSDIO_ConfigureDataPosition
Data Position	
niHSDIO Configure Data Position Delay	niHSDIO_ConfigureDataPositionDelay
niHSDIO Configure	niHSDIO_ConfigureDigitalEdgeStartTrigger
Trigger (Dolymorphic \/l)	niHSDIO_ConfigurePatternMatchStartTrigger
	niHSDIO_ConfigurePatternMatchStartTriggerU32
	niHSDIO_ConfigureSoftwareStartTrigger
	<u>niHSDIO_DisableStartTrigger</u>
	niHSDIO_ConfigureDigitalEdgeRefTrigger
	niHSDIO_ConfigurePatternMatchRefTrigger
	niHSDIO_ConfigurePatternMatchRefTriggerU32
	niHSDIO_ConfigureSoftwareRefTrigger
	niHSDIO_DisableRefTrigger
	niHSDIO_ConfigureDigitalEdgeAdvanceTrigger
	niHSDIO_ConfigurePatternMatchAdvanceTrigger

	niHSDIO_ConfigurePatternMatchAdvanceTriggerU32
	niHSDIO_ConfigureSoftwareAdvanceTrigger
	niHSDIO_DisableAdvanceTrigger
	niHSDIO_ConfigureDigitalLevelPauseTrigger
	niHSDIO_ConfigurePatternMatchPauseTrigger
	niHSDIO_ConfigurePatternMatchPauseTriggerU32
	<u>niHSDIO_DisablePauseTrigger</u>
<u>niHSDIO Export</u> <u>Signal</u>	niHSDIO_ExportSignal
niHSDIO Configure Ref Clock	niHSDIO_ConfigureRefClock

**Note** Some advanced attributes are not available through a configuration VI or function. Set these attributes directly using the <u>NI-HSDIO Property Node</u> VI or one of the niHSDIO SetAttribute functions. Refer to <u>Using Attributes with NI-HSDIO</u> for more information on attribute programming. Refer <u>Advanced Attributes</u> for a list of those attributes configurable only through a property node or SetAttribute function.

 $\overline{\mathbb{N}}$ 

## **Generation Configuration Functions**

LabVIEW VIs	C Functions
niHSDIO Configure Sample Clock	niHSDIO_ConfigureSampleClock
niHSDIO Configure Generation Mode	niHSDIO_ConfigureGenerationMode
niHSDIO Configure Generation Repeat	niHSDIO_ConfigureGenerationRepeat
<u>niHSDIO Configure</u> <u>Waveform to</u> <u>Generate</u>	niHSDIO_ConfigureWaveformToGenerate
niHSDIO Configure Script to Generate	niHSDIO_ConfigureScriptToGenerate
niHSDIO Configure	niHSDIO_ConfigureInitialState
Initial <u>State</u> (Polymorphic VI)	niHSDIO_ConfigureInitialStateU32
niHSDIO Configure	<u>niHSDIO_ConfigureIdleState</u>
Idle State (Polymorphic VI)	niHSDIO_ConfigureIdleStateU32
niHSDIO Configure	<u>niHSDIO_ConfigureDataVoltageLogicFamily</u>
Voltage (Polymorphic VI)	niHSDIO_ConfigureDataVoltageCustomLevels
	niHSDIO_ConfigureTriggerVoltageLogicFamily
	<pre>niHSDIO_ConfigureTriggerVoltageCustomThresholds</pre>
	niHSDIO_ConfigureEventVoltageLogicFamily
	niHSDIO_ConfigureEventVoltageCustomLevels
<u>niHSDIO Configure</u> <u>Data Position</u>	niHSDIO_ConfigureDataPosition
niHSDIO Configure Data Position Delay	niHSDIO_ConfigureDataPositionDelay
niHSDIO Configure	<u>niHSDIO_ConfigureDigitalEdgeStartTrigger</u>
Trigger (Polymorphic \/l)	niHSDIO_ConfigurePatternMatchStartTrigger
	niHSDIO_ConfigurePatternMatchStartTriggerU32

	niHSDIO_ConfigureSoftwareStartTrigger
	niHSDIO_DisableStartTrigger
	niHSDIO_ConfigureDigitalEdgeScriptTrigger
	<u>niHSDIO_ConfigureDigitalLevelScriptTrigger</u>
	niHSDIO_ConfigureSoftwareScriptTrigger
	niHSDIO_DisableScriptTrigger
	<u>niHSDIO_ConfigureDigitalLevelPauseTrigger</u>
	niHSDIO_ConfigurePatternMatchPauseTrigger
	niHSDIO_ConfigurePatternMatchPauseTriggerU32
	niHSDIO_DisablePauseTrigger
niHSDIO Export	niHSDIO_ExportSignal
<u>Signal</u>	
niHSDIO Configure	niHSDIO_ConfigureRefClock
Ref Clock	

Note Some advanced attributes are not available through a configuration VI or function. Set these attributes directly using the <u>NI-HSDIO Property Node</u> VI or one of the niHSDIO SetAttribute functions. Refer to <u>Using Attributes with NI-HSDIO</u> for more information on attribute programming. Refer <u>Advanced Attributes</u> for a list of those attributes configurable only through a property node or SetAttribute function.

#### **Advanced Attributes**

These advanced attributes, called properties in LabVIEW, are not available through a configuration VI or function. Set these attributes directly using the <u>niHSDIO Property Node</u> or the niHSDIO Set Attribute functions. Refer to <u>Using Attributes with NI-HSDIO</u> for more information on attribute programming.

#### Advanced Acquisition Attributes

LabVIEW Property	C Attribute
<u>Sample</u> <u>Clock</u> Impedance	NIHSDIO_ATTR_SAMPLE_CLOCK_IMPEDANCE
<u>Ref Clock</u> Impedance	NIHSDIO_ATTR_REF_CLOCK_IMPEDANCE
<u>Digital Edge</u> <u>Start Trigger</u> Impedance	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_IMPE
<u>Digital Edge</u> <u>Start Trigger</u> <u>Position</u>	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_POSI [^]
<u>Digital Edge</u> <u>Start Trigger</u> <u>Terminal</u> Configuration	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_TERM
<u>Digital Edge</u> <u>Reference</u> <u>Trigger</u> <u>Impedance</u>	NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_IMPEDA
<u>Digital Edge</u> <u>Reference</u> <u>Trigger</u> <u>Position</u>	NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_POSITIC
<u>Digital Edge</u> <u>Reference</u> <u>Trigger</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_TERMIN
<u>Digital Edge</u> <u>Advance</u> <u>Trigger</u> <u>Impedance</u>	NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIGGER_IN

<u>Digital Edge</u> <u>Advance</u> <u>Trigger</u> <u>Position</u>	NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIGGER_P
Digital Edge Advance Trigger Terminal Configuration	NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIGGER_T
<u>Ready For</u> <u>Advance</u> <u>Event</u> <u>Terminal</u> Configuration	NIHSDIO_ATTR_READY_FOR_ADVANCE_EVENT_TERMI
<u>Ready For</u> <u>Start Event</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_READY_FOR_START_EVENT_TERMINA
<u>Digital Level</u> <u>Pause</u> <u>Trigger</u> Impedance	NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGGER_IMP
<u>Digital Level</u> <u>Pause</u> <u>Trigger</u> <u>Position</u>	NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGGER_POS
Digital Level Pause Trigger Terminal Configuration	NIHSDIO ATTR DIGITAL LEVEL PAUSE TRIGGER TER
<u>Exported</u> <u>Sample</u> <u>Clock Mode</u>	NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_MODE
Exported Sample	NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_DELAY

<u>Clock Delay</u>	
<u>Fetch</u>	NIHSDIO_ATTR_FETCH_RELATIVE_TO
Relative to	
Fetch Offset	NIHSDIO_ATTR_FETCH_OFFSET
<u>Fetch</u>	NISHDIO_ATTR_FETCH_BACKLOG
<u>Backlog</u>	

#### **Advanced Generation Attributes**

LabVIEW Property	C Attribute
<u>Sample</u> <u>Clock</u> Impedance	NIHSDIO_ATTR_SAMPLE_CLOCK_IMPEDANCE
<u>Ref Clock</u> Impedance	NIHSDIO_ATTR_REF_CLOCK_IMPEDANCE
<u>Data Active</u> <u>Event</u> Position	NIHSDIO_ATTR_DATA_ACTIVE_EVENT_POSITION
<u>Marker Event</u> <u>Pulse</u> Polarity	NIHSDIO_ATTR_MARKER_EVENT_PULSE_POLARITY
Marker Event Position	NIHSDIO_ATTR_MARKER_EVENT_POSITION
<u>Exported</u> <u>Sample</u> <u>Clock Mode</u>	NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_MODE
<u>Exported</u> <u>Sample</u> <u>Clock Delay</u>	NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_DELAY
<u>Digital Edge</u> <u>Start Trigger</u> <u>Impedance</u>	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_IMPE
Digital Edge Start Trigger Position	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_POSI [_]
Digital Edge Start Trigger Terminal Configuration	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_TERN
Digital Edge Reference	NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_IMPEDA

<u>Trigger</u> Impedance	
<u>Digital Edge</u> <u>Reference</u> <u>Trigger</u> <u>Position</u>	NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_POSITIC
<u>Digital Edge</u> <u>Reference</u> <u>Trigger</u> <u>Terminal</u> Configuration	<u>NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_TERMIN</u>
<u>Digital Edge</u> <u>Script</u> <u>Trigger</u> <u>Impedance</u>	NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGGER_IMPI
Digital Edge Script Trigger Terminal Configuration	NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGGER_TERI
<u>Digital Level</u> <u>Script</u> <u>Trigger</u> Impedance	NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGGER_IMF
Digital Level Script Trigger Terminal Configuration	NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGGER_TEF

#### Hardware Comparison Attributes

LabVIEW Property	C Attribute
Advanced:Hardware Compare:Hardware Compare Mode	NIHSDIO_ATTR_HWC_HARDWARE_COMPARE_M
Advanced:Hardware Compare:Filter Repeated Sample Errors	NIHSDIO ATTR HWC FILTER REPEATED SAMPI
Advanced:Hardware Compare:Samples Compared	NIHSDIO_ATTR_HWC_SAMPLES_COMPARED
Advanced:Hardware Compare:Number Of Sample Errors	NIHSDIO ATTR HWC NUM SAMPLE ERRORS
Advanced:Hardware Compare:Sample Error Backlog	NIHSDIO ATTR HWC SAMPLE ERROR BACKLO
Advanced:Hardware Compare:Samples Error Buffer Overflowed	NIHSDIO ATTR HWC SAMPLE ERROR BUFFER

#### Acquiring or Generating Static Data

After you configure the acquisition or generation, you can <u>acquire or</u> <u>generate the static data</u> using the static read/write VIs and functions.



## **Reading and Writing Static Data**

Static acquisition and static generation are software-timed operations that can be performed on any number of channels. Static operations can performed using the following functions:

LabVIEW VIs	C Functions
niHSDIO Assign Static Channels	niHSDIO_AssignStaticChannels
niHSDIO Read Static (U32)	niHSDIO_ReadStaticU32
<u>niHSDIO Write Static (U32)</u>	niHSDIO_WriteStaticU32

#### **Acquiring Dynamic Data**

After you have configured the acquisition, you can acquire data using the <u>read functions</u> or using the <u>initiate and fetch functions</u>.



# Read

The Read functions are an easy way to acquire data from the device. These functions initiate an acquisition, fetch the acquired data, and return control to your program after all the requested data has been acquired. If you have not configured a <u>Start trigger</u>, the device immediately begins acquiring the data.

The Read VIs/functions are shown in the following table.

LabVIEW VIs	C Functions
Use one of the following instances of the	niHSDIO_ReadWaveformU32
niHSDIO Read Waveform polymorphic	niHSDIO_ReadWaveformU16
<ul> <li>niHSDIO Read Waveform (U32)</li> <li>niHSDIO Read Waveform (U16)</li> </ul>	niHSDIO_ReadWaveformU8
	niHSDIO_ReadMultiRecordU32
<ul> <li>niHSDIO Read Waveform (U8)</li> </ul>	niHSDIO_ReadMultiRecordU16
<ul> <li>niHSDIO Read Waveform (WDT)</li> </ul>	niHSDIO_ReadMultiRecordU8
<ul> <li>niHSDIO Read Multi Record (2D U32)</li> </ul>	
<ul> <li>niHSDIO Read Multi Record (2D U16)</li> </ul>	
<ul> <li>niHSDIO Read Multi Record (2D U8)</li> </ul>	
<ul> <li>niHSDIO Read Multi Record (1D WDT)</li> </ul>	

You can configure the maximum length of time to allow the measurement operation to complete. If the measurement operation does not complete within this time interval, the function returns a timeout error.

 $\overline{\mathbb{N}}$ 

**Note** If you are programming in C, you must declare an array in your program. This array allocates space for the data that is acquired with the read function. LabVIEW users do not need to declare an array because the <u>niHSDIO Read Waveform</u> VI handles the array.

#### **Initiate and Fetch**

Use the Initiate and Fetch functions to perform other operations while the device acquires data. Initiate begins the acquisition operation and returns control to your program. Fetch transfers the acquired data from acquisition onboard memory to your application.

The Initiate and Fetch VIs/functions are shown in the following table.

LabVIEW VIs	C Functions
niHSDIO Initiate	niHSDIO_Initiate
Use one of the following instances of the	niHSDIO_FetchWaveformU32
niHSDIO Fetch Waveform polymorphic	niHSDIO_FetchWaveformU16
VI: • niUSDIO Eatch Wayoform (U22)	niHSDIO_FetchWaveformU8
<ul> <li>niHSDIO Fetch Waveform (U16)</li> </ul>	niHSDIO_FetchMultiRecordU32
<ul> <li>niHSDIO Fetch Waveform (U8)</li> </ul>	niHSDIO_FetchMultiRecordU16
<ul> <li>niHSDIO Fetch Waveform (WDT)</li> </ul>	niHSDIO_FetchMultiRecordU8
<ul> <li>niHSDIO Fetch Multi Record (2D</li> </ul>	
<ul> <li>niHSDIO Fetch Multi Record (2D U16)</li> </ul>	
<ul> <li>niHSDIO Fetch Multi Record (2D U8)</li> </ul>	
<ul> <li>niHSDIO Fetch Multi Record (1D WDT)</li> </ul>	

Note If you are programming in C, you must declare an array in your program. This array allocates space for the data that is acquired with the fetch functions. LabVIEW users do not need to declare an array because the Fetch VIs allocate the array.

If you fetch the data while the device is still acquiring, the function waits until all the requested data has been acquired. If the data is not acquired within the time specified with the timeout parameter, NI-HSDIO returns an error. If you want to stop the device before it finishes, use the <u>niHSDIO</u> <u>Abort</u> VI or the <u>niHSDIO</u> <u>Abort</u> function. However, you cannot fetch after aborting.

- Tip Read the <u>NIHSDIO_ATTR_FETCH_BACKLOG</u> attribute before calling the fetch function to determine the number of samples available, or use the <u>Fetch Backlog</u> property in LabVIEW (Dynamic Acquisition»Fetch Backlog) for the same purpose.
- **Note** The <u>read functions</u> combine the initiate and fetch functions into one call. The read functions are more suitable for simple acquisitions, while initiate/fetch is better suited for complex applications that involve triggering.

#### **Making Multirecord Acquisitions**

NI-HSDIO supports multirecord acquisitions, which allow you to capture multiple, triggered waveforms without software intervention. NI-HSDIO stores each <u>record</u> in separate memory locations on the NI device.

The main benefit of multirecord acquisitions is that you can quickly acquire numerous triggered waveforms. Multirecord acquisitions allow hardware rearming of the NI device before the data is <u>fetched</u>. Therefore, the rearm time, or the time when the NI device is not ready for a trigger, is extremely small, often from 1 to 100  $\mu$ s, depending on the record length and the device. This short rearm time allows you to capture data whether the triggers occur microseconds or many days apart.

#### **Fetching Multirecord Acquisitions**

You use the same <u>fetch functions</u> for retrieving multirecord acquisitions as you do for single-record acquisitions. However, you must also specify the starting record and the number of records to fetch.

Fetching multiple records with a single fetch function requires understanding the order of the returned waveforms. If you are using a Cbased language, the waveforms are packed into a one-dimensional array. For LabVIEW users, the waveforms are returned as a two-dimensional array (where rows represent records and columns represent samples). You can use the Index Array VI to extract the waveform of interest.

You can also fetch each record individually using the following procedure:

- 1. Set the number of records to fetch to 1.
- 2. Use a loop to set the starting record as the zero-based index of the record you want to fetch.
- 3. Call one of the <u>fetch functions/VIs.</u>

#### **Generating Dynamic Data**

After you configure the generation and write your waveform(s) to the device, you can generate the data by calling an <u>initiate function</u>. You can generate data one waveform at a time, or you can use <u>scripts</u> to generate complex sequences of waveforms.



#### Writing Waveforms to Your Device

Before you can generate any data, you must write your waveform(s) to the device <u>onboard memory</u>. Use the Write Named Waveform VIs/functions to write waveform data from your PC memory to your onboard device memory.

The Write Named Waveform VIs/functions are shown in the following table.

LabVIEW VIs	C Functions
Use one of the following	niHSDIO_WriteNamedWaveformU32
instances of the <u>niHSDIO</u>	niHSDIO_WriteNamedWaveformU16
polymorphic VI:	niHSDIO_WriteNamedWaveformU8
niHSDIO Write	niHSDIO_WriteNamedWaveformWDT
<ul> <li>Named Waveform (U32)</li> <li>niHSDIO Write Named Waveform (U16)</li> <li>niHSDIO Write Named Waveform (U8)</li> <li>niHSDIO Write Named Waveform (WDT)</li> <li>niHSDIO Write Named Waveform Erom File (HWS)</li> </ul>	niHSDIOWriteNamedWaveformFromFileHWS

You can associate names with each waveform you write to the device. Naming waveforms is optional if you are writing a single waveform to the device and are not using scripts. You must name each waveform if you write multiple waveforms to your device. Use

<u>niHSDIO_ConfigureWaveformToGenerate</u> to select which named waveform is generated at <u>Initiate</u>. However, you must also name each waveform when using scripts, as the script generate statement uses the waveform name to know which waveform to generate. Note Select Programming»Reference»Script Instructions from the table of contents of this help file for more information on the generate statement and other scripting instructions.

When using very large waveforms, it may be problematic to allocate enough PC memory to perform a single Write Named Waveform call. You can write large waveforms to your device by writing smaller blocks at a time. Use the <u>niHSDIO Allocate Named Waveform</u> VI and <u>niHSDIO AllocateNamedWaveform</u> function and Write Named Waveform VI/functions (listed in the previous table) to accomplish this task.

Note An easier way to handle the task in the example below is by using niHSDIO Write Named Waveform From File (HWS), as this VI/function handles memory allocation for you.

Refer to the following code snippets for an example of writing a 1 MS waveform to onboard memory in LabVIEW and in C.



#define BLOCK_SIZE 8192

ViUInt32 data[BLOCK_SIZE];

niHWS_OpenFile("mydata.hws", niHWS_Val_ReadOnly, &fileHandle;. niHWS_GetWfmReference (fileHandle, VI_NULL, VI_NULL, &wfmRef);

/* reserve onboard memory, name the waveform "myWfm" */

niHWS_GetWfmI32Attribute (wfmRef, niHWS_Attr_WaveformSize, &wfmSize); niHSDIO_AllocateNamedWaveform (instrHdl, "myWfm", wfmSize);

```
/* write waveform 1 block at a time */
numSamplesWritten = 0;
while (numSamplesWritten <= wfmSize)
{
    /* Read BLOCK_SIZE samples from .hws file, put in data */
    niHWS_ReadDigitalU32(wfmRef, BLOCK_SIZE, data,
    &cactualSamplesRead);
    niHSDIO_WriteNamedWaveformU32 (instrHdl, "myWfm",
    actualSamplesRead, data);
    numSamplesRead = numSamplesRead + actualSamplesRead;
}
.</pre>
```

Each call to a Write Named Waveform VI/function writes to the end of the most previously written data.



**Note** Closing the session using the <u>niHSDIO close</u> VI or the <u>niHSDIO_close</u> function also deletes all waveforms from your device. You can manually delete a single named waveform from <u>onboard memory</u> by calling the <u>niHSDIO Delete Named</u> <u>Waveform</u> VI or the <u>niHSDIO_DeleteNamedWaveform</u> function.

Note If you try to write past the end of a waveform, NI-HSDIO returns an error.

Note NI digital waveform generator/analyzers require blocks be multiples of 32 samples for NI 654x/655x devices or 64 samples for NI 656x devices when writing to preallocated waveforms. The overall waveform size does not have this restriction, but it must be even for the NI 654x/655x devices or a multiple of four for the NI 656x (a multiple of eight if the NI 656x is in DDR mode). The last call to Write Named Waveform should write enough data to fill the waveform.

#### **Generating Data in Single-Waveform Mode**

You can generate data in one of two generation modes: waveform or <u>scripted</u>. Use the <u>niHSDIO Configure Generation Mode</u> VI or the <u>niHSDIO_ConfigureGenerationMode</u> function to switch between the two modes. The default mode is waveform.

In waveform mode, call <u>initiate</u> to generate the waveform you specified by calling the <u>niHSDIO Configure Waveform to Generate</u> VI or the <u>niHSDIO_ConfigureWaveformToGenerate</u> function. Once you have generated the waveform, you can call the <u>niHSDIO Configure Waveform</u> To Generate VI or the <u>niHSDIO_ConfigureWaveformToGenerate</u> function again to switch to a different named waveform.

Tip You can configure whether you want to generate a single waveform once, n times, or continuously by calling the <u>niHSDIO</u> <u>Configure Generate Repeat</u> VI or the <u>niHSDIO_ConfigureGenerationRepeat</u> function.

# Generating Multiple Waveforms/Linking & Looping

The second generation mode, scripted, allows you to link and loop multiple waveforms in complex combinations.

Use the <u>niHSDIO Configure Generation Mode</u> VI or the <u>niHSDIO_ConfigureGenerationMode</u> function to switch to scripted mode.

Write all waveforms that are referenced in the script using a <u>Write Named</u> <u>Waveform</u> call, and associate the proper names to them.

After waveforms are written to your device, use the <u>niHSDIO Write</u> <u>Script</u> VI or the <u>niHSDIO_WriteScript</u> function to write the script(s) containing the generation instructions to be executed. Multiple scripts can exist on your device simultaneously. If you write multiple scripts to your device, you must select the one you wish to execute by calling the <u>niHSDIO Configure Script To Generate</u> VI or the <u>niHSDIO_ConfigureScriptToGenerate</u> function. Call <u>initiate</u> to execute the selected script.

Note Internally, the script stores physical device memory locations to refer to named waveforms. Thus, write all waveforms to the device *before* writing the script, or the device does not know where the waveform is located. The <u>initiate function</u> produces an error if this rule is violated. If you delete waveforms and rewrite them, rewrite the script to update it with the new locations, even if the script text has not changed.

#### **Related Topics:**

- <u>Scripts</u>
- <u>Common Scripting Use Cases</u>
- <u>Scripting Instructions</u>

#### **Comparing Response Data with Expected Data**

You can generate stimulus data and acquire the response data for analysis by synchronizing the generation and acquisition sessions.

To perform software comparison, you must transfer all data to the host computer for post-processing, which makes this method suitable for slower-speed applications. Transferring all the data to the host computer may exceed computer bandwidth limitations if more data is to be acquired than can fit on the testers onboard memory. For this situation and other cases that require faster comparison rates, <u>real-time hardware comparison</u> may be used.

# $\overline{\mathbb{N}}$

**Note** Real-time hardware comparison (HWC) is supported only on the <u>NI 655x</u> family of devices.

There are two configurations for comparing expected response data with actual response data: response-only and stimulus-response. In the response-only mode, the device does not drive any data, it only acquires and compares. A generation session downloads the expected waveform. In the stimulus-response mode, the device drives and compares data in the same session. You can download waveforms with drive and/or compare data. All digital states supported in this mode.

#### **Response-Only Mode**

• NI 655x Only In Response-Only mode, the device does not drive any data; it only acquires data and compares the acquired data against an expected waveform. A generation session is exclusively used to download expected data.



* Generation session required to preload DUT with expected response data.

#### **Stimulus-Response Mode**

• NI 655x Only In Stimulus-Response mode, the device drives and compares data in the same operation. You can download waveforms with drive and/or compare data. This mode support all <u>digital states</u>, which means that the waveform contains both generation and expected data.



#### **Comparing Response Data in Software**

During a software comparison application, the tester generates the stimulus data, captures the actual response data, and then performs analysis of the response data after it is stored in the host PC memory. The actual response data analysis is performed entirely in software and not in real-time.

The following steps describe software comparison in more detail.

- 1. Enter original test data or read the data from a file. The test data may contain both stimulus and response data.
- 2. The tester extracts stimulus data from the test data. 1s and 0s in the test data specify stimulus data; all other characters indicate that no data is generated, so the voltage drivers are disabled.
- 3. The digital tester generates stimulus data onto the channel and acquires the response data. The generation and acquisition operations occur in parallel. During this step, <u>reducing the round-trip delay</u> can be important.
- 4. Once the generation and acquisition are complete, the application program performs the comparison on a per bit basis in software. The final pass/fail decision is only affected by the response data acquired when an H or L was present in the original test data.

To perform software comparison, you must transfer all data to the host computer for post-processing, which makes this method suitable for slower-speed applications. Transferring all the data to the host computer may exceed computer bandwidth limitations if more data is to be acquired than can fit on the testers onboard memory. For this situation and other cases that require faster comparison rates, <u>real-time hardware comparison</u> may be used.

#### **Comparing Response Data in Hardware**

Leveraging the onboard FPGA to compare the acquired response data to the expected data drastically increases the speed and reduces the necessary post-processing data analysis of stimulus-response applications. To develop a program to compare the response data in hardware, complete the following steps:

- 1. The original test data is entered by a user or read from a file. The test data may contain both stimulus and response data.
- 2. Stimulus data is extracted from the test data. 1s and 0s in the test data specify stimulus data; all other characters indicate that no data is generated, so the voltage drivers are disabled.
- 3. Use the <u>Hardware Compare Mode</u> property or the <u>NIHSDIO_ATTR_HWC_HARDWARE_COMPARE_MODE</u> attribute to enable the hardware comparison block on the NI 655*x* during the configuration stage of both the generation and acquisition sessions. After you enable the hardware comparison, a digital waveform created with the <u>six logic states</u> dictate the NI 655*x* operations—acquisition, generation, and hardware comparison—which eliminates the need for any parsing and software analysis functions.
- 4. The stimulus data is generated onto the channel by the digital tester, and the response data is acquired. The generation and acquisition operations occur in parallel, so <u>reducing the round-trip</u> <u>delay</u> can be important.
- 5. Once the generation and acquisition are complete, the application program performs the comparison on a per bit basis in software. The final pass/fail decision is only effected by the response data that was acquired when an H or L was present in the original test data.
- 6. For applications requiring more complex fault analysis, a fetch function can acquire the faulty data and any samples surrounding that error. For every sample that is in error, you can retrieve the following information:
  - Sample number of the fault
  - Channel(s) at fault
  - Total number of repeated errors (useful if the Filter
<u>Repeated Sample Errors</u> property or the <u>NIHSDIO_ATTR_HWC_FILTER_REPEATED_SAMPLE_ERR(</u> attribute are enabled)

Use the <u>Sample Error Backlog</u> property or the <u>NIHSDIO_ATTR_HWC_SAMPLE_ERROR_BACKLOG</u> attribute to query how many errors can be returned using the <u>niHSDIO HWC Fetch Sample</u> <u>Errors (U32)</u> VI or the <u>niHSDIO_HWC_FetchSampleErrors</u> function. Use the <u>Number Of Sample Errors</u> property or

<u>NIHSDIO_ATTR_HWC_NUM_SAMPLE_ERRORS</u> attribute, along with the <u>Samples Compared</u> property or the

NIHSDIO_ATTR_HWC_SAMPLES_COMPARED attribute, to calculate the sample error rate. By capturing this information, you can perform more detailed fault analysis.

The hardware performs all the data comparison on a per sample basis, which significantly reduces the time spent analyzing the data in software. Using this hardware comparison method, you can program the NI 655x for high-performance functional tests and other stimulus-response applications.

For a complete hardware compare example, refer to the "Hardware Compare - Fetch Error Records" example included with NI-HSDIO.

### **Hardware Comparison Functions**

You use the same VIs/functions as you normally would to configure your <u>acquisition</u> and <u>generation</u> sessions, but you can also use the following VI/function to fetch the fault data.

LabVIEW VIs	C Functions
niHSDIO HWC Fetch Sample Errors (U32)	niHSDIO_HWC_FetchSampleErrors

Note Some advanced attributes are not available through a configuration VI or function. Set the attributes directly using the <u>NI-HSDIO Property Node</u> VI or one of the niHSDIO SetAttribute functions. Refer to <u>Using Attributes with NI-HSDIO</u> for more information on attribute programming. Refer to <u>Advanced</u> <u>Attributes</u> for a list of those attributes configurable only through a property node or SetAttribute function.

## **Using Attributes with NI-HSDIO**

Attributes, which are typically called properties in LabVIEW, serve as a base for parameters.

NI-HSDIO contains high-level functions that set most of the instrument attributes. Some attributes are not accessible through the high-level functions. For example, input impedance is not set with any of the NI-HSDIO configuration functions. The values for these attributes must be set using the attribute.

#### **Accessing Attributes**

In LabVIEW, you can find attributes in the NI-HSDIO property node. To access them, complete the following steps:

- 1. Open a VI.
- Make sure that you are viewing the block diagram. Navigate to the NI-HSDIO palette at All Functions»NI Measurements»NI-HSDIO»Static and Dynamic Acquisition or All Functions»NI Measurements»NI-HSDIO»Static and Dynamic Generation, and then drag the property node icon to the block diagram.
- 3. Left-click the property node, and select the attribute you want to use.
- 4. To add additional attributes, resize the property node.

In C, attributes are accessed with the niHSDIO Set and Get Attribute functions. These functions correspond to a particular data type. For example, to set the input impedance, which has a data type or ViReal64, use <u>niHSDIO_SetAttributeViReal64</u>.

Refer to the Function Reference section for a complete listing of available <u>attributes</u> and <u>properties</u>.

## **Closing Your Session**

For any application you write, you must close the specified session to close communication with the device and free resources that it has reserved. If the session is running, it is first aborted.

You can close your session by using the following functions:

LabVIEW VI	C Function		
niHSDIO Close	niHSDIO_close		

Use these functions for both dynamic and static data operations.

Note To prevent generating unwanted signal glitches between initializing and closing sessions, no front panel terminals or channels are tristated by the close functions—they are all left driving whatever voltage they would have been driving had you simply used the <u>niHSDIO Abort</u> VI or the <u>niHSDIO_Abort</u> function. Use the <u>niHSDIO Reset</u> VI or the <u>niHSDIO_reset</u> function before using the close functions if you want to tristate your terminals and channels before closing your session.

## **Using the NI Digital Waveform Editor**

The NI Digital Waveform Editor (DWE) provides a simple way to create source data for your digital waveform generation application. Refer to <u>ni.com/catalog</u> for more information about how you can purchase this software.

Refer to the *NI Digital Waveform Editor Help* for detailed information about creating waveforms.



#### Features

Expand this book to view topics that explain how to configure certain features of the NI digital waveform generator/analyzer with NI-HSDIO.

## **Configuring Voltage Levels**

NI <u>654x/655x</u> devices support configuring <u>voltage levels</u> for use with a predefined <u>logic family</u> during a acquisition or generation session. If you are using an NI 655x device, you can also configure custom voltage levels. Refer to <u>Single-Ended Voltage Levels</u> for information about restrictions when configuring custom voltage levels.

You can configure these voltage levels using the following functions:

LabVIEW VIs	C Functions
Use one of the following	niHSDIO_ConfigureDataVoltageLogicFamily,
instances of the <u>niHSDIO</u>	niHSDIO_ConfigureDataVoltageCustomLevels,
Configure Voltage	niHSDIO_ConfigureEventVoltageLogicFamily,
polymorphic VI:	niHSDIO_ConfigureEventVoltageCustomLevels,
niHSDIO Configure	niHSDIO_ConfigureTriggerVoltageLogicFamily,
Data Voltage Logic	or
Family	niHSDIO_ConfigureTriggerVoltageCustomLevels
niHSDIO Configure	
Data Voltage	
Custom Levels	
niHSDIO Configure	
Event Voltage	
Logic Family	
niHSDIO Configure	
Event Voltage	
Custom Levels	
niHSDIO Configure	
Trigger Voltage	
Logic Family	
niHSDIO Configure	
Trigger Voltage	
Custom Levels	

## **Configuring Generation/Acquisition Frequencies**

You can configure the <u>generation or acquisition frequencies</u> of your device using the following functions:

LabVIEW VI	C Function		
niHSDIO Configure Sample Clock	niHSDIO_ConfigureSampleClock		

Additionally, you can export a clock using the <u>niHSDIO Export Signal</u> VI or the <u>niHSDIO_ExportSignal</u> function.

## **Configuring Data Interpretation**

You can configure the <u>data interpretation</u> for use during static or dynamic acquisition operations using the following functions:

LabVIEW VI	C Function
niHSDIO Configure Data	niHSDIO_ConfigureDataInterpretation
Interpretation	

## **Configuring Initial and Idle States**

You can configure <u>Initial and Idle States</u> for use during a <u>dynamic</u> <u>generation</u> operation using the following functions:

LabVIEW VIs	C Functions		
Initial Sta	ate		
Use one of the following instances of the niHSDIO Configure Initial State polymorphic VI:	<u>niHSDIO_ConfigureInitialState</u> or		
<ul> <li>niHSDIO Configure Initial State (String)</li> <li>niHSDIO Configure Initial State (U32)</li> </ul>	<u>niHSDIO_ConfigureInitialStateU32</u>		
Idle Sta	te		
Use one of the following instances of the <u>niHSDIO Configure Idle State</u> polymorphic VI:	<u>niHSDIO_ConfigureIdleState</u> or		
<ul> <li>niHSDIO Configure Idle State (String)</li> <li>niHSDIO Configure Idle State (U32)</li> </ul>	<u>niHSDIO_ConfigureIdleStateU32</u>		

## **Configuring Data Position**

You can configure the <u>data position</u> for acquisition or generation operations using the following functions:

LabVIEW VI	C Function
niHSDIO Configure Data Position	niHSDIO_ConfigureDataPosition
niHSDIO Configure Data Position Delay	niHSDIO_ConfigureDataPositionDelay

## **Configuring Input Impedance**

You can configure input impedance on particular channels by setting the following attributes:

LabVIEW Property	C Attribute				
Input Impedance	NIHSDIO_ATTR_INPUT_IMPEDANCE				

Note Configurable input impedance is not supported on <u>NI 654x/656x</u> devices.

Refer to <u>Using Attributes with NI-HSDIO</u> for more information on setting attribute-based configuration options.

# **Configuring PFI Terminal Configuration**

You can configure PFI 3 on the  $\underline{NI 656x}$  to be <u>single-ended</u> or <u>LVDS</u> terminals using the following properties/attributes:

LabVIEW Property	C Attribute
<u>Start Trigger</u> <u>Digital Edge</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGER_TERM
<u>Script</u> <u>Trigger</u> <u>Digital Edge</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGGER_TER
<u>Script</u> <u>Trigger</u> <u>Digital Level</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGGER_TEF
Reference Trigger Digital Edge Terminal Configuration	NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_TERMIN
<u>Pause</u> <u>Trigger</u> <u>Digital Level</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGGER_TER
Marker Event Terminal Configuration	NIHSDIO_ATTR_MARKER_EVENT_TERMINAL_CONFIGU
<u>Ready for</u> <u>Start Event</u> <u>Terminal</u>	NIHSDIO_ATTR_READY_FOR_START_EVENT_TERMINA

<b>Configuration</b>	
<u>Ready for</u> <u>Advance</u> <u>Event</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_READY_FOR_ADVANCE_EVENT_TERM
<u>Data Active</u> <u>Event</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO ATTR DATA ACTIVE EVENT TERMINAL CO
<u>End of</u> <u>Record</u> <u>Event</u> <u>Terminal</u> <u>Configuration</u>	<u>NIHSDIO_ATTR_END_OF_RECORD_EVENT_TERMINAL</u>
<u>Exported</u> <u>Start Trigger</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_EXPORTED_START_TRIGGER_TERMIN
<u>Exported</u> Advance Trigger Terminal Configuration	NIHSDIO_ATTR_EXPORTED_ADVANCE_TRIGGER_TERM
<u>Exported</u> <u>Reference</u> <u>Trigger</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO_ATTR_EXPORTED_REF_TRIGGER_TERMINAL
<u>Exported</u> <u>Script</u> <u>Trigger</u> <u>Terminal</u> <u>Configuration</u>	NIHSDIO ATTR EXPORTED SCRIPT TRIGGER TERMIN
<u>Exported</u> <u>Pause</u>	NIHSDIO_ATTR_EXPORTED_PAUSE_TRIGGER_TERMIN

<u>Trigger</u> <u>Terminal</u> <u>Configuration</u>

**Note** This feature is supported only on NI 656*x* devices.

Refer to <u>Using Attributes with NI-HSDIO</u> for more information on setting attribute-based configuration options.

## **Configuring a Data Rate Multiplier**

You can configure the <u>data rate multiplier</u> for acquisition or generation operations using the following property/attribute:

LabVIEW Property	C Attribute				
Data Rate Multiplier	<u>NIHSDIO</u>	ATTR	DATA	RATE	<u>MULTIPLIER</u>

## **Configuring Data Width**

You can configure the <u>data width</u> for acquisition operations using the following property/attribute:

LabVIEW Property	C Attribute			
Advanced:Data Width	<u>NIHSDIO</u>	ATTR	DATA	WIDTH

Valid values for data width vary by device.

NI 654 <i>x</i>	1, 2, 4
NI 655 <i>x</i>	1, 2, 4
NI 656 <i>x</i>	1, 2

# **Configuring Triggers**

You can configure triggers with NI-HSDIO using the following functions:

LabVIEW VIs	C Functions
Use one of the	niHSDIO_ConfigureDigitalEdgeStartTrigger
following instances of	niHSDIO_ConfigurePatternMatchStartTrigger
the niHSDIO	niHSDIO_ConfigurePatternMatchStartTriggerU32
Configure Trigger	niHSDIO_ConfigureSoftwareStartTrigger
polymorphic VI:	<u>niHSDIO_DisableStartTrigger</u>
<ul> <li>Digital Edge</li> </ul>	niHSDIO_ConfigureDigitalEdgeRefTrigger
Start Trigger	niHSDIO_ConfigurePatternMatchRefTrigger
Pattern Match	niHSDIO_ConfigurePatternMatchRefTriggerU32
Start Trigger	niHSDIO_ConfigureSoftwareRefTrigger
<ul> <li>Software Start</li> </ul>	<u>niHSDIO_DisableRefTrigger</u>
	<u>niHSDIO_ConfigureDigitalEdgeAdvanceTrigger</u>
- Disable Ctart	niHSDIO_ConfigurePatternMatchAdvanceTrigger
	niHSDIO_ConfigurePatternMatchAdvanceTriggerU32
rigger	niHSDIO_ConfigureSoftwareAdvanceTrigger
Digital Edge	<u>niHSDIO_DisableAdvanceTrigger</u>
Ref Trigger	<u>niHSDIO_ConfigureDigitalEdgeScriptTrigger</u>
<ul> <li>Pattern Match</li> </ul>	<u>niHSDIO_ConfigureDigitalLevelScriptTrigger</u>
Ref Trigger	<u>niHSDIO_ConfigureSoftwareScriptTrigger</u>
<ul> <li>Software Ref</li> </ul>	niHSDIO_DisableScriptTrigger
Trigger	niHSDIO_ConfigureDigitalLevelPauseTrigger
Disable Ref	<u>niHSDIO_ConfigurePatternMatchPauseTrigger</u>
Trigger	niHSDIO_ConfigurePatternMatchPauseTriggerU32
Digital Edge	<u>niHSDIO_DisablePauseTrigger</u>
Advance	
Trigger	
Dattorn Match	
Triggor	
• Somware	
Advance	
irigger	
Disable	
Advance	

<ul> <li>Trigger</li> <li>Digital Edge Script Trigger</li> <li>Digital Level Script Trigger</li> <li>Software Script Trigger</li> <li>Disable Script Trigger</li> <li>Digital Level Pause Trigger</li> <li>Pattern Match Pause Trigger</li> <li>Disable Pause Trigger</li> </ul>	
<u>niHSDIO Send</u> <u>Software Edge</u> <u>Trigger</u>	niHSDIO_SendSoftwareEdgeTrigger

## **Configuring Events**

You can configure events with NI-HSDIO using the following functions:

LabVIEW VIC FunctionsniHSDIO Export SignalniHSDIO_ExportSignal

## **Eliminating Round Trip Delay**

While performing a stimulus-response application, the time required for data to move from the digital tester, through the cable and DUT, and back to the tester is known as *round trip delay* (RTD).

One way to account for round-trip delay is by exporting a signal with an edge that is synchronous to the start of the stimulus data. This signal should be routed through equal lengths of cable to the acquisition Start trigger so that the signal has the same round trip delay as the data. The Data Active event can be used to accomplish this task, since it is synchronous to the start of the stimulus data.

The following figure illustrates the signal routing involved in elimination round trip delay.



For example, you can export the Data Active event on PFI 1 and route it to PFI 2, which you can configure as the acquisition Start trigger source. Then you can export the generation Sample clock to DDC CLK OUT and configure the acquisition Sample clock source as STROBE. Match your cable lengths so that the signals are routed with the same round trip delay as the data. This method ensures that clocks, control signals, and data signals all arrive at the device at the same time.

Another method to account for round-trip delay is to internally route a delayed version of your Data Active event to your acquisition Start trigger. To use this method, you must first know your total round-trip delay. Once known, set the Data Active Internal Route Delay property or the NIHSDIO_ATTR_DATA_ACTIVE_INTERNAL_ROUTE_DELAY attribute to that number of clock cycles. Then set the Data Position Delay property or the NIHSDIO_ATTR_DATA_POSITION_DELAY attribute to the desired fractional delay.

#### Reference

Expand this book for reference information about programming with NI-HSDIO.

#### LabVIEW Reference

This section describes the VIs and properties included with NI-HSDIO that you can use to configure and operate your digital waveform generator/analyzer.

### **VI Reference**

Use the VIs on the NI-HSDIO palette to build the block diagram. Click the icons for VI and function descriptions.



## **Dynamic and Static Acquisition Subpalette**

Use the VIs located on the **NI-HSDIO**»**Dynamic and Static Acquisition** palette to program acquisition operations.

Click the icons for VI and function descriptions.



#### niHSDIO Init Acquisition Session

Creates a new acquisition session. You can perform static and dynamic acquisition operations with this session.

Creating a new session does not automatically tristate your front panel terminals or channels possibly driving voltages from previous sessions. Refer to the <u>niHSDIO Close</u> VI for more information on leaving lines driving after closing a session.

Set **reset instrument** to TRUE to place your device in a known start-up state when creating a new session. This action is equivalent to using the <u>niHSDIO Reset</u> VI, and it tristates the front panel terminals and channels.



- **resource name** specifies the device name, for example "PXI1Slot3," where "PXI1Slot3" is a device name assigned by Measurement & Automation Explorer.
- **id query** specifies whether the driver performs an ID query on the instrument.

When **id query** is set to TRUE, the driver ensures compatibility between the instrument and the driver. When **id query** is set to FALSE, the driver skips the ID query.

**reset instrument** specifies whether the driver resets the device during initialization of the session. TRUE means that the device is reset; FALSE means that the device is not reset.

Refer to <u>niHSDIO Reset</u> for more information on what happens during an instrument reset.



**Note** Resetting your device resets the *entire* device. Acquisition or generation operations in progress will be aborted and cleared.

- option string is currently unused. Leave this parameter unwired.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The

### niHSDIO Assign Dynamic Channels

Configures channels for dynamic acquisition (if **instrument handle** is an acquisition session) or dynamic generation (if **instrument handle** is a generation session).



**Note** A channel cannot simultaneously be assigned to static generation and dynamic generation.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** identifies which channels are reserved for dynamic operation.

Examples of valid syntax for this control are 0:31 or 0-15,16-31 or 0-30,31. The order of these channels specifies the order when using the <u>waveform data type</u> VIs (<u>niHSDIO Read Waveform</u> and <u>niHSDIO Write Named Waveform</u>).

Leave **channel list** blank to specify all channels. Use "none" to unassign all channels.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an

#### niHSDIO Configure Sample Clock

Configures the Sample clock. This VI allows you to specify the clock source and rate for the Sample clock.

If **clock source** is set to **On Board Clock**, NI-HSDIO coerces the rate to a value that is supported by the hardware. Select **Timing»Sample Clock»Rate** from the <u>property node</u> to get the coerced value of the Sample clock rate. **clock source** can be set to **STROBE** only for acquisition sessions.

Refer to <u>Clocks for Digital Waveform Generator/Analyzers</u> for more information about the Sample clock sources. Refer to the Clocking book for your <u>device</u> for a block diagram of the clocking circuitry.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **clock source** specifies the Sample clock source. You can select **On Board Clock**, **ClkIn**, **PXI STAR Line**, or **Strobe** as the value for this control.
- **clock rate** specifies the Sample clock rate, expressed in Hz.

You must set this property even when you supply an external clock because NI-HSDIO uses this property for a number of reasons, including optimal error checking and certain pulse width selections.

- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

#### niHSDIO Configure Acquisition Size

Configures the size of the acquisition, including how many records are acquired and the minimum record size.



1/0	<b>instrument handle</b> identifies your instrument session. <b>instrument handle</b> was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.	
1321	<b>samples per record</b> sets the number of samples to acquire per record. If you need pretrigger and posttrigger points, configure a Reference trigger and specify the number of pretrigger points.	
132	number of records sets how many records are acquired.	
	error in describes error conditions that occur before this VI or function runs.	
	<b>status</b> is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.	
	<b>code</b> is the error or warning code. The default is 0. If <b>status</b> is TRUE, <b>code</b> is a negative error code. If <b>status</b> is FALSE, <b>code</b> is 0 or a warning code.	
	<b>source</b> identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.	
170	<b>instrument handle out</b> passes a reference to your instrument session to the next VI. <b>instrument handle</b> was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.	
	<b>error out</b> contains error information. If <b>error in</b> indicates that an error occurred before this VI or function ran, <b>error out</b> contains the same error information. Otherwise, it describes the error status that this VI or function produces.	

- **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
- **code** is the error or warning code. If **status** is TRUE, **code** is a popyore error code. If **status** is EALSE **code** is 0 or a

#### niHSDIO Read Waveform

Initiates a waveform acquisition on channels enabled for dynamic acquisition, waits to acquire the number of samples specified in **samples to read**, and returns the acquired data.

Use the pull-down menu to select an instance of this VI.

Select an instance 🔽

This instance of niHSDIO Read Waveform returns the binary representation of the acquired waveform.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

This instance of niHSDIO Read Waveform returns the acquired waveform as an array of unsigned 16-bit data.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.

**samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

This instance of niHSDIO Read Waveform returns the acquired waveform as an array of unsigned 8-bit data.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

#### WDT

This instance of niHSDIO Read Waveform returns the acquired waveform data with the waveform data type.



- timestamp type specifies whether the timestamp for the waveform data is relative or absolute.
- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.

This instance of niHSDIO Read Waveform returns the acquired waveform as a two-dimensional array of unsigned 32-bit data.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error,

This instance of niHSDIO Read Waveform returns the acquired waveform as a two-dimensional array of unsigned 16-bit data.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error,
2D U8

This instance of niHSDIO Read Waveform returns the acquired waveform as a two-dimensional array of unsigned 8-bit data.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

If samples to read is greater than the number of samples in the device memory and all the available samples are acquired before a timeout, NI-HSDIO returns the available samples.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error,

# 1D WDT

This instance of niHSDIO Read Waveform returns the acquired waveform as an array of waveform data types.



- **timestamp type** specifies whether the timestamp for the waveform data is relative or absolute.
- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

If samples to read is greater than the number of samples in the device memory and all the available samples are acquired before a timeout, NI-HSDIO returns the available samples.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.

# niHSDIO Close

Closes the specified session and frees resources that it has reserved. If the session is running, it is first aborted.

To prevent generating unwanted signal glitches between initializing and closing sessions, no front panel terminals or channels are tristated by the niHSDIO Close VI. The terminals and channels are all left driving whatever voltage they would have been driving had you used the niHSDIO Abort VI. Use the niHSDIO Reset VI before niHSDIO Close if you want to tristate all terminals and channels before closing your session.

instrument handle	
error in (no error)	

- I/0 | **instrument handle** identifies your instrument session. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error in describes error conditions that occur before this VI or function runs.
  - TF status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - abc **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran. error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - TF status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - 132 code is the error or warning code. If status is TRUE, code is a nonzero error code. If status is FALSE, code is 0 or a warning code.



Recurse identifies where and why an error occurred. The

# **Acquisition Configuration Subpalette**

Use the VIs located on the **NI-HSDIO**»**Dynamic and Static Acquisition**»**Acquisition Configuration** palette to configure acquisition operations.

Click the icons for VI and function descriptions.



# niHSDIO Configure Trigger

Configures the specified trigger.

Refer to <u>Triggers</u> for more information about the triggers that are available with your NI digital waveform generator/analyzer.

Use the pull-down menu to select an instance of this VI.

Select an instance

-

# Digital Edge Start Trigger

This instance of niHSDIO Configure Trigger configures the Start trigger for digital edge triggering.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- source specifies the trigger source. You can choose PFI <0..3>, PXI Trigger Line/RTSI <0..7>, or PXI STAR Line.
- edge specifies the edge to detect. You can choose **Rising Edge** or **Falling Edge**.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a

# Pattern Match Start Trigger

This instance of niHSDIO Configure Trigger configures the Start trigger for pattern match triggering. This VI is only valid for acquisition sessions.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.

- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

This string expression creates a mask for the pattern. This expression is composed of characters:

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

The first character in the expression corresponds to the first channel in **channel list**. The number of characters in pattern must correspond to the number of channels specified in **channel list** or an error is returned.

For example, the following two examples are valid and do the same thing:

- channel list = "19-0" and pattern = "0000 0XXX XX11 1111 1111"
- channel list = "0-19" and pattern = "1111 1111 11XX XXX0 0000"
- **trigger when** specifies the conditions under which the trigger is sent.

**Pattern Matches** means that the trigger is sent when the sampled pattern matches the pattern specified in **pattern**. **Pattern Does Not Match** means that the trigger is sent when the

# Pattern Match Start Trigger (U32)

This instance of niHSDIO Configure Trigger configures the Start trigger for pattern match triggering. The pattern you specify in this instance only represents logic high and logic low. If you require more choices for your pattern, use the niHSDIO Configure Pattern Match Start Trigger instance of this VI.

This VI is only valid for acquisition sessions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

Bits on channels not specified in **channel list** are ignored.

**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.
Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is EALSE, **code** is 0 or a warning code.

# Software Start Trigger

This instance of niHSDIO Configure Trigger configures the Start trigger for software triggering. Use the niHSDIO Send Software Edge Triggering VI to assert the trigger condition.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the

# **Disable Start Trigger**

This instance of niHSDIO Configure Trigger configures the instrument to not wait for a Start trigger after the niHSDIO Initiate VI is used. This VI is only necessary if you have configured a Start trigger and now want to disable it.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the

# Digital Edge Ref Trigger

This instance of niHSDIO Configure Trigger configures the Reference trigger for digital edge triggering. If the Reference trigger is asserted before the required number of pretrigger samples are acquired, it is ignored. This VI is valid only for acquisition sessions.



- **pretrig samples** specifies the number of pretrigger samples the device must receive before the Reference trigger is acknowledged.
- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **source** specifies the trigger source. You can choose **PFI <0..3>**, **PXI Trigger Line/RTSI <0..7>**, or **PXI STAR Line**.
- edge specifies the edge to detect. You can choose Rising Edge or Falling Edge.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

# Pattern Match Ref Trigger

This instance of niHSDIO Configure Trigger configures the Reference trigger for pattern match triggering. If the Reference trigger asserts before the required number of pretrigger samples are acquired, it is ignored. This VI is only valid for acquisition sessions.



**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- **pretrig samples** specifies the number of pretrigger samples the device must receive before the Reference trigger is acknowledged.
- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

This string expression creates a mask for the pattern. This expression is composed of characters:

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

# Pattern Match Ref Trigger (U32)

This instance of niHSDIO Configure Trigger configures the Reference trigger for pattern match triggering. The pattern you specify in this instance only represents logic high and logic low. If you require more choices for your pattern, use the niHSDIO Configure Pattern Match Ref Trigger instance of this VI.

If the Reference trigger asserts before the required number of pretrigger samples are acquired, it is ignored. This VI is only valid for acquisition sessions.



**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- **pretrig samples** specifies the number of pretrigger samples the device must receive before the Reference trigger is acknowledged.
- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

Bits on channels not specified in **channel list** are ignored.

error in describes error conditions that occur before this VI or function runs.

**status** is TRUE (X) if an error occurred before this VI or

# Software Ref Trigger

This instance of niHSDIO Configure Trigger configures the Start trigger for software triggering. If the Reference trigger asserts before the required number of pretrigger samples are acquired, it is ignored. Use the niHSDIO Send Software Edge Triggering VI to assert the trigger condition. This VI is valid only for acquisition sessions.



- **pretrig samples** specifies the number of pretrigger samples the device must receive before the Reference trigger is acknowledged.
- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **Figure** code is the error or warning code. If status is TPUE code

# Disable Ref Trigger

This instance of niHSDIO Configure Trigger configures the acquisition to have no Reference trigger. This VI is only necessary if you have configured a Reference trigger and now want to disable it. This VI is valid only for acquisition sessions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the

# Digital Edge Advance Trigger

This instance of niHSDIO Configure Trigger configures the Advance trigger for digital edge triggering. This VI is valid only for acquisition sessions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **source** specifies the trigger source. You can choose **PFI <0..3>**, **PXI Trigger Line/RTSI <0..7>**, or **PXI STAR Line**.
- edge specifies the edge to detect. You can choose **Rising Edge** or **Falling Edge**.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code**

# Pattern Match Advance Trigger

This instance of niHSDIO Configure Trigger configures the Advance trigger for pattern match triggering. This VI is only valid for acquisition sessions.



**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.

**pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

This string expression creates a mask for the pattern. This expression is composed of characters:

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

The first character in the expression corresponds to the first channel in **channel list**. The number of characters in pattern must correspond to the number of channels specified in **channel list** or an error is returned.

For example, the following two examples are valid and do the

## Pattern Match Advance Trigger (U32)

This instance of niHSDIO Configure Trigger configures the Advance trigger for pattern match triggering. The pattern you specify in this instance only represents logic high and logic low. If you require more choices for your pattern, use the niHSDIO Configure Pattern Match Advance Trigger instance of this VI.

This VI is only valid for acquisition sessions.



**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.
Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

Bits on channels not specified in **channel list** are ignored.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is EALSE, **code** is 0 or a warning code.

#### Software Advance Trigger

This instance of niHSDIO Configure Trigger configures the Advance trigger for software triggering. Use the niHSDIO Send Software Edge Triggering VI to assert the trigger condition. This VI is valid only for acquisition sessions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the

## **Disable Advance Trigger**

This instance of niHSDIO Configure Trigger configures the acquisition to not use an Advance trigger. This VI is only necessary if you have configured an Advance trigger and now want to disable it.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the

# Digital Edge Script Trigger

This instance of niHSDIO Configure Trigger configures the Script trigger for digital edge triggering. This VI is valid only for generation sessions.



**trigger id** specifies the instance of the Script trigger.

You can choose Script Trigger 0, Script Trigger 1, Script Trigger 2, or Script Trigger 3.

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- source specifies the trigger source. You can choose PFI <0..3>, PXI Trigger Line/RTSI <0..7>, or PXI STAR Line.
- edge specifies the edge to detect. You can choose Rising Edge or Falling Edge.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE

# **Digital Level Script Trigger**

This instance of niHSDIO Configure Trigger configures a Script trigger for level triggering. This VI is valid only for generation sessions.



**trigger id** specifies the instance of the Script trigger.

You can choose Script Trigger 0, Script Trigger 1, Script Trigger 2, or Script Trigger 3.

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **Source** specifies the trigger source. You can choose **PFI <0..3**>, **PXI Trigger Line/RTSI <0..7**>, or **PXI STAR Line**.
- **trigger when** specifies the active level for the desired trigger.

trigger when can be either of the following:

- High Level: Trigger is active while its source is high
- Low Level: Trigger is active while its source is low
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains

## Software Script Trigger

This instance of niHSDIO Configure Trigger configures the Script trigger for software triggering. Use the niHSDIO Send Software Edge Triggering VI to assert the trigger condition. This VI is valid only for generation sessions.



**trigger id** specifies the instance of the Script trigger.

You can choose Script Trigger 0, Script Trigger 1, Script Trigger 2, or Script Trigger 3.

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code**

## **Disable Script Trigger**

This instance of niHSDIO Configure Trigger disables a Script trigger. This VI is only necessary if you have configured a Script trigger and now want to disable it. This VI is valid only for generation sessions.



**trigger id** specifies the instance of the Script trigger.

You can choose Script Trigger 0, Script Trigger 1, Script Trigger 2, or Script Trigger 3.

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a

# **Digital Level Pause Trigger**

This instance of niHSDIO Configure Trigger configures the Pause trigger as a level trigger.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **source** specifies the trigger source. You can choose **PFI <0..3**>, **PXI Trigger Line/RTSI <0..7**>, or **PXI STAR Line**.

#### **trigger when** specifies the active level for the desired trigger.

trigger when can be either of the following:

- High Level: Trigger is active while its source is high
- Low Level: Trigger is active while its source is low
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

# Pattern Match Pause Trigger

This instance of niHSDIO Configure Trigger configures the Pause trigger for pattern match triggering. This VI is only valid for acquisition sessions.



**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.
Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

This string expression creates a mask for the pattern. This expression is composed of characters:

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

The first character in the expression corresponds to the first channel in **channel list**. The number of characters in pattern must correspond to the number of channels specified in **channel list** or an error is returned.

For example, the following two examples are valid and do the same thing:

# Pattern Match Pause Trigger (U32)

This instance of niHSDIO Configure Trigger configures the Pause trigger for pattern match triggering. The pattern you specify in this instance only represents logic high and logic low. If you require more choices for your pattern, use the niHSDIO Configure Pattern Match Start Trigger instance of this VI.

This VI is only valid for acquisition sessions.



**trigger when** specifies the conditions under which the trigger is sent.

Pattern Matches means that the trigger is sent when the sampled pattern matches the pattern specified in pattern.
Pattern Does Not Match means that the trigger is sent when the sampled pattern does not match the pattern specified in pattern.

Refer to <u>Pattern Match Trigger</u> for an illustration of when your data is acquired when using a pattern match trigger for acquisitions.

- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies which channels are configured for pattern matching using the pattern string. The order of channels in the list determines the order of the pattern string.
- **pattern** specifies the binary pattern that activates the pattern match trigger under the conditions specified in **trigger when**.

Bits on channels not specified in **channel list** are ignored.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is EALSE, **code** is 0 or a warning code.

# **Disable Pause Trigger**

This instance of niHSDIO Configure Trigger configures the data operation to have no Pause trigger. This VI is only necessary if you configured the Pause trigger and now want to disable it.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the

# niHSDIO Export Signal

Routes signals (clocks, triggers, and events) to the output terminal you specify. Refer to your device documentation for valid signal destinations.

Any routes created within a session persist after the session closes to prevent signal glitching. To unconfigure signal routes created in previous sessions, set the **reset instrument** parameter in the <u>niHSDIO Init</u> <u>Generation Session</u> VI or the <u>niHSDIO Init Acquisition Session</u> VI to TRUE or use <u>niHSDIO Reset Device</u>. <u>Details</u>.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- signal specifies the signal to export. You can select Sample Clock, Reference Clock, Start Trigger, Reference Trigger, Advance Trigger, Pause Trigger, Script Trigger, Data Active Event, Ready for Start Event, Ready for Advance Event, Marker Event, End of Record Event, or Onboard Ref Clock.

Selecting **Script Trigger** or **Marker Event** requires a **signal identifier** to describe the particular signal.

**signal identifier** describes the signal being exported.

You can select Script Trigger 0, Script Trigger 1, Script Trigger 2, Script Trigger 3, Marker 0, Marker 1, Marker 2, Marker 3, or None.

- output terminal specifies the terminal where the signal will be exported. You can choose Do not export signal, PFI <0..3>, PXI Trigger Line/RTSI <0..7>, ClkOut on SMB Connector, or ClkOut on Digital Data & Control connector.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is

#### niHSDIO Export Signal Details

If you export a signal with this VI and <u>commit</u> the session, the signal is routed to the output terminal you specify. If you then reconfigure the signal to have a different output terminal, the previous output terminal is tristated after the session is committed. If you change **output terminal** to **Do Not Export** at <u>commit</u>, the previous output terminal is tristated.

# niHSDIO Configure Voltage

Configures the voltage levels for the data, trigger, and event channels. You can use either predefined logic families or custom voltage levels.

For more information on the voltage options available with your device, refer to <u>Voltage Levels</u> and <u>Logic Families</u>.



Note NI 656x devices do not support configuring voltage levels. NI-HSDIO returns an error if you use this VI when programming those devices.

Use the pull-down menu to select an instance of this VI.

•

Select an instance

#### Data Voltage Logic Family

This instance of niHSDIO Configure Voltage configures the voltage levels for the data channels using a logic family. Refer to the device documentation for descriptions of logic families and possible voltage restrictions.

channel list (all:"")	
instrument handle	··········· instrument handle out
logic family (3.3V logic) -	error out
error in (no error)	entor odc

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **Iogic family** identifies the settings for the instrument generation and acquisition operations. You can choose 5.0 V, 3.3 V, 2.5 V, or 1.8 V Logic for the logic family.
- **channel list** identifies which channels to apply settings.

Leave channel list blank to apply to all channels.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **Status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

# Data Voltage Custom Levels

This instance of niHSDIO Configure Voltage configures the voltage levels of the data channels using the specified high and low levels. Refer to the device documentation for possible voltage restrictions.



**Note** If you are using an NI 654*x* device for generation sessions, you must set **high level** to the appropriate logic family value, and you must set **low level** to 0. For acquisition sessions with the NI 654*x*, select the same value for **high level** and **low level** from the following list: 0.9 V, 1.25 V, or 1.65 V.



- instrument handle identifies your instrument session.
   instrument handle was obtained from the <u>niHSDIO Init</u>
   Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **high level** specifies what voltage identifies high level.
- **channel list** identifies which channels to apply settings.

Leave channel list blank to apply to all channels.

- **Iow level** specifies what voltage identifies low level.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an

# Trigger Voltage Logic Family

This instance of niHSDIO Configure Voltage configures the voltage levels for the trigger channels using a logic family. Refer to the device documentation for descriptions of logic families and possible voltage restrictions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **Iogic family** identifies the settings for the instrument generation and acquisition operations. You can choose 5.0 V, 3.3 V, 2.5 V, or 1.8 V Logic for the logic family.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a

# **Trigger Voltage Custom Levels**

This instance of niHSDIO Configure Voltage configures the voltage levels of the trigger channels using user-defined high and low levels. Refer to the device documentation for possible voltage restrictions.



**Note** If you are using an NI 654*x* device for generation sessions, you must set **high level** to the appropriate logic family value, and you must set **low level** to 0. For acquisition sessions with the NI 654*x*, select the same value for **high level** and **low level** from the following list: 0.9 V, 1.25 V, or 1.65 V.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **high level** specifies what voltage identifies high level.
- **Iow level** specifies what voltage identifies low level.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or EALSE

# **Event Voltage Logic Family**

This instance of niHSDIO Configure Voltage configures the voltage levels for the event channels using a logic family. Refer to the device documentation for descriptions of logic families and possible voltage restrictions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- Iogic family identifies the settings for the instrument generation and acquisition operations. You can choose 5.0 V, 3.3 V, 2.5 V, or 1.8 V Logic for the logic family.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a
#### **Event Voltage Custom Levels**

This instance of niHSDIO Configure Voltage configures the voltage levels of the event channels using user-defined high and low levels. Refer to the device documentation for possible voltage restrictions.



**Note** If you are using an NI 654*x* device for generation sessions, you must set **high level** to the appropriate logic family value, and you must set **low level** to 0. For acquisition sessions with the NI 654*x*, select the same value for **high level** and **low level** from the following list: 0.9 V, 1.25 V, or 1.65 V.



- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **high level** specifies what voltage identifies high level.
- **Iow level** specifies what voltage identifies low level.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **Status** is TRUE (X) if an error occurred or EALSE

# niHSDIO Configure Data Interpretation

Use this VI to select between acquiring high/low data or valid/invalid data during a static or dynamic acquisition operation.

Select **High or Low** to get logic high or logic low values. Select **Valid or Invalid** to tell if the signal is within the specified voltage range, (above Data Voltage Low Level but below Data Voltage High Level) or outside the range (below Data Voltage Low Level or above Data Voltage High Level).

Note NI 654x/656x devices only support the high/low mode of data interpretation. NI-HSDIO returns an error if you select valid/invalid mode for an acquisition with these devices.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** identifies which channels to apply settings.

Leave **channel list** blank to apply to all channels.

**data interpretation** specifies the value for data interpretation.

**High or Low** means that the data read represents logical values (logic high or logic low).

Valid or Invalid means that the data read represents whether channel data is within a specified voltage range.

- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

# niHSDIO Configure Data Position

Configures the various ways the data is clocked relative to the Sample clock.

Refer to the Acquisition and Generation books for your <u>device</u> for timing diagrams that illustrate the effects of <u>data position</u> changes.



N

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** identifies which channels to apply settings.

Leave **channel list** blank to apply to all channels.

**position** specifies which edge of the Sample clock signal is used to time the operation. You can also configure the device to generate data at a configurable delay past each rising edge of the Sample clock.

You can choose Sample Clock Rising Edge, Sample Clock Falling Edge, or Delay from Rising Edge of Sample Clock. If you choose Delay from Rising Edge of Sample Clock, specify the delay using the delay parameter of the <u>Configure Data</u> <u>Position Delay</u> VI.

**Notes** The **Delay from Sample Clock Rising Edge** setting has more <u>jitter</u> than the rising or falling edge values.

Certain devices have sample clock frequency limitations when a custom delay is used. Refer to the device documentation for details.

To configure a delay on NI 656*x* devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **1321** code is the error or warning code. The default is 0. If

# niHSDIO Configure Data Position Delay

Configures the data position delay with respect to the Sample clock. To configure the data delay, select **Delay from Rising Edge of Sample Clock** as the value for the **position** parameter of the <u>niHSDIO</u> <u>Configure Data Position</u> VI.



**Note** To configure a delay on NI 656*x* devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.

channel list (all:"")
instrument handle
delay (0.000)
error in (no error)

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** identifies which channels to apply settings.

Leave **channel list** blank to apply to all channels.

**delay** specifies the delay after the Sample clock rising edge when the device generates or acquires a new data sample. Data delay is expressed as a fraction of the clock period.

All the channels in the session that use delayed sample clock to position data must have the same delay value.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init

# **Advanced Timing Subpalette**

Use the VIs located on the NI-HSDIO»Dynamic and Static Acquisition» Acquisition Configuration»Adv Timing palette to configure advanced timing properties.

Click the icons for VI and function descriptions.



# niHSDIO Configure Ref Clock

Configures the Reference clock. Use this VI when you are using the On Board Clock as a Sample clock, and you want the Sample clock to be phase-locked to a reference signal. Phase-locking the Sample clock to a Reference clock prevents the Sample clock from "drifting" relative to the Reference clock.

Refer to <u>Clocks for Digital Waveform Generator/Analyzers</u> for more information about the Reference clock.



- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- source specifies the <u>PLL Reference clock source</u>. You can choose None, ClkIn, PXI Clock 10 Line, or RTSI 7.

Set **clock source** to **None** if you do not want to phase lock the onboard clock with a Reference clock.

- **clock rate** specifies the reference clock rate, expressed in hertz.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains

# niHSDIO Adjust Sample Clock Relative Delay

Delays the Sample clock relative to the Reference clock. Use this VI to align the Sample clock of your device with the Sample clock of another device in your system. The adjustment takes effect immediately. Only call this VI after your session is committed.

This function returns an error if the **clock source** parameter of the <u>niHSDIO Configure Ref Clock</u> VI is set to **none**.

This VI can only align the device Sample clock to another sample clock if the other device is using the same reference clock source.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- adjustment time specifies the time, in seconds, by which you want to delay the Sample clock. Values range between 0 and the Sample clock period (1/Sample Clock Rate).
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

## niHSDIO Tristate Channels

Forces a channel into a high-impedance state. The effect is immediate; it does not require the session be committed. The channel remains tristated regardless of what other software commands are called. Call this VI again and wire FALSE to the **tristate** terminal to allow other software commands to control the channel normally.

Channels stay tristated while the session remains open. Closing the session does not affect the high-impedance state of the channel, but future sessions can now control it.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** identifies which channels will be tristated. Channels not specified in this list are unaffected.

Syntax examples: "2-15" or "0-3, 5, 8-15", or "0, 3, 10."

- **TFI tristate** specifies whether the channels specified in **channel list** remain tristated. If **tristate** is TRUE, the channels specified in **channel list** remain tristated, ignoring future software commands. If **tristate** is FALSE, the channels specified in **channel list** can have the tristate condition removed by future software commands.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- **instrument handle out** passes a reference to your instrument

# **Advanced Acquisition Control Subpalette**

Use the VIs located on the **NI-HSDIO**»Dynamic and Static Acquisition»Advanced Acquisition Control palette to program advanced acquisition operations.

Click the icons for VI and function descriptions.



# niHSDIO Initiate

Commits any pending attributes to hardware and starts the dynamic operation.

Refer to the niHSDIO Commit VI for more information on committing.

For a generation operation with a configured <u>Start trigger</u>, the niHSDIO Initiate VI causes the channels to go to their <u>Initial states</u>. Refer to the <u>niHSDIO Configure Initial State</u> VI for more information on Initial states.

This function is valid for only dynamic operations (acquisition or generation). It is not valid for static operations.

instrument handle NIHSDIO ~ instrument handle out -11 error in (no error) error out

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

# niHSDIO Abort

Stops a running dynamic session. This VI is generally not required on finite data operations, as these operations complete once the last data point is generated or acquired. This VI is generally only required for continuous operations or if you wish to interrupt an incomplete finite operation.

This VI is valid only for dynamic operations (acquisition or generation). It is not valid for static operations.

 $\overline{\mathbb{N}}$ 

**Note** To avoid receiving hardware clocking errors when reconfiguring an external clock, you should explicitly call the niHSDIO Abort VI after your finite operation has completed before performing any clocking reconfiguration. An external clock that stops sending pulses to the device (even after a finite operation has completed) may cause NI-HSDIO to think the clock became unlocked if the device has not implicitly aborted yet.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an

# niHSDIO Wait Until Done

Pauses execution of your program until the dynamic data operation is completed or the VI returns a timeout error. niHSDIO Wait Until Done is a blocking VI that periodically checks the operation status. This VI returns control to the calling program if the operation completes successfully or if an error occurs (including a timeout error).

Use this VI for finite data operations that you expect to complete within a certain time.

instrument handle	www.willson	
max time milliseconds (10 s		
error in (no error)	error out	

- 1/0 **instrument handle** identifies your instrument session. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- 132 max time milliseconds specifies the number of milliseconds to allow the function to complete before returning. If the specified time elapses before the data operation has completed, the function returns a timeout error.

Setting a value of 0 causes the function to return immediately. This setting can be useful to manually poll for hardware errors after a data operation has been initiated. If no other error has occurred and the data operation is still not complete, the function returns a timeout error.

Setting a value of -1 causes the function to never timeout. Be careful not to use this value during a continuous operation, as it never returns unless a hardware error occurs. Perform a manual device reset from Measurement & Automation Explorer if you get stuck in this state or use the niHSDIO Reset VI or niHSDIO Reset Device from the other session of the device.

- 200 error in describes error conditions that occur before this VI or function runs.
  - TFI status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - 132 **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.



the course identifies where an error accurred. The course

# niHSDIO Fetch Waveform

Transfers acquired waveform data from device memory to PC memory. The data was acquired to <u>onboard memory</u> previously by the hardware after it was initiated.

If the number of samples specified in **samples to read** is not available after the time duration specified in **max time milliseconds**, this VI returns no data with a timeout error.

The fetch position can be modified by selecting the appropriate **Fetch Relative To** and **Fetch Offset** properties in the <u>property node</u>. The default Fetch Relative To value is **Current Read Position**. The default Offset value is 0.

The niHSDIO Fetch Waveform (U32) VI is not necessary if you use the niHSDIO Read Waveform VI, as the fetch is performed as part of that function.

Use the pull-down menu to select an instance of this VI.

Select an instance 🔽

This instance of niHSDIO Fetch Waveform fetches the data as an array of unsigned 32-bit integers and returns the number of samples read.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.

**samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

This instance of niHSDIO Fetch Waveform fetches the data as an array of unsigned 16-bit integers and returns the number of samples read.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.

**samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

This instance of niHSDIO Fetch Waveform fetches the data as an array of unsigned 8-bit integers and returns the number of samples read.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.

**samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

#### WDT

This instance of niHSDIO Fetch Waveform fetches the data using the waveform data type and returns the number of samples read.



- **timestamp type** specifies whether the timestamp for the waveform data is relative or absolute.
- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.

This instance of niHSDIO Fetch Waveform fetches the data as a twodimensional array of unsigned 32-bit integers and returns the number of samples read.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source

This instance of niHSDIO Fetch Waveform fetches the data as a twodimensional array of unsigned 16-bit integers and returns the number of samples read.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source

This instance of niHSDIO Fetch Waveform fetches the data as a twodimensional array of unsigned 8-bit integers and returns the number of samples read.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source

#### 1D WDT

This instance of niHSDIO Fetch Waveform fetches the data as an array of waveform data type data and returns the number of samples read.



- **timestamp type** specifies whether the timestamp for the waveform data is relative or absolute.
- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.

#### **Direct DMA**

Transfers acquired waveform data from device memory directly to PC memory allocated by a Direct DMA-compatible device. The size of the sample that is transferred is determined by the the data width for your device.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples to read** specifies the number of samples to fetch.

If you specify a value for **samples to read** that is greater than the number of samples in the device memory, NI-HSDIO returns the samples that are acquired after **max time milliseconds**. Setting this parameter to -1 acquires the **samples per record** specified in <u>niHSDIO Configure Acquisition Size</u>.

**max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

For single record operations, if you set this parameter to 0, the VI returns immediately with up to the number of samples specified in **samples to read**. Setting this parameter to 0 is not supported for multirecord acquisitions.

- **buffer address** specifies the location for the buffer in memory at which to transfer acquired data.
- **buffer size** specifies the size (in bytes) of the buffer in memory at which to transfer acquired data.
- record to fetch specifies the record number to fetch. The records are zero-indexed.
- error in describes error conditions that occur before this VI or function runs.

# niHSDIO HWC Fetch Sample Errors (U32)

Returns the sample error information from a hardware comparison operation.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **samples errors to read** specifies the number of sample errors to fetch.
- **max time milliseconds** specifies in milliseconds the time allotted for the VI to complete before returning a timeout error.

If you set the value to -1, the VI never times out.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- **sample numbers** returns the number of samples with errors.
- **error bits** returns the bit numbers that were in error for each sample that has an error.
- **number of sample errors read** returns the total number of sample errors read from device memory.
- [132] repeat count returns the number of times the error was repeated

# **Static Acquisition Subpalette**

Use the VIs located on the **NI-HSDIO**»**Dynamic and Static Acquisition**»**Static Acquisition** palette to program static acquisition operations.

Click the icons for VI and function descriptions.



# niHSDIO Assign Static Channels

Configures channels for <u>static acquisition</u> (if **instrument handle** is an acquisition session) or for <u>static generation</u> (if **instrument handle** is a generation session).

A channel cannot simultaneously be assigned to static generation and dynamic generation.



- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** identifies which channels will be configured as static.

Examples of valid syntax for this control are 0:31 or 0-15,16-31 or 0-30,31. Leave **channel list** blank to specify all channels. Use "none" to unassign all channels.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

# niHSDIO Read Static (U32)

Immediately reads the digital value on channels configured for static acquisition.

You can configure a channel for <u>static acquisition</u> using the <u>niHSDIO</u> <u>Assign Static Channels</u> VI. Channels not configured for static acquisition return a zero.

Values obtained from static read operations are affected by the **data interpretation** parameter of the <u>niHSDIO Configure Data</u> <u>Interpretation</u> VI.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- **read data** is the bit-value of data read from channels configured for static acquisition.

The least significant bit of **read data** corresponds to the lowest physical channel number. For example, if **read data** returns a value of 0x00F0, channels 4 -7 are logic one and the remaining channels are logic zero or are not configured for static acquisition.

**error out** contains error information. If **error in** indicates that an

# **Utility Subpalette**

Use the VIs located on the **NI-HSDIO»Dynamic and Static Acquisition»Utility** palette to access utility features of NI-HSDIO.

Click the icons for VI and function descriptions.



### niHSDIO Is Done

Checks the hardware to determine if the device completed the dynamic data operation or if any errors have occurred. You can also use this VI for continuous dynamic data operations to poll for error conditions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- done returns the state of the completed data operation. The VI returns TRUE if the data operation is complete or an error has occurred. The VI returns FALSE if the data operation has not completed.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code**

## niHSDIO Commit

Programs the hardware for the data operation using the properties you select. Before entering the committed state, most property values are stored in software only; these values have not yet been programmed to the hardware. Once the session is committed, the hardware is configured. <u>Details</u>

Use the pull-down menu to select an instance of this VI.

Select an instance 🔽

#### Dynamic

This instance of niHSDIO Commit configures the hardware for a dynamic operation. Start the operation with the <u>niHSDIO Initiate</u> VI. If you commit while the dynamic operation is in progress, you receive an error.

For many operations it is not necessary to explicitly use the Dynamic instance of the niHSDIO Commit VI because the following VIs implicitly commit: <u>niHSDIO Initiate</u>, <u>niHSDIO Read Waveform</u>, <u>niHSDIO Write Named Waveform</u>, and <u>niHSDIO Write Script</u>.

instrument handle ****	MINSON INStrument handle out
error in (no error)	error out

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code**

#### Static

This instance of niHSDIO Commit configures the hardware for a static operation. For most static operations, it is not necessary to explicitly use the Static instance of the niHSDIO Commit VI because the following VIs implicitly commit: <u>niHSDIO Read Static (U32)</u> and <u>niHSDIO Write Static (U32)</u>.

instrument handle	e out
error in (no error)	

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The

#### niHSDIO Commit Details

Committing only programs the hardware properties that have changed since previous commits.



**Note** Committing some properties (for example, voltage levels), may have immediate effects seen on the device connectors.



**Note** Before committing a session that requires an external clock, ensure the external clock is available. Otherwise you receive an error that the device could not find or lock to the external clock.

# niHSDIO Reset

Resets the session to its Initial state. All channels and front panel terminals are put into a high-impedance state. All software attributes are reset to their initial values. During a reset, routes of signal between this and other devices are released, regardless of which device created the route. For instance, a trigger signal being exported to a PXI trigger line and used by another device will no longer be exported. <u>Details</u>



- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

#### niHSDIO Reset Details

The niHSDIO Reset VI is applied to the *entire* device. If you have both a generation and an acquisition session active, the niHSDIO Reset VI resets the current session, including attributes, and invalidates the other session if it is committed or running. The other session must be closed.

## niHSDIO Reset Device

Resets the device to its Initial state and reloads the FPGA. All channels and front panel terminals are put into a high-impedance state. All software attributes are reset to their initial values. The entire contents of the FPGA and EEPROM files are reloaded. Use this function to reenable your device if it has disabled itself because the device temperature has risen above its optimal operating temperature.

During a device reset, routes of signals between this and other devices are released, regardless of which device created the route. For instance, a trigger signal being exported to a PXI trigger line and used by another device is no longer exported. <u>Details</u>



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

#### niHSDIO Reset Device Details

The niHSDIO Reset Device VI is applied to the *entire* device. If you have both a generation and an acquisition session active, the niHSDIO Reset Device VI resets the current session, including attributes, and invalidates the other session if it is committed or running. The other session must be closed.

Generally, using the <u>niHSDIO Reset</u> VI is acceptable in place of the niHSDIO Reset Device VI. The <u>niHSDIO Reset</u> VI executes more quickly.
## niHSDIO Send Software Edge Trigger

Forces a particular edge-based trigger to occur.

This VI only applies to the triggers listed below, and is valid if the particular trigger has been configured for edge, pattern match, or software triggering:

- Start Trigger
- Reference Trigger
- Advance Trigger
- Script Trigger

For edge or pattern match triggers, you can use this VI as a software override.

instrument handle	ent handle out
trigger (start trigger)	t

1/01	<b>instrument handle</b> identifies your instrument session. <b>instrument handle</b> was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
U16	trigger specifies the trigger to assert.
	You can select <b>Start Trigger</b> , <b>Reference Trigger</b> , <b>Advance</b> <b>Trigger</b> , or <b>Script Trigger</b> as the value for this control.
abc	trigger identifier specifies the trigger to assert.
	<b>trigger identifier</b> can be ScriptTrigger0, ScriptTrigger1, ScriptTrigger2, or ScriptTrigger3, or you could leave this parameter blank for the Start and Reference triggers.
	<b>error in</b> describes error conditions that occur before this VI or function runs.
	<b>status</b> is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
	<b>code</b> is the error or warning code. The default is 0. If <b>status</b> is TRUE, <b>code</b> is a negative error code. If <b>status</b> is FALSE, <b>code</b> is 0 or a warning code.
	<b>source</b> identifies where an error occurred. The source string includes the name of the VI that produced the error.

what inputs are in error, and how to eliminate the error.

### niHSDIO Error Message

Takes the error code returned by the NI-HSDIO VIs, interprets it, and returns it as a readable string.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error code is the error code returned by the device. The default value is 0, which means no errors occurred.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error message returns a user-readable message string that corresponds to the status code you specify.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a

## niHSDIO Self Test

Performs a self-test on the instrument and returns the test results. The niHSDIO Self Test VI performs a simple series of tests that ensure the instrument is powered up and responding. Complete functional testing and calibration are not performed by this function.

This function is internal and does not affect external I/O connections or connections between devices.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- **self test result** contains the value returned from the device self-test. A 0 means the self-test passed; anything else means the test failed.
- **self test message** returns the self-test response string from the device.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

# **Calibration Subpalette**

Use the VIs located on the NI-HSDIO»Dynamic and Static Acquisition»Utility»Calibration palette to access calibration operations.

Click the icons for VI and function descriptions.



## niHSDIO Self Cal

Self-calibrates the device. During self-calibration, the VCXO oscillator phase D/A converters are recalibrated.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where and why an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the orror

## niHSDIO Change Ext Cal Password

Changes the password that is required to initialize an external calibration session. **password** may be up to four characters long.

You can call this VI from an acquisition, generation, or calibration session.



- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **old password** specifies the old (current) external calibration password.
- **new password** specifies the new (desired) external calibration password.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

## niHSDIO Init Ext Cal

Creates and initializes a special NI-HSDIO external calibration session. **instrument handle out** is an NI-HSDIO session that can be used during the calibration session.

Multiple calls to this function return the same session ID. Calibration sessions are mutually exclusive with acquisition and generation sessions.

resource name	manager NH HSDID	
password ("") error in (no error)	error out	

- resource name specifies the device name, for example "PXI1Slot3," where "PXI1Slot3" is a device name assigned by Measurement & Automation Explorer.
- **password** is the current calibration password for the device. This password is case sensitive. The default password for all NI products is NI.
- error in (no error) can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

**status** is either TRUE (X) for an error, or FALSE (checkmark) for no error or a warning.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

- **code** identifies the error or warning. The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.
- **source** describes the origin of the error or warning.

The pop-up option **Explain Error** (or Explain Warning) gives more information about the error displayed.

- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an

## niHSDIO Cal Adjust Channel Voltage

Adjusts the voltage of the selected channel(s). The only errors that can be returned are actual calibration process errors.



**Notes** This function is not supported for the NI 654x/656x devices.

This function runs a static loopback test before doing adjusting the voltage. You must disconnect the cable from your device to run this function.

channel ("")

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel** specifies channels on which voltage will be adjusted.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred

## niHSDIO Close Ext Cal

Closes an NI-HSDIO external calibration session and, if specified, stores the new calibration constants and calibration data in the onboard EEPROM.



**Note** Whether you commit or cancel, the device is reset and the FPGA is reloaded afterwards.



instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.

**calibration action** specifies the action to perform upon closing.

Cancel	No changes are made to the calibration constants and data in the EEPROM.
Commit	The new calibration constants and data determined
	during the external calibration session are stored in the
	onboard EEPROM, given that the calibration was
	complete and passed successfully.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

### niHSDIO Convert Binary to WDT

Converts unsigned binary data into the digital waveform data type (WDT). Use this VI to easily convert extended state (0, 1, H, L, X, Z) digital waveforms from binary data. <u>Details.</u>

Use the pull-down menu to select an instance of this VI.

•

Select an instance

### Stimulus Data

Converts binary data to drive low (0) or drive high (1) digital waveform data types. You can specify a bitmask to create tristate (Z) data.



- **dt** specifies the time between values in **digital waveform**.
- **signal list** specifies which bits from **data** to include in the **digital waveform**.
- **data** specifies the high and low values (assuming **drive enabled** is enabled) in the **digital waveform**.
- **drive enable** specifies the bitmask that selects between tristating and driving the value selected in the **data**. Bits set to 0 translate to Z (tristate), and bits set to 1 translate to the value of the corresponding bit in **data**. If **drive enable** is empty, all the values in the array translate to 0 and 1.
- **compress data** specifies whether the **digital waveform** data is compressed.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- **digital waveform** contains the converted data for the waveform.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

#### **Response Data**

Converts binary data to expect low (L) or expect high (H) digital waveform data types. A bitmask may be specified to create don't care (X) data.



- **dt** specifies the time between values in **digital waveform**.
- signal list specifies which bits from data to include in the digital waveform.
- **data** specifies the high and low values (assuming **drive enabled** is enabled) in the **digital waveform**.
- **compare enable** specifies the bitmask that selects between X (ignore) and a compare value determined by the data array. Bits set to 0 translate to X, and bits set to 1 translate to L (compare low) or H (compare high) depending on the value of the corresponding bit in the data . If this array is empty, all the values in the data array translate to L and H.
- **compress data** specifies whether the **digital waveform** data is compressed.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- **digital waveform** contains the converted data for the waveform.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.

#### **Stimulus and Response Data**

Compiles binary data to bidirectional drive (0, 1, Z) and compare (H, L, X) data.

**Note:** A bit cannot be set to 1 in both the **drive enable** and **compare enable** arrays. Compare values (H, L, 0, and X) always cause the channel to be set to tristate. <u>Details.</u>



- **dt** specifies the time between values in **digital waveform**.
- signal list specifies which bits from data to include in the digital waveform.
- **data** specifies the high and low values (assuming **drive enabled** is enabled) in the **digital waveform**.
- **drive enable** specifies the bitmask that selects between tristating and driving the value selected in the **data**. Bits set to 0 translate to Z (tristate), and bits set to 1 translate to the value of the corresponding bit in **data**. If **drive enable** is empty, this VI translates all bits to 0 or 1 (all **drive enable** bits are assumed to be set to 1). Refer to <u>Details</u> for more information about how **drive enable** and **compare enable** interact.
- compare enable specifies the bitmask that selects between X (ignore) and a compare value determined by the data array. Bits set to 0 translate to X, and bits set to 1 translate to L (compare low) or H (compare high) depending on the value of the corresponding bit in data. If compare enable is empty, this VI translates all bits to 0, L, or Z based on data and drive enable. Refer to Details for more information about how drive enable and compare enable interact.
- **compress data** specifies whether the **digital waveform** data is compressed.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

#### niHSDIO Convert Binary To WDT Stimulus And Response Data (U32) Details

The following table shows how the Stimulus and Response instance of this VI translates bits, depending on the values set for **drive enable** and **compare enable**.

		drive	enable
		0	1
compare enable	0	Z	0/1*
compare enable	1	L/H*	Error
*This value is determined by th	e co	orrespondir	ng bit in <b>data</b> .

### niHSDIO Get Session Reference

Returns a session reference you can pass to other VIs. Session references are of generic type, which means that the corresponding wires are blue-green, unlike the wires for regular instrument driver sessions.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- session reference references the device session. session reference is a generic ViSession reference that can be passed to other VIs.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is

# **Dynamic and Static Generation Subpalette**

Use the VIs located on the **NI-HSDIO**»**Dynamic and Static Generation** palette to program generation operations.

Click the icons for VI and function descriptions.



## niHSDIO Init Generation Session

Creates a new generation session. You can perform static and dynamic generation operations with this session.

Creating a new session does not automatically tristate your front panel terminals or channels that might have been left driving voltages from previous sessions. Refer to the <u>niHSDIO Close</u> VI for more information on leaving lines driving after closing a session.

Set **reset instrument** to TRUE to place your device in a known start-up state when creating a new session. This action is equivalent to using the <u>niHSDIO Reset</u> VI, and it tristates the front panel terminals and channels.



- **resource name** specifies the device name, for example "PXI1Slot3," where "PXI1Slot3" is a device name assigned by Measurement & Automation Explorer.
- id query specifies whether the driver performs an ID query upon the instrument.

When **id query** is set to TRUE, the driver ensures compatibility between the instrument and the driver. When **id query** is set to FALSE, the driver skips the ID query.

**reset instrument** specifies whether the driver resets the device during initialization of the session. TRUE means that the device is reset; FALSE means that the device is not reset.

Refer to <u>niHSDIO Reset</u> for more information on what happens during an instrument reset.



**Note** Resetting your device resets the *entire* device. Acquisition or generation operations in progress will be aborted and cleared.

- option string is currently unused. Leave this parameter unwired.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The

### niHSDIO Write Named Waveform

Transfers waveform data from PC memory to onboard memory. If you specify a **waveform name** not already being used on the instrument, the appropriate amount of onboard memory is allocated (if available) and the data is stored in that new location. Supported devices for the binary instances of this VI depend on the <u>data width</u> for your device, not on the number of assigned dynamic channels. The WDT and HWS instances do not have this restriction. <u>Details</u>.

Use the pull-down menu to select an instance of this VI.

Select an instance 🔹

#### 1D U32

This instance of niHSDIO Write Named Waveform writes the waveform to onboard memory from a one-dimensional array of unsigned 32-bit data. This instance is supported for devices with a data width of 4.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.

- You are using scripts
- You want to download multiple waveforms into the hardware
- **data** is the waveform where acquired samples are written.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces

### 1D U16

This instance of niHSDIO Write Named Waveform writes the waveform to onboard memory from a one-dimensional array of unsigned 16-bit data. This instance is supported for devices with a data width of 2.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.

- You are using scripts
- You want to download multiple waveforms into the hardware
- **data** contains the data to be written.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces

#### 1D U8

This instance of niHSDIO Write Named Waveform writes the waveform to onboard memory from a one-dimensional array of unsigned 8-bit data. This instance is supported for devices with a data width of 1.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.

- You are using scripts
- You want to download multiple waveforms into the hardware
- **data** contains the data for the waveform to be written.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If
     status is TRUE, code is a negative error code. If status is
     FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces

#### WDT

This instance of niHSDIO Write Named Waveform writes the waveform to onboard memory as waveform data type data.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.

You should name waveforms using this VI under either of the following conditions:

- You are using scripts
- You want to download multiple waveforms into the hardware
- data is the waveform where acquired samples are written.
- **Use rate from waveform** controls how the sample rate is computed.

Setting this value to TRUE computes the sample rate from the WDT value. If the sample rate has been configured using the niHSDIO Configure Sample Clock VI, Use rate from waveform overrides the sample rate.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- **instrument handle out** passes a reference to your instrument

#### HWS

This instance of niHSDIO Write Named Waveform writes the waveform to onboard memory from a .hws (Hierarchical Waveform Storage) file.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.

You should name waveforms using this VI under either of the following conditions:

- You are using scripts
- You want to download multiple waveforms into the hardware
- **file path** specifies the path and file name of the HWS file to open. The .hws extension is typically used for HWS files, although using this extension is optional.
- **Use rate from waveform** controls how the sample rate is computed.

Setting this value to TRUE computes the sample rate from the WDT value. If the sample rate has been configured using the <u>niHSDIO Configure Sample Clock</u> VI, **Use rate from waveform** overrides the sample rate.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

### **Direct DMA**

Writes a specified amount of data from a Direct DMA-compatible device to the waveform in onboard memory. The sample size is 4 bytes for NI 654x/655x devices, and 2 bytes for the NI 656x in SDR mode or 1 byte in DDR mode.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- window address specifies the window address from the Direct DMA-compatible data source.
- **samples to write** specifies the number of samples to write from the Direct DMA-compatible data source.
- waveform name represents the name to associate with the allocated waveform memory.

- You are using scripts
- You want to download multiple waveforms into the hardware
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u>

#### niHSDIO Write Named Waveform Details

Data is always written to memory starting at the waveform write position. The initial write position for a new waveform is the start of the allocated memory. This VI moves the next write position to the end of the data just written, so subsequent uses of this VI appends data to the end of previously written data. You may also manually change the write position with the <u>niHSDIO Set Named Waveform Next Write Position</u> VI. If you try to write past the end of the allocated space, the VI returns an error.

Waveforms are stored contiguously in onboard memory. You cannot resize an existing named waveform. Instead, delete the existing waveform using the <u>niHSDIO Delete Named Waveform</u> VI and then recreate it with the new size using the same name.

This VI also calls the Dynamic instance of the <u>niHSDIO Commit</u> VI.

When you explicitly call the <u>niHSDIO Allocate Named Waveform</u> VI and write waveforms using multiple niHSDIO Write Named Waveform VIs, each waveform block written must be a multiple of 32 samples for the NI 654*x*/655*x* devices or a multiple of 64 samples for the NI 656*x* devices (128 samples in DDR mode).



**Note** If the waveform was allocated 10 KB, but this function has only written a 5 KB waveform, the remaining 5 KB contain invalid data.

# **Generation Configuration Subpalette**

Use the VIs located on the **NI-HSDIO**»Dynamic and Static Generation»Generation Configuration palette to configure generation operations.

Click the icons for VI and function descriptions.



## niHSDIO Configure Generation Repeat

Specifies how many times to generate a waveform or whether it should be continuously generated.

This VI is only valid when the **generation mode** parameter of the <u>Configure Generation Mode</u> VI is set to **Waveform**. This VI does not apply in scripted mode.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- repeat mode specifies the repeat mode to configure. You can select either **Finite** or **Continuous** as the value of this parameter.

If you choose **Finite**, specify the number of times to repeat in the **repeat count** parameter. If you choose **Continuous**, calling the <u>niHSDIO Initiate</u> VI generates the named waveform continuously until the <u>niHSDIO Abort</u> VI is called. **repeatCount** is ignored.

repeat count specifies the number of times to generate the waveform.

repeat count is ignored if repeat mode is set to Continuous.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> Generation Session VI

## niHSDIO Configure Initial State

Sets the <u>Initial state</u> of the channels for a dynamic generation operation.

The Initial state of each channel is driven once the operation is initiated using the <u>niHSDIO Initiate</u> VI. Channels remain unchanged until the operation is initiated. The Initial state is active once the session is initiated until the first waveform sample is generated.

Use the pull-down menu to select an instance of this VI.

Select an instance 🔽

### Initial state (String)

You can specify **initial state** in either a binary format or a string. This instance of niHSDIO Configure Initial State uses a string format to represent the Initial state of a dynamic pattern generation session.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies the channels being sampled.
- **initial state** describes the Initial state of a dynamic generation operation.

This expression is composed of characters:

- X or x: keeps the previous value
- 1: sets the channel to logic high
- 0: sets the channel to logic low
- Z or z: disables the channel or sets it to a high-impedance state



**Note** NI 656*x* devices do not support the high-impedance (Z) Initial state.

The first character in **initial state** corresponds to the first channel in **channel list**. The number of characters in the pattern must equal the number of channels specified in **channel list** or an error is returned.

**error in** describes error conditions that occur before this VI or function runs.

- **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
- **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
- **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

### Initial state (U32)

This instance of niHSDIO Configure Initial State uses a binary format (1s and 0s) to represent the Initial state of a dynamic generation session. If you require more choices for your Initial state, use the niHSDIO Configure Initial State (String) instance of the niHSDIO Configure Initial State VI.

instrument handle	instrument handle out
error in (no error)	error out

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- initial state describes the Initial state of a dynamic generation operation. initial state defines the bit mask representing the Initial state. High is specified with a 1, and low is specified with a 0. If you need to specify values other than high and low, use the niHSDIO Configure Initial State (String) instance of this VI.

The first character in **initial state** corresponds to the first channel in **channelList**. The number of characters in the pattern must equal the number of channels specified in **channelList** or an error is returned.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains

### niHSDIO Configure Idle State

Sets the <u>Idle state</u> of the channels for a dynamic generation operation.

The operation may be idle when the generation operation completes normally, when the generation operation pauses from an active Pause trigger, or when the generation operation terminates because of an underflow error.

Use the pull-down menu to select an instance of this VI.

Select an instance 🔽

#### Idle state (String)

You can specify **idle state** in either a binary or a string format. This instance of niHSDIO Configure Idle State uses a string format to represent the Initial state of a dynamic pattern generation session.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **channel list** specifies the channels being sampled.
- **idle state** describes the Idle state of a dynamic generation operation.

This expression is composed of characters:

- X or x: Keeps the previous value
- 1: Sets the channel to logic high
- 0: Sets the channel to logic low
- Z or z: Disables the channel (sets it to a high-impedance state).



**Note** NI 656*x* devices do not support the high-impedance (Z) Idle state.

The first character in **idle state** corresponds to the first channel in **channel list**. The number of characters in the pattern must equal the number of channels specified in **channel list** or an error is returned.

**error in** describes error conditions that occur before this VI or function runs.

- **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
- **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
- **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.

#### Idle state (U32)

This instance of niHSDIO Configure Idle State uses a binary format to only represent logic high and logic low. If you require more choices for your Idle state, use the niHSDIO Configure Idle State (String) VI.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- **idle state** describes the Idle state across all channels configured for dynamic generation.

**idle state** defines the bit mask representing the Idle state. High is specified with a 1, and low is specified with a 0. If you need to specify values other than high or low, use the niHSDIO Configure Idle State (String) instance of this VI.

The first character in **idle state** corresponds to the first channel in **channel list**. The number of characters in the pattern must equal the number of channels specified in **channel list** or an error is returned.

Each binary digit of this value is applied to the corresponding channel if it is configured for dynamic generation.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.

## **Waveform Control Subpalette**

Use the VIs located on the **NI-HSDIO**»Dynamic and Static Generation»Generation Configuration»Waveform Control palette to configure generation operations.

Click the icons for VI and function descriptions.



### niHSDIO Configure Waveform To Generate

Sets the waveform to be generated upon a call to the <u>niHSDIO</u> <u>Initiate</u> VI when the **generation mode** parameter of the <u>niHSDIO</u> <u>Configure Generation Mode</u> VI is set to **Waveform** 

If this function is not called and you have multiple waveforms in onboard memory, NI-HSDIO generates an error at initiate.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name of the waveform to be generated at initiate.
- **error in** describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code**

## niHSDIO Allocate Named Waveform

Reserves waveform space in onboard memory and associates a waveform name with it. Individual waveforms are stored contiguously in <u>onboard memory</u>.

The name given to the waveform is the same name used in the <u>niHSDIO Write Named Waveform</u> VI to populate the waveform with data, as well as the waveform name referenced in <u>scripts</u>.

If space is unavailable to accommodate a waveform of size **size in samples**, an error is returned and no memory space is allocated.

This VI does not change any data on the device itself, but rather adds the named reference in software. Use the <u>niHSDIO Write Named</u> <u>Waveform</u> VI to fill the onboard memory with waveform data to be generated.

waveform name (default:"")	
<b>instrument handle</b> size in samples (10000) error in (no error)	error out

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.

You should name waveforms using this VI under either of the following conditions:

- You are using scripts
- You want to download multiple waveforms into the hardware
- **size in samples** specifies the number of samples to allocate for the named waveform.

The number of bits in the allocated samples differs depending on the device you are using.

Device	Bits Per Sample
NI 654 <i>x</i>	32
NI 655 <i>x</i>	32
NI 656 <i>x</i>	16 in SDR mode 8 in DDR mode

Г	-		T	1
	5	۰.		

error in describes error conditions that occur before this VI or
### niHSDIO Delete Named Waveform

Frees waveform space in onboard memory.

The niHSDIO Delete Named Waveform VI releases <u>onboard memory</u> space previously allocated by either the <u>niHSDIO Allocate Named</u> <u>Waveform</u> VI or the <u>niHSDIO Write Named Waveform</u> VI. Any future references to the deleted waveform result in an error. However, previously written scripts that still reference the deleted waveform will not generate an error at initiation.

An error is generated if the waveform name is not allocated in onboard memory.



- waveform name is the name of the waveform to delete.
- instrument handle identifies your instrument session.
  instrument handle was obtained from the <u>niHSDIO Init</u>
  <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **Status** is TRUE (X) if an error occurred or FAUSE

# niHSDIO Set Named Waveform Next Write Position

Modifies where within a named waveform to next write data.

The <u>niHSDIO Write Named Waveform</u> VI always begins writing at the current write position. Existing data in the waveform is overwritten. <u>Details</u>



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- waveform name represents the name to associate with the allocated waveform memory.
- position Specifies where to place the write position, in conjunction with offset. You can choose either Current Write Position or Start of Waveform as the values for this control.

If you choose **Start of Waveform**, the offset is relative to the beginning of the waveform. If you choose **Current Write Position**, the offset is relative to the current write position in the waveform.

**offset** allows you to set the write position of the named waveform, in conjunction with **position**. **offset** is in samples.

Before issuing a write waveform command, **offset** relative to the start of the waveform must be a multiple of 32 samples for the NI 654x/655x or a multiple of 64 samples for the NI 656x (128 samples for the NI 656x in DDR mode).

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a negative error code. If **status** is FALSE, **code** is 0 or a warning code.

### niHSDIO Set Named Waveform Next Write Position Details

**position** and **offset** are used together to determine where the next write position will be. **position** describes an absolute or relative move. **offset** is the number of samples to shift the next write position. You must always set the write position at a position that is a multiple of 32 samples for the NI 654x/655x or a multiple of 64 samples for the NI 656x (128 samples for the NI 656x in DDR mode).

The write position is moved to the end of the most recently written data after each use of the <u>niHSDIO Write Named Waveform</u> VI. Thus you do not need to explicitly use this VI unless you want to. Attempting to set the write position beyond the bounds of the allocated space results in an error.

Position	Offset	Next Write Position
Start of Waveform	0	Start of waveform.
Start of Waveform	5	Sixth sample of waveform.
Start of Waveform	-1	ERROR. These settings would try to place write position before start of waveform.
Current Write Position	0	No effect. These settings leave the next write position unchanged.
Current Write Position	10	Shift write position 10 samples ahead from current location. This position setting is only valid if the current write position plus this offset is in the waveform.
Current Write Position	-10	Shift write position 10 samples back from current location. This position setting is only valid if the current write position is greater than 10.

Examples of combinations of **position** and **offset**:

# **Scripting Subpalette**

Use the VIs located on the **NI-HSDIO**»Dynamic and Static Generation»Generation Configuration»Scripting palette to configure generation operations.

Click the icons for VI and function descriptions.



### niHSDIO Configure Generation Mode

Specifies whether to initiate the waveform generation based on a specified script or based on a waveform.

Initiation occurs upon calling the <u>niHSDIO Initiate</u> VI.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **generation mode** specifies the generation mode to configure. You can choose **Waveform** or **Scripted** as the values for this parameter.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the niHSDIO Init Acquisition Session VI or the niHSDIO Init Generation Session VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a

### niHSDIO Write Script

Writes a string containing scripts that govern the generation of waveforms.

If this function is called repeatedly, previously written <u>scripts</u> with unique names remain loaded. Previously written scripts with identical names to those being written are replaced. If there are multiple scripts loaded when the <u>niHSDIO Initiate</u> VI is called, then one of the scripts must be designated as the script to generate. If there is only one script in memory, you do not need to designate the script to generate.

An error is returned if the script uses incorrect syntax. This VI calls the Dynamic instance of the <u>niHSDIO Commit</u> VI. All pending attributes are committed to hardware.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> Acquisition Session VI or the <u>niHSDIO Init Generation Session</u> VI.
- **script** contains the text of the script you want to use for your generation operation.

For more information on scripting syntax, select **Programming»Reference»Scripting Instructions** from the table of contents in this help file.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u>

### niHSDIO Configure Script To Generate

Sets the script to be generated upon a call to the <u>niHSDIO Initiate</u> VI when the **generation mode** parameter of the <u>niHSDIO Configure</u> <u>Generation Mode</u> VI is set to **Scripted**. If there are multiple scripts loaded when <u>niHSDIO Initiate</u> is called, then one of the scripts must be designated the script to generate or you receive an error.

This function need only be called if multiple <u>scripts</u> are present in <u>onboard memory</u>.

instrument handle	NIHSDIO
script name	
error in (no error)	

- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- script name specifies a string containing a syntactically correct script.
- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
  - code is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** identifies where an error occurred. The source string includes the name of the VI that produced the error, what inputs are in error, and how to eliminate the error.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle was obtained from the <u>niHSDIO Init Acquisition Session</u> VI or the <u>niHSDIO Init</u> <u>Generation Session</u> VI.
- error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error

# **Static Generation Subpalette**

Use the VIs located on the **NI-HSDIO»Dynamic and Static Generation»Static Generation** palette to program generation operations.

Click the icons for VI and function descriptions.



### niHSDIO Write Static (U32)

Writes to channels configured for static generation. You can configure a channel for static generation using the <u>niHSDIO Assign Static</u> <u>Channels</u> VI.



- instrument handle identifies your instrument session. instrument handle was obtained from the <u>niHSDIO Init</u> <u>Acquisition Session</u> VI or the <u>niHSDIO Init Generation Session</u> VI.
- write data is the bit-value of data to drive on channels configured for static generation. 1 corresponds to logic high level, 0 corresponds to logic low level.

The least significant bit of **write data** corresponds to the lowest physical channel number. For example, a **write data** value of 0xFF00 sets the lower 8 channels to 0, while setting the upper 8 channels to logic high level.

Data values in **write data** corresponding to channels not configured for static generation are ignored.

Static channels explicitly disabled with the <u>niHSDIO Tristate</u> <u>Channels</u> VI remain disabled, but the channel data value changes internally. Re-enabling a channel with <u>niHSDIO Tristate</u> <u>Channels</u> VI causes the channel to drive any value that you have written to it, even while the channel was disabled.

**channel mask** specifies the bit-value of channels to leave unchanged. 1 means to change the channel to whatever is reflected by **write data**. 0 means do not alter the channel, regardless of **write data**.

The least significant bit of **channel mask** corresponds to the lowest physical channel number. For example, a **write data** value of 0xFFFF and **channel mask** of 0x0080 means set only channel 7 to 1; all other channels remain unchanged.

- error in describes error conditions that occur before this VI or function runs.
  - **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE

# **Utility Subpalette**

Use the VIs located on the NI-HSDIO»Dynamic and Static Generation»Utility palette to access utility features of NI-HSDIO.

Click the icons for VI and function descriptions.



### **NI-HSDIO Express (Acquisition) VI**

Acquires a digital waveform from a National Instruments digital waveform generators/analyzer.

**Dialog Box Options** 

**Block Diagram Inputs** 

Block Diagram Outputs

### **Dialog Box Options**

Parameter	Description		
Configuration	Contains the following options:		
	<ul> <li>Device—Specifies the device used. Unavailable devices are disabled.</li> </ul>		
	• Enable HW Compare—Specifies whether this Express VI compares the acquired data to an expected response. Configure the expected response data source using the HW Compare tab.		
	• Tristate channels before acquisition— Specifies whether configured channels that were previously left generating data are tristated.		
	Checked: Configured channels previously left generating data are tristated.		
	Unchecked: Configured channels previously left generating data are not tristated.		
	<ul> <li>Channels—Displays the channels currently configured to acquire data.</li> </ul>		
	<ul> <li>Modify—Click Modify to launch the Selection Channels dialog box.</li> </ul>		
	<ul> <li>Timing—Contains the following options:</li> <li>Rate (Hz)—Specifies the Sample clock frequency for the acquisition.</li> </ul>		
	Units: Samples per second (S/s)		
	<ul> <li>Samples to Read—Specifies the number of samples to acquire.</li> </ul>		
Voltage	<ul> <li>Contains the following options:</li> <li>Logic family—Specifies whether you configure your voltage levels using one of the predefined voltage families for this device.</li> <li>Custom levels (V)—Specifies whether you configure custom high and low voltage thresholds for the acquisition. This feature is not supported on all devices.</li> <li>High—Specifies the high voltage threshold for the acquisition. This feature is not</li> </ul>		

### Block Diagram Inputs

Parameter	Description
data	Contains the expected response data for the comparison operation. <b>data</b> does not exist when you select the Read from File option in the NI-HSDIO Express (Acquisition) Results page.
close	<ul> <li>Determines whether the instrument session remains open when the VI finishes execution. Use this parameter for loop optimization by setting it to FALSE on all iterations other than the last iteration. close is TRUE by default.</li> <li>Note This input is not intended to be used to share the session between Express VIs. If you have a loop containing multiple Express VIs that use the same device, you must wire in TRUE for this input.</li> </ul>
max time	Specifies the timeout value for the Express VI.
error in	Describes error conditions that occur before this Express VI runs.

### **Block Diagram Outputs**

Parameter	Description
data	Contains the digital data acquired by the device.
passed	Returns the pass/fail result of the last hardware comparison operation.
number of sample errors	Returns the number of sample errors found in the last hardware comparison operation.
BER	Returns the bit error rate (BER). BER is calculated by taking the number of sample errors and dividing it by the total number of samples compared.
BER per channel	Returns the bit error rate (BER) for each channel. BER for an individual channel is calculated by taking the number of bit errors found for that channel and dividing it by the total number of samples compared.
error locations	Returns the bit numbers and the bit locations that were in error for each sample that has an error.
error out	Contains error information. If <b>error in</b> indicates that an error occurred before this Express VI ran, <b>error out</b> contains the same error information. Otherwise, it describes the error status that this Express VI produces.

This Express VI uses the functionality of the following VIs and functions:

niHSDIO Init Acquisition Session niHSDIO Tristate Channels niHSDIO Assign Dynamic Channels niHSDIO Configure Sample Clock niHSDIO Configure Voltage niHSDIO Configure Acquisition Size niHSDIO Configure Data Position niHSDIO Configure Data Position Delay niHSDIO Configure Ref Clock niHSDIO Configure Trigger niHSDIO Read Waveform niHSDIO Close

### **NI-HSDIO Express (Generation) VI**

Acquires a digital waveform from a National Instruments digital waveform generators/analyzer.

**Dialog Box Options** 

**Block Diagram Inputs** 

Block Diagram Outputs

### **Dialog Box Options**

Parameter	Description	
Configuration	Contains the following options:	
	• <b>Device</b> —Specifies the device to use for the generation. This ring control lists all devices installed on this computer which can be used by this Express VI. If you reopen the NI-HSDIO Express (Generation) configuration page and the current device is dimmed, you can no longer select it.	
	<ul> <li>Generation Mode—You can select one of the following modes:</li> </ul>	
	<ul> <li>Finite—Configures the device to generate a single waveform once. In this generation mode, the Express VI waits until the waveform generation is complete or Max time expires before exiting.</li> </ul>	
	<ul> <li>Start Continuous—Configures the device to generate the same waveform continuously until it is stopped or a new waveform is downloaded. In this generation mode, the Express VI returns without waiting for the waveform generation to complete.</li> <li>Stop Continuous—Configures the Express VI to stop a generation previously started by an instance of this Express VI configured in Start continuous generation mode. This VI stops the device and releases all device resources used by the generation.</li> </ul>	
	Channels—Displays the channels currently	
	<ul> <li>Modify—Click Modify to launch the Select Channels dialog box.</li> </ul>	
	• <b>Timing</b> —Contains the following options:	
	<ul> <li>Extract rate from waveform—Specifies whether this Express VI configures the device sample rate using the value in the waveform at the data input terminal or the</li> </ul>	

### Block Diagram Inputs

Parameter	Description
data	Contains the digital waveform to generate. <b>data</b> does not exist when you select the Read from File option in the NI- HSDIO Express (Generation) configuration page.
close	<ul> <li>Determines whether the instrument session remains open when the Express VI finishes execution. Use this parameter for loop optimization by setting it to FALSE on all iterations other than the last iteration. close is TRUE by default.</li> <li>Note This input is not intended to be used to share the session between Express VIs. If you have a loop containing multiple Express VIs that use the same device, you must wire in TRUE for this input.</li> </ul>
max time	Specifies the timeout value for the generation.
error in	Describes error conditions that occur before this Express VI runs.

### **Block Diagram Outputs**

Parameter	Description
error out	Contains error information. If <b>error in</b> indicates that an error occurred before this Express VI ran, <b>error out</b> contains the same error information. Otherwise, it describes the error status that this Express VI produces.

This Express VI uses the functionality of the following VIs and functions:

niHSDIO Init Generation Session niHSDIO Assign Dynamic Channels niHSDIO Configure Sample Clock niHSDIO Configure Voltage niHSDIO Configure Generation Repeat niHSDIO Export Signal niHSDIO Export Signal niHSDIO Property Node niHSDIO Configure Data Position niHSDIO Configure Data Position Delay niHSDIO Configure Trigger niHSDIO Configure Trigger niHSDIO Write Named Waveform niHSDIO Initiate niHSDIO Initiate niHSDIO Wait Until Done niHSDIO Abort niHSDIO Close

### niHSDIO Property Node

The niHSDIO Property Node is used to set, get, or check properties.

Some NI-HSDIO properties are channel based. When a property is channel based, you must specify an active channel before setting, getting, or checking properties.

# **Active Channels**

#### Short Name: ActiveChannels

Specifies part of the session to which subsequent properties apply. The Active Channels property is most often used to specify a channel or channels. You can also use the Active Channels property to specify a script trigger or marker.

You can set the Active Channels property more than once within a property node, if you want to independently set and/or get properties that apply to different parts of a session. The following example shows how to set Voltage Levels: Data High to 5 for channel 0 and to 3 for channel 1.



If you want to set and/or get properties that apply to a part of the session, and then get and/or set properties that apply to the session as a whole within the same property node, set Active Channels to an empty string before the properties that apply to session as a whole. The following example shows how to set Data High for channel 0 and a Samples per Record for the session.



Data Type	ViString
Permissions	R/W
Channel Based	No

# **Dynamic Channels**

#### Short Name: DynamicChannels

Configures channels for dynamic operation. The group of dynamic channels is reconfigured each time this property is set. For example, setting this property to 0-10 and then setting it to 5-8 results in only channels 5-8 being assigned to dynamic. Channels 0-4 and 9-10 are unconfigured by the second configuration.

Writing an empty string to this property configures *all* channels for dynamic operation. Writing the value None unconfigures all channels for dynamic operation. The session must be committed before this property takes effect (refer to the <u>niHSDIO Commit</u> VI for more information on committing a session).

You can configure a channel for more than one simultaneous operation. A channel can be simultaneously configured for the following operations:

- Dynamic generation and any (static and/or dynamic) acquisition
- Static generation and any (static and/or dynamic) acquisition
- Both static and dynamic acquisition
- Note You *cannot* configure a particular channel for simultaneous dynamic *and* static generation.

Unconfiguring a dynamic generation channel frees that channel to be reconfigured for static generation. Unconfiguring a dynamic generation channel does not stop the channel from driving its current valueany value already written to the channel continues to be driven.

Syntax examples:

- 2-15 or 15-2 both set channels 2 through 15 to dynamic
- 0-3, 5, 8-15
- 0, 3, 10 or 3, 10, 0
- " (empty string)configure all channels for dynamic
- Noneunconfigure all dynamic channels

Data Type	ViString
Permissions	R/W
Channel Based	No

# **Static Channels**

#### Short Name: StaticChannels

Configures channels for static operation. The group of static channels is reconfigured each time this property is set. For example, setting this property to 0-10 and then setting it to 5-8 results in only channels 5-8 being assigned to static. Channels 0-4 and 9-10 are unconfigured by the second configuration. Writing an empty string to this property configures *all* channels for static. Writing the value *None* unconfigures all channels for static. The channel is not configured or unconfigured until a call to the niHSDIO Read Static (U32) VI (for acquisition sessions) or the niHSDIO Write Static (U32) VI (for generation sessions).

Writing an empty string to this property configures *all* channels for dynamic operation. Writing the value None unconfigures all channels for dynamic operation. The session must be committed before this property takes effect (refer to the <u>niHSDIO Commit</u> VI for more information on committing a session).

You can configure a channel for more than one simultaneous operation. A channel can be simultaneously configured for the following operations:

- Dynamic generation and any (static and/or dynamic) acquisition
- Static generation and any (static and/or dynamic) acquisition
- Both static and dynamic acquisition



**Note** You *cannot* configure a particular channel for simultaneous dynamic *and* static generation.

Unconfiguring a static generation channel frees that channel to be reconfigured for dynamic generation. Unconfiguring a static generation channel does not stop the channel from driving its current value any static value already written to the channel continues to be driven.

Syntax examples:

- 2-15 or 15-2 both set channels 2 through 15 to dynamic
- 0-3, 5, 8-15
- 0, 3, 10 or 3, 10, 0
- " (empty string)configure ALL channels for dynamic
- Noneunconfigure all dynamic channels

Data Type	ViString
Permissions	R/W
Channel Based	No

# Voltage Levels:Data High

#### Short Name: DataVolt.High

This attribute sets the high data voltage level for the session. For an acquisition session, this sets the Acquisition Voltage High Level. For a generation session, this sets the Generation Voltage High Level.

This property applies to static and dynamic data operations.



Note NI 656x devices do not support configuring voltage levels. NI-HSDIO returns an error if you select this property when programming those devices.

Units: volts

Data Type	ViReal64
Permissions	R/W
Channel Based	Yes

### **Voltage Levels:Data Low**

#### Short Name: DataVolt.Low

Specifies the data voltage low level for the session. For an acquisition session, this sets the Acquisition Voltage Low Level. For a generation session, this sets the Generation Voltage Low Level.

This property applies to static and dynamic data operations.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this property when programming those devices.

Units: volts

Data Type	ViReal64
Permissions	R/W
Channel Based	Yes

# Voltage Levels:Trigger High

#### Short Name: TrigVolt.High

Specifies the trigger voltage high level for the session.

If you do not explicitly set this property, NI-HSDIO assumes the same value as the <u>Data Voltage High Level</u> property for the acquisition session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this property when programming those devices.

Units: volts

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Voltage Levels:Trigger Low

#### Short Name: TrigVolt.Low

Specifies the trigger voltage low level for the session.

If you do not explicitly set this property, NI-HSDIO assumes the same value as the <u>Data Voltage Low Level</u> property for the acquisition session.



Note NI 656x devices do not support configuring voltage levels. NI-HSDIO returns an error if you use this property when programming those devices.

Units: volts

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# **Voltage Levels:Event High**

#### Short Name: EventVolt.High

Specifies the high event voltage level for the session.

If you do not explicitly set this property, NI-HSDIO assumes the same value as the <u>Data Voltage High Level</u> property for the generation session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this property when programming those devices.

Units: volts
Data Type	ViReal64
Permissions	R/W
Channel Based	No

# **Voltage Levels: Event Low**

#### Short Name: EventVolt.Low

Specifies the low event voltage level for the session.

If you do not explicitly set this property, NI-HSDIO assumes the same value as the **Data Voltage Low Level** property for the generation session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this property when programming those devices.

Units: volts

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# **Dynamic Acquisition:Samples Per Record**

#### Short Name: SampsPerRecord

Specifies the number of samples to be acquired per record. If you are using a <u>Reference trigger</u>, this includes both pretrigger and posttrigger samples.

This property is valid only for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Dynamic Acquisition:Number Of Records To Acquire

Short Name: NumRecords

Specifies the total number of <u>records</u> you want to acquire.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	niHSDIO Configure Acquisition Size

# **Dynamic Acquisition:Fetch:Fetch Relative To**

#### Short Name: FetchRelativeTo

Specifies the absolute location within the <u>acquired record</u> from which to begin fetching. The default value is **Current read position**. However, NI-HSDIO changes the default value internally as follows. If the <u>Reference</u> <u>trigger</u> is enabled (not disabled), then the fetch occurs from the first pretrigger sample. If the Reference trigger is disabled, then the fetch occurs from the first sample.

Most recent sample (46)	Specifies that fetching occur relative to the most recently acquired data. The <u>Fetch Offset</u> property must be negative.
First sample (47)	Specifies that fetching occurs at the first sample acquired by the device. If the device wraps its buffer, then the first sample is no longer available. In this case, NI-HSDIO returns an error if the fetch offset is in the overwritten data.
Reference trigger (48)	Specifies that fetching occur relative to the Reference trigger. This value behaves like <b>First Sample</b> if no Reference trigger is configured.
First pretrigger sample (49)	Specifies that fetching occur relative to the first pretrigger sample acquired. This value behaves like <b>First Sample</b> if no Reference trigger is configured.
Current read position (50)	Specifies that fetching occur after the last fetched sample.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Dynamic Acquisition:Fetch:Fetch Offset**

#### Short Name: FetchOffset

Specifies the offset in samples to start fetching acquired waveform data. The offset is applied relative to the <u>Fetch Relative To</u> position. Offset can be a positive or negative value.

If the specified offset would cause the fetch to exceed the end of the waveform, NI-HSDIO returns a data overwrite error. If the selected offset would cause the fetch location to occur before the start of the waveform, the fetch location is coerced to the beginning of the waveform.

This property is only valid for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Dynamic Acquisition:Fetch:Fetch Backlog**

#### Short Name: FetchBacklog

Queries how many acquired data points remain in onboard memory.

This property is only valid for acquisition sessions.

This property is used with the <u>Fetch Offset</u> and <u>Fetch Relative To</u> properties. This property returns the number of samples available from the given Fetch Relative To and Fetch Offset values.

Data Type	Vilnt32
Permissions	RO
Channel Based	No

# **Dynamic Acquisition:Fetch:Records Done**

Short Name: RecordsDone

Returns the number of <u>records</u> that have been acquired.

Data Type	Vilnt32
Permissions	RO
Channel Based	No
High-Level VI	None

# **Dynamic Acquisition:Input Impedance**

#### Short Name: InputImpedance

Use this property to change input impedance on the front panel connector.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI	High impedance
654 <i>x</i>	
	Refer to the <u>specifications</u> document for your device for more information on the supported high-impedance values.
NI 655 <i>x</i>	50 or high impedance
	Refer to the <u>specifications</u> document for your device for more information on the supported high-impedance values.
NI 656 <i>x</i>	100 in <u>LVDS</u> terminal configuration 10,000 in single-ended terminal configuration

Refer to the <u>Termination</u> section for acquisition with your device for more information about choosing the input impedance.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	Yes

# **Dynamic Acquisition:Data Interpretation**

#### Short Name: DataInterpretation

Specifies whether you acquire high/low data or valid/invalid data during a static or dynamic acquisition session. Refer to your specific hardware documentation to understand how data is returned to you when the voltage level is above voltage level low but below voltage level high and you select **High or low**.



**Note** NI 654*x*/656*x* devices only support the high/low mode of data interpretation. NI-HSDIO returns an error if you select valid/invalid mode for an acquisition with these devices.

This property is only valid for acquisition sessions.

High or low (3)	Select <b>High or low</b> to get logic high or logic low values.
Valid or invalid (4)	Select <b>Valid or invalid</b> to tell if the signal was in the undefined voltage level (above voltage level high but below voltage level low).

Data Type	Vilnt32
Permissions	R/W
Channel Based	Yes

# **Dynamic Generation:Initial State**

#### Short Name: InitialState

Specifies a dynamic generation channel state after the session is initiated and before the first waveform sample is generated. The channel changes to the Initial state once the data operation has been initiated. When the start trigger occurs, the Initial state is replaced by the first sample in the waveform.

Channels explicitly disabled with the <u>niHSDIO Tristate Channels</u> VI remain disabled, but the channel data value changes internally. Reenabling a channel with the <u>niHSDIO Tristate Channels</u> VI while the device is waiting for a Start trigger causes the channel to go to its Initial state.

This property is valid only for generation sessions.

Tristate (24)	<ul> <li>Sets the channel to a high-impedance state.</li> <li>Note NI 656x devices do not support the tristate Initial state.</li> </ul>
Logic high (1)	Sets the channel to a logic-high (high level) state.
Logic low (0)	Sets the channel to a logic-low (low level) state.
Hold last value (27)	The channel retains its previous value.

Data Type	Vilnt32
Permissions	R/W
Channel Based	Yes

# **Dynamic Generation:Idle State**

#### Short Name: IdleState

Specifies a dynamic generation channel state while the device is idle. The following conditions cause the Idle state to become active:

- The generation session completes normally.
- The generation session pauses from an active Pause trigger.
- The generation session terminates because of an underflow error.
- Channels explicitly disabled with the <u>niHSDIO Tristate Channels</u> VI remain disabled, but the channel data value changes internally. Re-enabling a channel with the <u>niHSDIO Tristate Channels</u> VI while the device is idle causes the channel to go into an Idle state.

This property is valid only for generation sessions.

Tristate (24)	Sets the channel to a high-impedance state. <b>Note</b> NI 656 <i>x</i> devices do not support the	
	tristate Idle state.	
Logic high (1)	Sets the channel to a logic-high (high level) state.	
Logic low (0)	Sets the channel to a logic-low (low level) state.	
Hold last value (27)	The channel retains its previous value.	

Data Type	Vilnt32
Permissions	R/W
Channel Based	Yes

# **Dynamic Generation:Drive Type**

#### Short Name: DriveType

Specifies what the data channels generate when set to logic 1. Using the <u>open collector</u> setting to generate a Z is useful for wired logic buses, such as I²C or SMBus.



**Notes** <u>NI 656x</u> devices only support the active drive setting. NI-HSDIO returns an error if you try to configure the channels on these devices for open collector generation.

<u>NI 654x</u> devices support open collector generation only for <u>static</u> <u>generation</u>. NI-HSDIO returns an error if you try to configure the channels on these devices for open collector dynamic generation.

This property is only valid for generation sessions.

Active drive (75)	The Generation Voltage High Level for the device is produced at the channel electronics when the Pattern Generation Engine generates a binary 1.
Open collector (76)	The channel electronics assume a high-impedance state when the Pattern Generation Engine generates a binary 1.

Data Type	Vilnt32
Permissions	R/W
Channel Based	Yes

# **Dynamic Generation:Repeat Mode**

#### Short Name: RepeatMode

Specifies whether or not to generate a single waveform continuously. This property is valid only when the <u>Generation Mode</u> property is set to **Waveform**; it is not used if you select **Scripted**.

If this property is set to **Finite**, then use the <u>Repeat Count</u> property to specify how many times the named waveform is generated.

This property is valid only for generation sessions.

Finite (16)	Calling the <u>niHSDIO Initiate</u> VI generates the named waveform a finite number of times. The number to repeat is defined by the <u>Repeat Count</u> property.
Continuous (17)	Calling <u>niHSDIO Init Generation Session</u> VI generates the named waveform continuously (until the <u>niHSDIO</u> <u>Abort</u> VI is called).

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Dynamic Generation:Repeat Count**

#### Short Name: RepeatCount

Specifies how many times to generate the waveform specified by the <u>Waveform To Generate</u> property. This property is valid only when the <u>Repeat Mode</u> property is set to **Finite**; it is not used when the <u>Repeat</u> <u>Mode</u> property is set to **Continuous**. This property is valid only when the <u>Generation Mode</u> property is set to **Waveform**—it is ignored if you select **Scripted**.

This property is only valid for generation sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Dynamic Generation: Generation Mode**

#### Short Name: GenerationMode

Use this property to specify whether to generate the waveform specified by the <u>Waveform To Generate</u> property or the script specified by the <u>Script To Generate</u> property upon calling the <u>niHSDIO Initiate</u> VI.

This property is valid only for generation sessions.

Waveform (14)	Calling the <u>niHSDIO Initiate</u> VI generates the named waveform represented by the <u>Waveform to Generate</u> property.
Scripted (15)	Calling <u>niHSDIO Initiate</u> generates the script represented by the <u>Script to Generate</u> property.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Dynamic Generation:Waveform to Generate**

#### Short Name: WaveformToGenerate

Specifies which named waveform in onboard memory is generated upon calling the <u>niHSDIO Initiate</u> VI when the <u>Generation Mode</u> is set to **Waveform**. If this attribute is not set to a valid waveform name and more than one waveform is in onboard memory, you receive an error upon calling <u>niHSDIO Initiate</u>. If only one waveform is in onboard memory and this property is set to "" (empty string), then that waveform is generated upon calling the <u>niHSDIO Initiate</u> VI.

This property is ignored when the <u>Generation Mode</u> property is set to **Scripted**, since the <u>Script To Generate</u> property defines the sequence of waveforms to generate.

This property is valid only for generation sessions.

Data Type	ViString
Permissions	R/W
Channel Based	No

# **Dynamic Generation:Script to Generate**

#### Short Name: ScriptToGenerate

Specifies which script in onboard memory is generated upon calling the niHSDIO Initiate VI when the Generation Mode property is set to Scripted. If this property is not set to a valid script and more than one script is in onboard memory, you receive an error upon calling niHSDIO Initiate. If only one script is in onboard memory and this property is set to "" (empty string), then that script is generated upon calling the niHSDIO Initiate VI.

This property is ignored when <u>Generation Mode</u> is set to **Waveform**, since the <u>Waveform To Generate</u> property defines which waveform to generate.

This property is valid only for generation sessions.

Data Type	ViString
Permissions	R/W
Channel Based	No

# Dynamic Generation:Data Transfer:Streaming:Enable

Short Name: StreamingEnable

Enables streaming of data from host memory to the device.

This property is valid only for dynamic generation sessions.
Data Type	ViBoolean
Permissions	R/W
Channel Based	No
High-Level VI	None

## Dynamic Generation:Data Transfer:Streaming:Streaming Waveform Name

Short Name: StreamingWaveformName

Specifies the name of the waveform for streaming. Use this property in conjunction with the <u>Streaming Enable</u> property.

Note You cannot stream an unnamed waveform.

Data Type	ViString
Permissions	R/W
Channel Based	No
High-Level VI	None

## Dynamic Generation:Data Transfer:Streaming:Space Available in Streaming Waveform

Short Name: SpaceAvailInStreamingWfrm

Specifies the space (in samples) available in the streaming waveform.

This property is valid only when streaming.

Data Type	Vilnt32
Permissions	RO
Channel Based	No
High-Level VI	None

# Dynamic Generation:Data Transfer:Direct DMA:Enable

Short Name: DirectDMAEnable

Enables direct DMA.

This property is valid only for dynamic generation sessions.

Data Type	ViBoolean
Permissions	R/W
Channel Based	No
High-Level VI	None

# Dynamic Generation:Data Transfer:Direct DMA:Window Size (in bytes)

Short Name: DirectDMAWindowSize

Specifies the direct DMA window size (in bytes).

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Dynamic Generation:Data Transfer:Direct DMA:Window Address

Short Name: DirectDMAWindowAddress

Specifies the start address for the direct DMA window.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Dynamic Generation:Data Transfer:Data Transfer Block Size

Short Name: DataTransferBlockSize

Specifies the number of samples to download to onboard memory at one time.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Timing:Sample Clock:Rate

#### Short Name: SampClk.Rate

Specifies the Sample clock rate.



**Note** You must set this property even when you supply an external clock because NI-HSDIO uses this property for a number of reasons, including optimal error checking and certain pulse width selections.

Units: hertz

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Timing:Sample Clock:Source

#### Short Name: SampClk.Source

Specifies the Sample clock source.

OnBoardClock	The device will use the onboard oscillator.
ClkIn	The device will use the clock present at the front panel CLK IN SMB jack connector.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.
STROBE	The device will use the clock present at the STROBE channel of the DDC connector.           Note         STROBE is valid only for acquisition operations.

Data Type	ViString
Permissions	R/W
Channel Based	No

# Timing:Sample Clock:Impedance

#### Short Name: SampClk.Impedance

Use this property to program the device input impedance when the Sample clock is supplied through the device front panel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Timing:Sample Clock:Export:Output Terminal

Short Name: ExportedSampClk.OutputTerm

Use this property to export the Sample clock to the specified terminal.

None	The signal is not exported.
ClkOut	The device will export the signal present to the front panel CLK OUT SMB jack connector.
DDC_ClkOut	The device will export the signal to the DDC CLK OUT channel in the front panel DDC connector.

Data Type	ViString
Permissions	R/W
Channel Based	No

# Timing:Sample Clock:Export:Mode

#### Short Name: ExportedSampClk.Mode

Specifies the position of the exported Sample clock relative to the Sample clock used by the device. When the Sample clock rate is set to less than 25 MS/s, this property must not be set to **Delayed**.

Noninverted (21)	The device exports the Sample clock without modifications.
Inverted (22)	The device inverts the Sample clock prior to exporting it.
Delayed (23)	The device delays the Sample clock prior to exporting it. Use the Exported Sample Clock Delay property to specify the delay value.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## **Timing:Sample Clock:Export:Delay**

#### Short Name: ExportedSampClk.Delay

Use this property to specify the delay of the exported Sample clock relative to the Sample clock used by the device. This property is relevant only when Exported Sample Clock Mode is set to Delayed. Otherwise, this property is ignored. This property is specified as fraction of the Sample clock interval, that is, as fraction of (1/Sample Clock Rate).

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Timing:Ref Clock:Rate

#### Short Name: RefClk.Rate

Specifies the rate of the Reference clock. 10 MHz is the only valid value for this property.

This property is ignored when <u>Reference Clock Source</u> is set to **None**.

Units: hertz

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Timing:Ref Clock:Source

#### Short Name: RefClk.Source

Specifies the Reference clock source.

None	The device will not use a Reference clock.
ClkIn	The device will use the clock present at the front panel CLK IN SMB jack connector.
PXI_CLK10	The device will use the PXI_CLK10 signal, which is present on the PXI backplane. This selection is valid only for PXI devices.
RTSI7	The device will use the signal present on RTSI trigger line 7. This selection is valid only for PCI devices.

Data Type	ViString
Permissions	R/W
Channel Based	No

# Timing:Ref Clock:Impedance

#### Short Name: RefClk.Impedance

Specifies the input impedance of the Reference clock when it is supplied through the device front panel. Valid values are 50 or 1000.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Timing:Ref Clock:Export Output Terminal

Short Name: ExportedRefClk.OutputTerm

Use this property to export the Reference clock to the specified terminal.

None	The signal is not exported.
ClkOut	The devices will use the signal present at the front panel
	CLK OUT SMB jack connector.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Timing:Onboard Ref Clock:Export Output Terminal

Short Name: ExportedOnboardRefClk.OutputTerm

Use this property to export the <u>Onboard Reference clock</u> to the specified terminal.

None	The signal is not exported.	
RTSI7	The signal is exported to RTSI 7.	
	<b>Note</b> The Onboard Reference clood	ck is only available on PCI

Data Type	ViString
Permissions	R/W
Channel Based	No

## **Timing:Data Position:Position**

#### Short Name: DataPos.Position

Specifies which edge of the Sample clock signal is used to time the acquisition or generation. You can also configure the device to acquire or generate data at a configurable delay past each rising edge of the Sample clock.

Sample clock rising edge (18)	The device samples or generates data on the Sample clock rising edge.
Sample clock falling edge (19)	The device samples or generates data on the Sample clock falling edge.
Delay from sample clock rising edge (20)	<ul> <li>The device samples or generates data with a delay from the Sample clock rising edge. Specify the delay using the Data Position Delay property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.</li> <li>Note To configure a delay on NI 656x devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.</li> </ul>
Data Type	Vilnt32
---------------	---------
Permissions	R/W
Channel Based	Yes

# Timing:Data Position:Delay

#### Short Name: DataPos.Delay

Specifies the delay after the Sample clock rising edge when the device acquires or generates a new data sample. Data delay is expressed as a fraction of the clock period (for example, a fraction of 1/<u>Sample Clock</u> <u>Rate</u>). This property is relevant only when <u>Data Position</u> is set to **Delay from Sample clock rising edge**.



**Note** To configure a delay on NI 656*x* devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.

Data Type	ViReal64
Permissions	R/W
Channel Based	Yes

## Timing:Advanced:Oscillator Phase DAC Value

#### Short Name: OscillatorPhaseDacValue

Use this attribute to phase shift the PLL circuit of the On Board Clock source. You can use this attribute to align the Sample clock of this device with another device that shares the same Reference clock. This property is not valid if <u>Ref Clock Source</u> is set to None. The valid range for this attribute is 0 to 4,095.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## Timing:Advanced:Exported Sample Clock Offset

Short Name: ExportedSampleClk.Offset

Use this attribute to offset the exported clock by a fixed time. Refer to Dynamic Generation Timing Diagrams for your <u>device</u> for more information about changing this value.

Valid values for this ViReal64 are 2.5e-9 and 0 for the  $\frac{NI 654x}{655x}$  devices and 1.6e-9 for the  $\frac{NI 656x}{656x}$  devices.

Units: seconds

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Triggers:Start:Type

#### Short Name: StartTrig.Type

Specifies whether you want the <u>Start trigger</u> to be a <u>digital edge</u>, <u>pattern</u> <u>match</u>, or <u>software</u> trigger.

<b>None</b> (28)	The data operation starts immediately after you call the niHSDIO Initiate VI.
<b>Digital edge</b> (29)	The data operation does not start until a digital edge is detected. The source of the digital edge is specified with the <u>Digital Edge Start Trigger Source</u> property, and the active edge is specified with the <u>Digital Edge Start Trigger Edge</u> property.
<b>Software</b> (32)	The data operation does not start until a software trigger occurs. You can assert the software trigger by calling the <u>niHSDIO Send Software Edge Trigger</u> VI and selecting <b>Start</b> <b>Trigger</b> as the <b>trigger</b> parameter.
Pattern match (31)	The data operation does not take effect until a specific data pattern matching condition is met. Configure the condition by setting the <u>Start Trigger Pattern Match Pattern</u> and <u>Start</u> <u>Trigger Pattern Match Trigger When</u> properties. This value is valid only for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Triggers:Start:Digital Edge:Source**

#### Short Name: StartTrig.DigEdge.Source

Specifies the source terminal for the Start trigger. This property is used only when <u>Start Trigger Type</u> is set to **Digital edge**.

PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
PXI_Trig7	PXI trigger line 7. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
	l

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.
RTSI7	RTSI trigger line 7. This selection is available only for PCI devices.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Triggers:Start:Digital Edge:Edge

Short Name: StartTrig.DigEdge.Edge

Specifies the active edge for the Start trigger. This property is used only <u>Start Trigger Type</u> is set to **Digital edge**.

**Rising edge** (12) Rising-edge trigger

Falling edge (13) Falling-edge trigger

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Triggers:Start:Digital Edge:Position**

#### Short Name: StartTrig.Position

Specifies the position where the <u>Start trigger</u> is latched, relative to the <u>Sample clock</u>. Trigger voltages and positions are only relevant if the trigger source is a front panel connector.

Sample clock rising edge (18)	The device samples or generates data on the Sample clock rising edge.
Sample clock rising edge (19)	The device samples or generates data on the Sample clock falling edge.
Delay from Sample clock rising edge (20)	The device samples or generates data with a delay from the Sample clock rising edge. Specify the delay using the <u>Data</u> <u>Position Delay</u> property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Start:Digital Edge:Terminal Configuration

Short Name: StartTrig.DigEdge.TermConfig

Specifies whether the <u>Start trigger</u> terminal is configured for <u>single-ended</u> or <u>LVDS</u> operation. Valid values for this property vary by <u>device</u>. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Start:Digital Edge:Impedance

Short Name: StartTrig.DigEdge.Impedance

Specifies the impedance on the channel configured for the digital edge Start trigger.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in <u>LVDS</u> <u>terminal configuration</u> 10000 in single-ended <u>terminal configuration</u>

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Start:Pattern Match:Pattern

Short Name: StartTrig.PatMatch.Pattern

Sets the pattern match mask for the <u>Start trigger</u>. This property is used when <u>Start Trigger Type</u> is set to **Pattern match**.

The pattern is a string of characters representing the entire pattern to match. Each character corresponds to a particular channel.

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge
- Spaces are ignored, and are useful for readability to segment long patterns

The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are 0 and channels 2-7 are 1.

The values seen by pattern matching are affected by the <u>Data</u> <u>Interpretation</u> property.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Triggers:Start:Pattern Match:Trigger When

**Short Name:** StartTrig.PatMatch.TrigWhen

Specifies whether a <u>pattern match</u> <u>Start trigger</u> asserts when a particular pattern is matched or not matched. This property is valid only for acquisition sessions.

Pattern matches (36)	The trigger asserts when the pattern matches.
Pattern does not match (37)	The trigger asserts when the pattern does not match.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Start:Export:Output Terminal

Short Name: ExportedStartTrig.OutputTerm

Specifies the destination terminal for exporting the <u>Start trigger</u>.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.

RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.	
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.	

Data Type	ViString
Permissions	R/W
Channel Based	No

# Triggers:Start:Export:Terminal Configuration

#### Short Name: ExportedStartTrig.TermConfig

Specifies whether the Start trigger output terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Ref:Type

#### Short Name: RefTrig.Type

Specifies the Reference trigger type. Depending on this property value, more properties may need to be set to fully configure the trigger.

<b>None</b> (28)	The device does not use a Reference trigger. The data operation starts immediately after you call the <u>niHSDIO</u> <u>Initiate</u> VI.
Digital edge (29)	The data operation does not start until a digital edge is detected. The source of the digital edge is specified with the <u>Digital Edge Ref Trigger Source</u> property, and the active edge is specified with <u>Digital Edge Ref Trigger Edge</u> property.
<b>Software</b> (32)	The data operation does not start until a software trigger occurs. You can assert the software trigger by calling niHSDIO Send Software Edge Trigger VI and selecting Start Trigger as the trigger parameter.
Pattern match (31)	The data operation does not take effect until a specific data pattern matching condition is met. Configure the condition by setting the <u>Pattern Match Ref Trigger Pattern</u> and <u>Pattern</u> <u>Match Ref Trigger Trigger When</u> properties. This value is valid only for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## **Triggers:Ref:Pretrigger Samples Per Record**

#### Short Name: RefTrig.PretrigSamples

Specifies the number of pretrigger samples, which are the samples acquired before the Reference trigger is received, to be acquired per record. The number of pretrigger samples cannot be greater than the <u>Samples Per Record</u> property.

This property is valid only for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Ref:Digital Edge:Edge

Short Name: RefTrig.DigEdge.Edge

Specifies the active edge for the Reference trigger. This property is used when <u>Reference Trigger Type</u> is set to **Digital edge**.

**Rising edge** (12) Rising-edge trigger

Falling edge (13) Falling-edge trigger

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Ref:Digital Edge:Source

#### Short Name: RefTrig.DigEdge.Source

Specifies the source terminal for the Reference trigger. This property is used only when <u>Reference Trigger Type</u> is set to **Digital edge**.

PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
PXI_Trig7	PXI trigger line 7. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
1	
RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.
RTSI7	RTSI trigger line 7. This selection is available only for PCI devices.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

Data Type	ViString
Permissions	R/W
Channel Based	No

# **Triggers:Ref:Digital Edge:Position**

#### Short Name: RefTrig.Position

Specifies the position where the Reference trigger is latched, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a front panel connector.

This property is valid only for acquisition sessions.

Sample clock rising edge (18)	The device samples or generates data on the Sample clock rising edge.
Sample clock rising edge (19)	The device samples or generates data on the Sample clock falling edge.
Delay from Sample clock rising edge (20)	The device samples or generates data with a delay from the Sample clock rising edge. Specify the delay using the Data Position Delay property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Ref:Digital Edge:Terminal Configuration

Short Name: RefTrig.DigEdge.TermConfig

Specifies whether the Reference trigger terminal is configured for singleended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Ref:Digital Edge:Impedance

Short Name: RefTrig.DigEdge.Impedance

Specifies the impedance on the channel configured for the digital edge Reference trigger.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in LVDS terminal configuration
	10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Ref:Pattern Match:Pattern

#### Short Name: RefTrig.PatMatch.Pattern

Sets the pattern match mask for the Reference trigger. This property is used when <u>Reference Trigger Type</u> is set to **Pattern match**.

The pattern is a string of characters representing the entire pattern to match. Each character corresponds to a particular channel.

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge
- Spaces are ignored, and are useful for readability to segment long patterns

The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are 0 and channels 2-7 are 1.

The values seen by pattern matching are affected by the <u>Data</u> <u>Interpretation</u> property.

Data Type	ViString
Permissions	R/W
Channel Based	No

# Triggers:Ref:Pattern Match:Trigger When

Short Name: RefTrig.PatMatch.TrigWhen

Specifies whether a pattern match Reference trigger asserts when a particular pattern is matched or not matched. This property is valid only for acquisition sessions.

Pattern matches (36)	The trigger asserts when the pattern matches.
Pattern does not match (37)	The trigger asserts when the pattern does not match.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Ref:Export:Output Terminal

Short Name: ExportedRefTrig.OutputTerm

Specifies the destination terminal for exporting the Reference trigger.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
I	

RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.	
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.	

Data Type	ViString
Permissions	R/W
Channel Based	No

# **Triggers:Ref:Export:Terminal Configuration**

#### Short Name: ExportedRefTrig.TermConfig

Specifies whether the exported Reference trigger output terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
<b>Single-Ended</b> (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Ref:Advanced:Start to Reference Trigger Holdoff

Short Name: StartToRefHoldoff

Specifies the amount of time after the <u>Start trigger</u> before the <u>Reference</u> <u>trigger</u> is recognized.

This attribute is especially useful when you want each device in a multidevice situation to recognize the Reference trigger at the same time, though the Reference trigger is shared among devices and each device has a different pretrigger count.

Units: seconds

Data Type	ViReal64
Permissions	R/W
Channel Based	No

## Triggers:Ref:Advanced:Reference to Reference Trigger Holdoff

Short Name: RefToRefHoldoff

Use this property to specify the amount of time until the next record's Reference trigger can be recognized.

This property is especially useful when you want each device in a multidevice situation to recognize the Reference trigger at the same time, though the Reference trigger is shared among devices and each device has a different record size.

Units: seconds

Data Type	ViReal64
Permissions	R/W
Channel Based	No

# Triggers:Advance:Type

### Short Name: AdvanceTrig.Type

Specifies whether you want the <u>Advance trigger</u> to be a digital edge, pattern match, or software trigger.

<b>None</b> (28)	No Advance trigger is configured.
<b>Digital edge</b> (29)	The Advance trigger is not asserted until a digital edge is detected. The source of the digital edge is specified with the Digital Edge Advance Trigger Source property, and the active edge is specified with the Digital Edge Advance Trigger Edge property.
<b>Software</b> (32)	The Advance trigger is not asserted until a software trigger occurs. You can assert the software trigger by calling the <u>niHSDIO Send Software Edge Trigger</u> VI with and selecting <b>Start Trigger</b> as the <b>trigger</b> parameter.
Pattern match (31)	The Advance trigger is not asserted until a specific data pattern matching condition is met. Configure the condition by setting the <u>Advance Trigger Pattern Match Pattern</u> and <u>Advance Trigger Pattern Match Trigger When</u> properties. This value is valid only for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Advance:Digital Edge:Source

Short Name: AdvanceTrig.DigEdge.Source

Specifies the source terminal for the <u>Advance trigger</u>. This property is used only when <u>Advance Trigger Type</u> is set to **Digital edge**.

PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
PXI_Trig7	PXI trigger line 7. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
1	

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.
RTSI7	RTSI trigger line 7. This selection is available only for PCI devices.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

Data Type	ViString
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Advance:Digital Edge:Edge

Short Name: AdvanceTrig.DigEdge.Edge

Specifies the active edge for the <u>Advance trigger</u>. This property is used only when <u>Advance Trigger Type</u> is set to **Digital edge**.

**Rising edge** (12) Rising-edge trigger

Falling edge (13) Falling-edge trigger

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Advance:Digital Edge:Position

Short Name: AdvanceTrig.DigEdge.Position

Specifies the position where the <u>Advance trigger</u> is latched, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a front panel connector.

Sample clock rising edge (18)	The device advances to the next record on the Sample clock rising edge.
Sample clock falling edge (19)	The device advances to the next record on the Sample clock falling edge.
Delay from Sample clock rising edge (20)	The device advances to the next record after a delay from the Sample clock rising edge. Specify the delay using the <u>Data</u> <u>Position Delay</u> property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Advance:Digital Edge:Terminal Configuration

Short Name: AdvanceTrig.DigEdge.TermConfig

Specifies whether the <u>Advance trigger</u> terminal is configured for <u>single-ended</u> or <u>LVDS</u> operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Advance:Digital Edge:Impedance

Short Name: AdvanceTrig.DigEdge.Impedance

Specifies the impedance on the channel configured for the <u>digital edge</u> <u>Advance trigger</u>.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in LVDS terminal configuration
	10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Advance:Pattern Match:Pattern

Short Name: AdvanceTrig.PatMatch.Pattern

Sets the pattern match mask for the Advance trigger. This property is used when <u>Advance Trigger Type</u> is set to **Pattern match**.

The pattern is a string of characters representing the entire pattern to match. Each character corresponds to a particular channel.

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge
- Spaces are ignored, and are useful for readability to segment long patterns

The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are 0 and channels 2-7 are 1.

The values seen by pattern matching are affected by the <u>Data</u> <u>Interpretation</u> property.

Data Type	ViString
Permissions	R/W
Channel Based	No
High-Level VI	None
## Triggers:Advance:Pattern Match:Trigger When

Short Name: AdvanceTrig.PatMatch.TrigWhen

Specifies whether a pattern match <u>Advance trigger</u> asserts when a particular pattern is matched or not matched. This property is valid only for acquisition sessions.

Pattern matches (36)	The trigger asserts when the pattern matches.
Pattern does not match (37)	The trigger asserts when the pattern does not match.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Advance:Export:Output Terminal

Short Name: ExportedAdvanceTrig.OutputTerm

Specifies the destination terminal for exporting the Advance trigger.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.

RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.	
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.	

Data Type	ViString
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Advance:Export:Terminal Configuration

Short Name: ExportedAdvanceTrig.TermConfig

Specifies whether the <u>Advance trigger</u> output terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Script:Type

#### Short Name: ScriptTrig.Type

Specifies the Script trigger type. Depending upon the value of this attribute, more attributes may be needed to fully configure the trigger.

This property is only valid for generation sessions.

<b>None</b> (28)	The device does not use a Script trigger. The data operation starts immediately after you call the <u>niHSDIO</u> <u>Initiate</u> VI.
Digital edge (29)	The data operation does not start until a digital edge is detected. The source of the digital edge is specified with the <u>Digital Edge Script Trigger Source</u> property, and the active edge is specified with the <u>Digital Edge Script</u> <u>Trigger Edge</u> property.
Digital level (30)	The Script trigger is active when the level of the Script trigger matches the desired level. The source of the Script trigger is specified with the <u>Digital Level Script</u> <u>Trigger Source</u> property, and the desired level is specified with the <u>Digital Level Script Trigger When</u> property.
Software (32)	The data operation does not start until a software trigger occurs. You can assert the software trigger by calling niHSDIO Send Software Edge Trigger VI and selecting Start Trigger as the trigger parameter.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## Triggers:Script:Digital Edge:Source

Short Name: ScriptTrig.DigEdge.Source

Specifies the source terminal for the Script trigger. This property is used when <u>Script Trigger Type</u> is set to **Digital edge**.

PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
PXI_Trig7	PXI trigger line 7. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
ļ	1

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.
RTSI7	RTSI trigger line 7. This selection is available only for PCI devices.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Triggers:Script:Digital Edge:Edge

Short Name: ScriptTrig.DigEdge.Edge

Specifies the active edge for the Script trigger. This property is used when <u>Script Trigger Type</u> is set to **Digital edge**.

**Rising edge** (12) Rising-edge trigger

Falling edge (13) Falling-edge trigger

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Script:Digital Edge:Terminal Configuration

Short Name: ScriptTrig.DigEdge.TermConfig

Specifies whether the Script trigger terminal is configured for singleended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Script:Digital Edge:Impedance

Short Name: ScriptTrig.DigEdge.Impedance

Specifies the impedance on the channel configured for the digital edge Script trigger.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in LVDS terminal configuration
	10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Script:Digital Level:Source

Short Name: ScriptTrig.DigLevel.Source

Specifies the source terminal for the Script trigger. This property is used when <u>Script Trigger Type</u> is set to **Digital level**.

PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
PXI_Trig7	PXI trigger line 7. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
ļ	1

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.
RTSI7	RTSI trigger line 7. This selection is available only for PCI devices.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Triggers:Script:Digital Level:Trigger When

Short Name: ScriptTrig.DigLevel.TrigWhen

Specifies the active level for the Script trigger. This property is used when <u>Script Trigger Type</u> is set to **Digital level**.

High (34) The data operation is paused when the trigger is high level.Low (35) The data operation is paused when the trigger is low level.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Script:Digital Level:Terminal Configuration

Short Name: ScriptTrig.DigLevel.TermConfig

Specifies whether the Script trigger terminal is configured for singleended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Script:Digital Level:Impedance

Short Name: ScriptTrig.DigLevel.Impedance

Specifies the impedance on the channel configured for the digital level Script trigger.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in <u>LVDS</u> <u>terminal configuration</u> 10000 in single-ended <u>terminal configuration</u>

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Script:Export:Output Terminal

Short Name: ExportedScriptTrig.OutputTerm

Specifies the output terminal for the exported Script trigger.

Setting this attribute to an empty string means that when you commit the session, the signal is removed from that terminal and, if possible, the terminal is tristated. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

The signal is not exported. None PFI 0 on the front panel SMB jack connector. **PFIO** PFI1 PFI 1 on the front panel DDC connector. PFI 2 on the front panel DDC connector. PFI2 PFI3 PFI 3 on the front panel DDC connector. PXI Trig0 PXI trigger line 0. This selection is available only for PXI devices. PXI Trig1 PXI trigger line 1. This selection is available only for PXI devices. PXI_Trig2 PXI trigger line 2. This selection is available only for PXI devices. PXI_Trig3|PXI trigger line 3. This selection is available only for PXI devices. PXI_Trig4 PXI trigger line 4. This selection is available only for PXI devices. PXI_Trig5 PXI trigger line 5. This selection is available only for PXI devices. PXI_Trig6 PXI trigger line 6. This selection is available only for PXI devices. RTSI trigger line 0. This selection is available only for PCI RTSI0 devices. RTSI trigger line 1. This selection is available only for PCI RTSI1 devices. RTSI trigger line 2. This selection is available only for PCI RTSI2

This attribute is valid only for generation sessions.

	devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Triggers:Script:Export:Terminal Configuration

Short Name: ExportedScriptTrig.TermConfig

Specifies whether the Script trigger output terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## Triggers:Pause:Type

#### Short Name: PauseTrig.Type

Specifies the Pause trigger type. Depending upon the value of this property, more properties may be needed to fully configure the trigger.

None (28)	No Pause trigger is configured.
Digital level (30)	The Pause trigger is active when the level of the Pause trigger matches the desired level. The source of the Pause trigger is specified with the <u>Digital Level Pause Trigger</u> <u>Source</u> property, and the desired level is specified with the <u>Digital Level Pause Trigger When</u> property.
Pattern match (31)	The data operation does not take effect until a specific data pattern matching condition is met. Configure the condition by setting <u>Pattern Match Pause Trigger Pattern</u> and <u>Pattern</u> <u>Match Pause Trigger When</u> . This is valid only for acquisition sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## Triggers:Pause:Digital Level:Source

Short Name: PauseTrig.DigLevel.Source

Specifies the source terminal for the Pause trigger.

This property only applies to acquisition operations.

PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
PXI_Trig7	PXI trigger line 7. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.
RTSI7	RTSI trigger line 7. This selection is available only for PCI devices.
PXI_STAR	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.
Data Type	ViString
---------------	----------
Permissions	R/W
Channel Based	No

## Triggers:Pause:Digital Level:Trigger When

**Short Name:** PauseTrig.DigLevel.TrigWhen

Specifies the active level for pausing the dynamic operation.

**High** (34) The data operation is paused when the trigger is high level.

**Low** (35) The data operation is paused when the trigger is low level.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

### Triggers:Pause:Digital Level:Position

Short Name: PauseTrig.DigLevel.Position

Specifies the position where the start trigger is latched, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a front panel connector.

Sample clock rising edge (18)	The Pause trigger asserts on the Sample clock rising edge.
Sample clock rising edge (19)	The Pause trigger asserts on the Sample clock falling edge.
Delay from Sample clock rising edge (20)	The Pause trigger asserts after a delay from the Sample clock rising edge. Specify the delay using the <u>Data Position</u> <u>Delay</u> property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# Triggers:Pause:Digital Level:Terminal Configuration

Short Name: PauseTrig.DigLevel.TermConfig

Specifies whether the Pause trigger terminal is configured for singleended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

### Triggers:Pause:Digital Level:Impedance

Short Name: PauseTrig.DigLevel.Impedance

Specifies the impedance on the channel configured for the digital level Pause trigger.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in LVDS terminal configuration
	10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

Data Type	ViReal64
Permissions	R/W
Channel Based	No
High-Level VI	None

### Triggers:Pause:Pattern Match:Pattern

Short Name: PauseTrig.PatMatch.Pattern

Sets the pattern match mask for the Pause trigger. This property is used when <u>Pause Trigger Type</u> is set to **Pattern match**.

The pattern is a string of characters representing the entire pattern to match. Each character corresponds to a particular channel.

- X or x: Ignore the channel
- 1: Match on a logic 1
- 0: Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge
- Spaces are ignored, and are useful for readability to segment long patterns

The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are 0 and channels 2-7 are 1.

The values seen by pattern matching are affected by the <u>Data</u> <u>Interpretation</u> property.

This property is only valid for acquisition sessions.

Data Type	ViString
Permissions	R/W
Channel Based	No

### Triggers:Pause:Pattern Match:Trigger When

Short Name: PauseTrig.PatMatch.TrigWhen

Specifies whether a pattern match Pause trigger asserts when a particular pattern is matched or not matched.

This property is valid only for acquisition sessions.

Pattern matches (36)	The trigger asserts when the pattern matches.
Pattern does not match (37)	The trigger asserts when the pattern does not match.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Triggers:Pause:Export:Output Terminal

Short Name: ExportedPauseTrig.OutputTerm

Specifies the output terminal for the exported Pause trigger.

This property is only valid for generation sessions.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
RTSI4	RTSI trigger line 4. This selection is available only for PCI

	devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.

Data Type	ViString
Permissions	R/W
Channel Based	No

## Triggers:Pause:Export:Terminal Configuration

Short Name: ExportedPauseTrig.TermConfig

Specifies whether the Pause trigger output terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## **Events:Ready For Start:Output Terminal**

Short Name: RdyForStartEvent.OutputTerm

Specifies the destination terminal for the Ready for Start event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.

Data Type	ViString
Permissions	R/W
Channel Based	No

### **Events:Ready For Start:Active Level**

Short Name: RdyForStartEvent.ActiveLvl

Specifies the output polarity of the Ready for Start Event.

Active high (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. The exported signal is low level while the event is deasserted.
Active low (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

### **Events:Ready For Start:Terminal Configuration**

#### Short Name: RdyForStartEvent.TermConfig

Specifies whether the Ready for Start event terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

### **Events:Ready For Advance:Output Terminal**

#### Short Name: RdyForAdvanceEvent.OutputTerm

Specifies the destination terminal for the Ready for Advance Event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

Data Type	ViString
Permissions	R/W
Channel Based	No
High-Level VI	None

### **Events:Ready For Advance:Active Level**

Short Name: RdyForAdvanceEvent.ActiveLvl

Specifies the output polarity of the Ready for Advance Event.

Active high (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. The exported signal is low level while the event is deasserted.
Active low (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

### Events:Ready For Advance:Terminal Configuration

Short Name: RdyForAdvanceEvent.TermConfig

Specifies whether the Ready for Advance event terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# **Events:End Of Record:Output Terminal**

#### Short Name: EndOfRecEvent.OutputTerm

Specifies the destination terminal for the End of Record Event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
	1

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.

Data Type	ViString
Permissions	R/W
Channel Based	No
High-Level VI	None

## **Events:End Of Record:Pulse Polarity**

Short Name: EndOfRecEvent.PulsePolarity

Specifies the output polarity of the Ready for Advance Event.

Active high (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. The exported signal is low level while the event is deasserted.
Active low (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None
# **Events: End Of Record: Terminal Configuration**

### Short Name: EndOfRecEvent.TermConfig

Specifies whether the End of Record event terminal is configured for single-ended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## **Events:Data Active:Output Terminal**

#### Short Name: DataActiveEvent.OutputTerm

Specifies the destination terminal for the Data Active Event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This attribute is valid only for generation sessions.

None The signal is not exported.

PFI0 PFI 0 on the front panel SMB jack connector.

PFI1 PFI 1 on the front panel DDC connector.

PFI2 PFI 2 on the front panel DDC connector.

PFI3 PFI 3 on the front panel DDC connector.

Data Type	ViString
Permissions	R/W
Channel Based	No

## **Events:Data Active:Active Level**

Short Name: DataActiveEvent.ActiveLvl

Specifies the output polarity of the <u>Data Active event</u>.

Active	The exported signal is low level while the event is deasserted.
high (10)	A high pulse occurs when the event asserts.
Active low (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## **Events:Data Active:Position**

#### Short Name: DataActiveEvent.Position

Specifies the position of the event relative to the Sample clock. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

Sample clock rising edge (18)	The event occurs on the Sample clock rising edge.
Sample clock rising edge (19)	The event occurs on the Sample clock falling edge.
Delay from Sample clock rising edge (20)	The event occurs after a delay from the Sample clock rising edge. Specify the delay using the <u>Data Position Delay</u> property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## **Events:Data Active:Terminal Configuration**

#### Short Name: DataActiveEvent.TermConfig

Specifies whether the Data Active event terminal is configured for singleended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

<b>LVDS</b> (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# **Events:Marker:Output Terminal**

Short Name: MarkerEvent.OutputTerm

Specifies the destination terminal for the Marker Event.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.
RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.

RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.	
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.	

Data Type	ViString
Permissions	R/W
Channel Based	No

# **Events:Marker:Pulse Polarity**

Short Name: MarkerEvent.PulsePolarity

Specifies the output polarity of the Marker Event.

This property is valid only for generation sessions.

Active high (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. The exported signal is low level while the event is deasserted.
Active low (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

## **Events:Marker:Position**

#### Short Name: MarkerEvent.Position

Specifies the position of the event relative to the Sample clock. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This property is valid only for generation sessions.

Sample clock rising edge (18)	The event occurs on the Sample clock rising edge.
Sample clock falling edge (19)	The event occurs on the Sample clock falling edge.
Delay from Sample clock rising edge (20)	The event occurs after a delay from the Sample clock rising edge. Specify the delay using the <u>Data Position Delay</u> property. This choice has more jitter than the rising or falling edge values. Certain devices have Sample clock frequency limitations on when a custom delay can be used. Refer to the device documentation for details.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# **Events:Marker:Terminal Configuration**

#### Short Name: MarkerEvent.TermConfig

Specifies whether the <u>Marker event</u> terminal is configured for singleended or LVDS operation. Valid values for this property vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

LVDS (64)	The terminal is configured for LVDS operation.
Single-Ended (65)	The terminal is configured for single-ended operation.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

# **Events:Sample Error:Output Terminal**

Short Name: SampleErrorEvent.OutputTerm

Specifies the destination terminal for the Sample Error event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

None	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PFI2	PFI 2 on the front panel DDC connector.
PFI3	PFI 3 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0. This selection is available only for PXI devices.
PXI_Trig1	PXI trigger line 1. This selection is available only for PXI devices.
PXI_Trig2	PXI trigger line 2. This selection is available only for PXI devices.
PXI_Trig3	PXI trigger line 3. This selection is available only for PXI devices.
PXI_Trig4	PXI trigger line 4. This selection is available only for PXI devices.
PXI_Trig5	PXI trigger line 5. This selection is available only for PXI devices.
PXI_Trig6	PXI trigger line 6. This selection is available only for PXI devices.
RTSI0	RTSI trigger line 0. This selection is available only for PCI devices.
RTSI1	RTSI trigger line 1. This selection is available only for PCI devices.
RTSI2	RTSI trigger line 2. This selection is available only for PCI devices.
RTSI3	RTSI trigger line 3. This selection is available only for PCI devices.

RTSI4	RTSI trigger line 4. This selection is available only for PCI devices.
RTSI5	RTSI trigger line 5. This selection is available only for PCI devices.
RTSI6	RTSI trigger line 6. This selection is available only for PCI devices.

Data Type	ViString
Permissions	R/W
Channel Based	No

## **Total Acquisition Memory Size**

#### Short Name: TotalAcqMemSize

Specifies the total onboard memory size for acquiring data. The number of samples is based on the default device <u>data width</u>.

If you configure your device to use a different data width, the total memory size is actually the value returned by this attribute multiplied by the quotient of the default data width divided by the configured data width. For example, if you configure 1-byte data width for a 2-byte device, the total acquisition memory size is twice the number of samples that is returned by this attribute.

Units: samples

Data Type	Vilnt32
Permissions	RO
Channel Based	No

## **Total Generation Memory Size**

#### Short Name: TotalGenMemSize

Specifies the total onboard memory size for generating data. The number of samples is based on the default device <u>data width</u>.

If you configure your device to use a different data width, the total memory size is actually the value returned by this attribute multiplied by the quotient of the default data width divided by the configured data width. For example, if you configure 1-byte data width for a 2-byte device, the total generation memory size is twice the number of samples that is returned by this attribute.

Units: samples

Data Type	Vilnt32
Permissions	RO
Channel Based	No

## **Serial Number**

**Short Name:** SerialNumber Returns the serial number of the device.

Data Type	ViString
Permissions	RO
Channel Based	No

# **Resource Descriptor**

#### Short Name: ResourceDescriptor

Specifies the resource descriptor used to identify the instrument in Measurement & Automation Explorer.

Data Type	Vilnt32
Permissions	RO
Channel Based	No

# Advanced:Data Width

#### Short Name: DataWidth

Specifies, in bytes, the size of a raw sample from the operation.

<b>1 byte</b> (1)	A raw sample is one byte. You can choose this value for NI 654 <i>x</i> /655 <i>x</i> /656 <i>x</i> .
<b>2 bytes</b> (2)	A raw sample is two bytes. You can choose this value for NI 654 <i>x</i> /655 <i>x</i> /656 <i>x</i> .
<b>4 bytes</b> (4)	A raw sample is four bytes. You can choose this value for NI 654 <i>x</i> /655 <i>x</i> .

Data Type	Vilnt32
Permissions	R/W for acquisition sessions, RO for generation sessions
Channel Based	No
High-Level VI	None

# **Advanced:Data Rate Multiplier**

#### Short Name: DataRateMultiplier

Specifies whether you want the device to acquire or generate in single data rate (SDR) mode or in double data rate (DDR) mode.

Single Data Rate (1)	In SDR mode, the device generates or acquires data on a single edge of the Sample clock. Therefore, you can generate or acquire data on the rising or falling edge of every Sample clock pulse or on a delayed version of the rising edge of the Sample clock.
Double Data Rate (2)	In DDR mode, the device generates or acquires data on both edges of the Sample clock. Therefore, you can generate or acquire data on every rising and falling edge of the Sample clock. Acquisition and generation sessions can be configured in DDR mode to acquire or generate the first data sample on the rising or falling edge of the clock or on a delayed version of the rising edge of the clock.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None

## **Advanced: Data Active Internal Route Delay**

Short name: DataActiveInternalRouteDelay

Configures the number of Sample clock cycles to delay the internal <u>Data</u> <u>Active event</u>. Internally routing a delayed version of this event is useful when you want to synchronize an acquisition trigger to the generation operation. Use this coarse delay together with the finer-resolution data delay to compensate for the round trip delay of data in stimulus/response operations.

You can configure the delayed Data Active event as the source for any acquisition trigger by manually entering DelayedDataActiveEvent as the trigger source parameter in the appropriate trigger configuration function or VI instance.

This attribute is only applicable in acquisition sessions.

Valid values for this attribute are 0 to 24.

Units: Sample clock cycles

Data Type	Vilnt32
Permissions	R/W
Channel Based	No
High-Level VI	None
# Advanced:Hardware Compare:Hardware Compare Mode

#### Short Name: HardwareCompare.Mode

Configures the device to compare expected data to actual data in realtime. This property must be set to the same value in both sessions.



**Note** To use this feature you must have an acquisition and a generation session running concurrently.

When you set this property to either **Stimulus And Expected Response** or **Expected Response Only**, the generation engine sends expected data to the acquisition session to compare against acquired data.

Use the Waveform Data Type (WDT) instance of the <u>niHSDIO Write</u> <u>Named Waveform</u> VI to write expected data to the device. The device drives any values of 0, 1, or Z in the waveform, while values of H, L, or X are treated as expected data values.

This property must be set before data is written to the device.

Disabled (77)	Comparison engine is disabled. Any attempts to write expected response data to the device results in an error.
Stimulus And Expected Response (78)	Device drives and compares data in parallel sessions, without any software reconfiguration. You can download waveforms with drive and/or compare data. All digital states supported in this mode.
Expected Response Only (79)	Device does not drive any data, it only acquires and compares. You cannot download drive data. A generation session downloads the expected waveform. Selecting this value pauses the generation lines to synchronize the acquisition and generation sessions.

Data Type	Vilnt32
Permissions	R/W
Channel Based	No

# Advanced:Hardware Compare:Sample Error Backlog

Short Name: HardwareCompare.SampleErrorBacklog

Returns the number of sample errors you can read using the <u>niHSDIO</u> <u>HWC Fetch Sample Errors</u> VI.

Data Type	Vilnt32
Permissions	RO
Channel Based	No

# Advanced:Hardware Compare:Number Of Sample Errors

Short Name: HardwareCompare.NumSampleErrors

Returns the total number of sample errors since the acquisition was initiated. Use this property along with the <u>Samples Compared</u> property to calculate the sample error rate.

Data Type	Vilnt32
Permissions	R0
Channel Based	No

# Advanced:Hardware Compare:Samples Compared

Short Name: HardwareCompare.SamplesCompared

Returns the total number of samples compared since the acquisition was initiated. Use this property, along with the <u>Number Of Sample Errors</u> property, to calculate the sample error rate.

Data Type	ViReal64
Permissions	R0
Channel Based	No

# Advanced:Hardware Compare:Filter Repeated Sample Errors

Short Name: HardwareCompare.FilterRepeatedErrors

Specifies whether the device stores and counts errors when the same error appears in consecutive samples. If this attribute is set to TRUE, the device only counts distinct errors. An error is defined as distinct if the expected response value and the actual sample error do not change over the same number of Sample clock cycles. The <u>Fetch Sample Errors</u> VI returns the number of clock cycles for which the repeated error occurred.

This property is helpful if your NI device clock rate is faster than your DUT clock rate. In this case, one error from the DUT could result in several identical errors on the device.

Data Type	ViBoolean	
Permissions	R/W	
Channel Based	No	

# Advanced:Hardware Compare:Samples Error Buffer Overflowed

Short Name: HardwareCompare.ErrorBufferOverflowed

Returns whether the FIFO used to store sample errors has overflowed. The NI 655x FIFO can contain 4,094 sample errors. If the FIFO overflows, the hardware stops storing error information for further errors, but it continues to compare data and count the sample errors encountered.

You can remove sample errors from the FIFO using the <u>niHSDIO HWC</u> <u>Fetch Sample Errors (U32)</u> VI. Removing sample errors creates room for additional sample errors to be stored in the FIFO.

Data Type	ViBoolean	
Permissions	RO	
Channel Based	No	

# **Alphabetical Property List and Default Values**

The following table lists the default values for each property you can configure for your device. An "N/A" in a table cell indicates that the listed property is not supported for that device. A dash indicates that the property does not have a default value or that it is a read-only property. "" is used in the following ways:

- In output terminal properties to indicate to the device not to export the relevant signal
- In trigger source properties to the device that the relevant trigger is not used

NI 654 <i>x</i> Default Value	NI 655 <i>x</i> Default Value	NI 656 <i>x</i> Default Value	Prope
			Active Channel
None	None	None	Triggers»Advaı
Active high	Active high	Active high	Events»Data A
			Events»Data A Terminal
Sample clock rising edge	Sample clock rising edge	Sample clock rising edge	Events»Data A
Single-Ended	Single-Ended	LVDS	Events»Data A Configuration
N/A	0	N/A	Advanced»Har Compare»Harc Mode
	NI 654x Default Value '''' None Active high '''' Sample clock rising edge Single-Ended	NI 654x Default ValueNI 655x Default Value""""""""NoneNoneActive highActive high""""Sample clock rising edgeSample clock rising edgeSingle-EndedSingle-EndedN/A0	NI 654x Default ValueNI 655x Default ValueNI 656x Default Value''''''''''''''''''''''''NoneNoneNoneActive high Active highActive high Active highActive high I'''''''''''''''Sample clock rising edgeSample clock rising edgeSample clock rising edgeSingle-EndedSingle-EndedLVDSN/A0N/A

• In dynamic and static channels to means "all channels"

Data Interpretation	High or low	High or low	High or low	Dynamic Acqui Interpretation
Data Position	Sample clock rising edge	Sample clock rising edge	Sample clock rising edge	Timing»Data P
Data Position Delay	0 %	0 %	0 %	Timing»Data P
<u>Data Rate</u> Multiplier	Single Data Rate	Single Data Rate	Single Data Rate	Advanced»Dat
<u>Data</u> <u>Transfer</u> <u>Block Size</u>	0	0	0	Dynamic Gene Transfer»Data
<u>Data Voltage</u> <u>High Level</u>	3.3 V logic family voltage levels. Refer to device specifications for more information.	3.3 V logic family voltage levels. Refer to device specifications for more information.	N/A	Voltage Levels:
<u>Data Voltage</u> <u>Low Level</u>	3.3 V logic family voltage levels. Refer to device specifications for more information.	3.3 V logic family voltage levels. Refer to device specifications for more information.	N/A	Voltage Levels:
<u>Data Width</u>	4 bytes	4 bytes	2 bytes	Advanced»Dat
Digital Edge Advance Trigger Edge	Rising edge	Rising edge	Rising edge	Triggers»Advar Edge»Edge
<u>Digital Edge</u> <u>Advance</u> <u>Trigger</u> <u>Impedance</u>	10000 Ω	10000 Ω	10000 Ω	Triggers»Advar Edge»Impedan
Digital Edge	Sample clock	Sample clock	Sample clock	Triggers»Advar

Advance Trigger Position	rising edge	rising edge	rising edge	Edge»Position
Digital Edge Advance Trigger Source				Triggers»Advar Edge»Source
Digital Edge Advance Trigger Terminal Configuration	Single-Ended	Single-Ended	LVDS	Triggers»Advar Edge»Terminal
<u>Digital Edge</u> <u>Ref Trigger</u> <u>Edge</u>	Rising edge	Rising edge	Rising edge	Triggers»Ref»E
<u>Digital Edge</u> <u>Ref Trigger</u> <u>Impedance</u>	10000 Ω	10000 Ω	10000 Ω	Triggers»Ref»E Edge»Impedan
<u>Digital Edge</u> <u>Ref Trigger</u> <u>Position</u>	Sample clock rising edge	Sample clock rising edge	Sample clock rising edge	Triggers»Ref»E
<u>Digital Edge</u> <u>Ref Trigger</u> <u>Source</u>			ш	Triggers»Ref»E
<u>Digital Edge</u> <u>Ref Trigger</u> <u>Terminal</u> <u>Configuration</u>	Single-Ended	Single-Ended	LVDS	Triggers»Ref»E Edge»Terminal
<u>Digital Edge</u> <u>Script</u> <u>Trigger Edge</u>	Rising edge	Rising edge	Rising edge	Triggers»Script
Digital Edge Script Trigger Impedance	10000 Ω	10000 Ω	10000 Ω	Triggers»Script Edge»Impedan
Digital Edge			1111	Triggers»Script

<u>Script</u> <u>Trigger</u> <u>Source</u>				Edge»Source
Digital Edge Script Trigger Terminal Configuration	Single-Ended	Single-Ended	LVDS	Triggers»Script Edge»Terminal
<u>Digital Edge</u> <u>Start Trigger</u> <u>Edge</u>	Rising edge	Rising edge	Rising edge	Triggers»Start»
<u>Digital Edge</u> <u>Start Trigger</u> Impedance	10000	10000	10000	Triggers»Start» Edge»Impedan
<u>Digital Edge</u> <u>Start Trigger</u> <u>Position</u>	Sample clock rising edge	Sample clock rising edge	Sample clock rising edge	Triggers»Start» Edge»Position
<u>Digital Edge</u> <u>Start Trigger</u> <u>Source</u>			III	Triggers»Start» Edge»Source
Digital Edge Start Trigger Terminal Configuration	Single-Ended	Single-Ended	LVDS	Triggers»Start» Edge»Terminal
Digital Level Pause Trigger Impedance	10000 Ω	10000 Ω	10000 Ω	Triggers»Paus Level»Impedar
Digital Level Pause Trigger Position	Sample clock rising edge	Sample clock rising edge	Sample clock rising edge	Triggers»Pause Level»Position
Digital Level Pause Trigger Terminal	Single-Ended	Single-Ended	LVDS	Triggers»Pause Level»Terminal

Configuration				
<u>Digital Level</u> <u>Pause</u> <u>Trigger</u> <u>Source</u>				Triggers»Pause Level»Source
<u>Digital Level</u> <u>Pause</u> Trigger <u>When</u>	High	High	High	Triggers»Pause Level»Trigger \
Digital Level Script Trigger Impedance	10000 Ω	10000 Ω	10000 Ω	Triggers»Pause Level»Impedar
Digital Level Script Trigger Terminal Configuration	Single-Ended	Single-Ended	LVDS	Triggers»Script Level»Terminal
<u>Digital Level</u> <u>Script</u> <u>Trigger</u> <u>Source</u>				Triggers»Script Level»Source
<u>Digital Level</u> <u>Script</u> <u>Trigger</u> <u>When</u>	High	High	High	Triggers»Script Level»Trigger \
<u>Direct DMA</u> <u>Enable</u>	FALSE	FALSE	FALSE	Dynamic Gene Transfer»Direct
Direct DMA Window Address	0	0	0	Dynamic Gene Transfer»Direct Address
<u>Direct DMA</u> <u>Window Size</u>	0	0	0	Dynamic Gene Transfer»Direct
Dynamic Channels				Dynamic Chanı

End of Record Event Output Terminal				Events»End of Terminal
<u>End of</u> <u>Record</u> <u>Event Pulse</u> <u>Polarity</u>	Active high	Active high	Active high	Events»End of Polarity
End of Record Event Terminal Configuration	Single-Ended	Single-Ended	LVDS	Events»End of Configuration
<u>Event</u> <u>Voltage High</u> <u>Level</u>	3.3 V logic family voltage levels. Refer to device specifications for more information.	3.3 V logic family voltage levels. Refer to device specifications for more information.	N/A	Voltage Levels:
<u>Event</u> Voltage Low Level	3.3 V logic family voltage levels. Refer to device specifications for more information.	3.3 V logic family voltage levels. Refer to device specifications for more information.	N/A	Voltage Levels:
Exported Advance Trigger Output Terminal				Triggers»Advaı Terminal
Exported Advance Trigger	Single-Ended	Single-Ended	LVDS	Triggers»Advar Configuration

Terminal Configuration				
Exported Onboard Ref Clock Output Terminal				Timing»Onboaı Output Termina
<u>Exported</u> <u>Pause</u> <u>Trigger</u> <u>Output</u> <u>Terminal</u>				Triggers»Pause Terminal
<u>Exported</u> <u>Pause</u> <u>Trigger</u> <u>Terminal</u> <u>Configuration</u>	Single-Ended	Single-Ended	LVDS	Triggers»Pause Configuration
Exported Ref Clock Output Terminal				Timing»Ref Clc Terminal
Exported Ref Trigger Output Terminal				Triggers»Ref»E Terminal
Exported Ref Trigger Terminal Configuration	Single-Ended	Single-Ended	LVDS	Triggers»Ref»E Configuration
<u>Exported</u> <u>Sample</u> <u>Clock Delay</u>	0 %	0 %	0 %	Timing»Sample Delay
<u>Exported</u> <u>Sample</u> Clock Mode	Noninverted	Noninverted	Noninverted	Timing»Sample Mode
<u>Exported</u> <u>Sample</u> <u>Clock Offset</u>	2.5 ns	2.5 ns	1.5 ns	Timing»Advanc Sample Clock (

Exported Sample Clock Output	m			Timing»Sample Output Termina
<u>Terminal</u>				<b>T</b>
Exported Script Trigger Output Terminal				Terminal
Exported Script Trigger Terminal Configuration	Single-Ended	Single-Ended	LVDS	Triggers»Script Configuration
<u>Exported</u> <u>Start Trigger</u> <u>Output</u> <u>Terminal</u>				Triggers»Start» Terminal
<u>Exported</u> <u>Start Trigger</u> <u>Terminal</u> <u>Configuration</u>	Single-Ended	Single-Ended	LVDS	Triggers»Start» Configuration
<u>Fetch</u> Backlog	—	—		Dynamic Acqui Backlog
Fetch Offset	0 S	0 S	0 S	Dynamic Acqui
<u>Fetch</u> <u>Relative To</u>	Most recent sample with no Reference trigger configured; Reference trigger with Reference trigger configured	Most recent sample with no Reference trigger configured; Reference trigger with Reference trigger configured	Most recent sample with no Reference trigger configured; Reference trigger with Reference trigger configured	Dynamic Acqui Relative To
<u>Filter</u>	N/A	FALSE	N/A	Advanced»Har

Repeated Sample Errors				Compare»Filte Errors
<u>Generation</u> <u>Mode</u>	Waveform	Waveform	Waveform	Dynamic Gene Mode
<u>Hardware</u> Compare Mode	Disabled	Disabled	Disabled	Advanced:Harc Compare:Hard Mode
Idle State	Hold last value	Hold last value	Hold last value	Dynamic Gene
Initial State	Hold last value	Hold last value	Hold last value	Dynamic Gene
<u>Input</u> Impedance	50 Ω	50 Ω	50 Ω	Dynamic Acqui Impedance
<u>Marker Event</u> <u>Output</u> <u>Terminal</u>				Events»Marker
<u>Marker Event</u> Position	Sample clock rising edge	Sample clock rising edge	Sample clock rising edge	Events»Marker
<u>Marker Event</u> <u>Pulse</u> Polarity	Active high	Active high	Active high	Events»Marker
<u>Marker Event</u> <u>Terminal</u> <u>Configuration</u>	Single-Ended	Single-Ended	LVDS	Events»Marker Configuration
<u>Number of</u> <u>Records to</u> <u>Acquire</u>	1	1	1	Dynamic Acqui Records to Acq
<u>Number of</u> <u>Sample</u> <u>Errors</u>				Advanced»Har Compare»Num Errors
<u>Oscillator</u> Phase DAC <u>Value</u>	0	0	0	Timing»Advanc DAC Value
Pattern	1111		1111	Triggers»Advar

<u>Match</u> <u>Advance</u> <u>Trigger</u> Pattern				Match»Pattern
Pattern Match Advance Trigger When	Pattern matches	Pattern matches	Pattern matches	Triggers»Advar Match»Trigger
<u>Pattern</u> <u>Match Pause</u> <u>Trigger</u> <u>Pattern</u>				Triggers»Pause Match»Pattern
<u>Pattern</u> <u>Match Pause</u> <u>Trigger</u> <u>When</u>	Pattern matches	Pattern matches	Pattern matches	Triggers»Pause Match»Trigger
<u>Pattern</u> <u>Match Ref</u> <u>Trigger</u> <u>Pattern</u>	1111			Triggers»Ref»F Match»Pattern
<u>Pattern</u> <u>Match Ref</u> <u>Trigger</u> <u>When</u>	Pattern matches	Pattern matches	Pattern matches	Triggers»Ref»F Match»Trigger
<u>Pattern</u> <u>Match Start</u> <u>Trigger</u> <u>Pattern</u>				Triggers»Start» Match»Pattern
<u>Pattern</u> <u>Match Start</u> <u>Trigger</u> <u>When</u>	Pattern matches	Pattern matches	Pattern matches	Triggers»Start» Match»Trigger
<u>Pause</u> Trigger Type	None	None	None	Triggers»Pause
Ready for	Active high	Active high	Active high	Events»Ready

<u>Advance</u> <u>Event Level</u> Active Level				Level
<u>Ready for</u> Advance Event Output Terminal				Events»Ready Terminal
<u>Ready for</u> <u>Advance</u> <u>Event</u> <u>Terminal</u> <u>Configuration</u>	Single-Ended	Single-Ended	LVDS	Events»Ready Edge»Impedan
<u>Ready for</u> <u>Start Event</u> <u>Level Active</u> <u>Level</u>	Active high	Active high	Active high	Events»Ready Level
<u>Ready for</u> <u>Start Event</u> <u>Output</u> <u>Terminal</u>				Events»Ready Terminal
<u>Ready for</u> <u>Start Event</u> <u>Terminal</u> <u>Configuration</u>	Single-Ended	Single-Ended	LVDS	Events»Ready Configuration
<u>Records</u> Done		—		Dynamic Acquisition»Fe
<u>Ref Clock</u> Impedance	50 Ω	50 Ω	100 Ω	Clock»Ready f
<u>Ref Clock</u> <u>Rate</u>	10M	10M	10M	Timing»Ref Clc
<u>Ref Clock</u> <u>Source</u>				Timing»Ref Clc
<u>Ref Trigger</u> <u>Pretrigger</u> Samples	0	0	0	Triggers»Ref»F Per Record

<u>Ref Trigger</u> <u>Type</u>	None	None	None	Triggers»Ref»1
<u>Repeat</u> <u>Count</u>	1	1	1	Dynamic Gene
<u>Repeat</u> <u>Mode</u>	Finite	Finite	Finite	Dynamic Gene
<u>Resource</u> <u>Descriptor</u>				Device Charact Descriptor
<u>Sample</u> <u>Clock</u> Impedance	50 Ω	50 Ω	100 Ω	Timing»Sample
<u>Sample</u> Clock Rate	50 MHz	50 MHz	50 MHz	Timing»Sample
<u>Sample</u> <u>Clock Source</u>	OnBoardClock	OnBoardClock	OnBoardClock	Timing»Sample
<u>Sample Error</u> Backlog	N/A		N/A	Advanced»Har Compare»Outr
<u>Sample Error</u> <u>Buffer</u> Overflowed	N/A		N/A	Advanced»Har Compare»Sam Overflowed
<u>Sample Error</u> Event Output <u>Terminal</u>	N/A		N/A	Events»Sample Terminal
<u>Samples</u> Compared	N/A		N/A	Advanced»Har Compare»Sam
<u>Samples Per</u> <u>Record</u>	1000	1000	1000	Dynamic Acqui Record
<u>Script To</u> Generate				Dynamic Gene Generate
<u>Script</u> Trigger Type	None	None	None	Triggers»Script
<u>Serial</u> Number				Device Charact Number
<u>Space</u>	0	0	0	Dynamic Gene

Available in Streaming Waveform				Transfer»Strea Available in Str
<u>Start Trigger</u> Type	None	None	None	Triggers»Start»
<u>Static</u> <u>Channels</u>				Static Channel
<u>Streaming</u> Enable	FALSE	FALSE	FALSE	Dynamic Gene Transfer»Strea
<u>Streaming</u> <u>Waveform</u> <u>Name</u>				Dynamic Gene Transfer»Strea Waveform Narr
<u>Total</u> Acquisition Size				Device Charact Acquisition Size
<u>Total</u> <u>Generation</u> <u>Size</u>				Device Charact Generation Siz
<u>Trigger</u> Voltage High <u>Level</u>	3.3 V logic family voltage levels. Refer to device specifications for more information.	3.3 V logic family voltage levels. Refer to device specifications for more information.	N/A	Voltage Levels:
<u>Trigger</u> <u>Voltage Low</u> <u>Level</u>	3.3 V logic family voltage levels. Refer to device specifications for more information	3.3 V logic family voltage levels. Refer to device specifications for more information	N/A	Voltage Levels:
<u>Waveform to</u> <u>Generate</u>		1111		Dynamic Gene Generate

# C/C++ Reference

This section describes the functions and attributes included with NI-HSDIO that you can use to configure and operate your digital waveform generator/analyzer.

# **NI-HSDIO Functions**

#### **Class/Panel Name**

Initialize Acquisition Session Initialize Generation Session Close

#### Voltage

Configure Data Voltage (Logic Family) Configure Data Voltage (Custom Levels) Configure Trigger Voltage (Logic Family) Configure Trigger Voltage (Custom Levels) Configure Event Voltage (Logic Family) Configure Event Voltage (Custom Levels) **Dynamic I/O** 

Assign Dynamic Channels Initiate Wait Until Done

Abort

#### **Dynamic Acquisition**

Configure Acquisition Size Configure Data Interpretation Read Waveform (1D U32) Fetch Waveform (1D U32) Read Waveform (1D U16) Fetch Waveform (1D U16) Read Waveform (1D U8) Fetch Waveform (1D U8) Read Multi Record (2D U32) Fetch Multi Record (2D U32) Func

niHSDIO_InitAcquisiti( niHSDIO_InitGenerati niHSDIO_close

niHSDIO_ConfigureD; niHSDIO_ConfigureT; niHSDIO_ConfigureTr niHSDIO_ConfigureE niHSDIO_ConfigureE

niHSDIO_AssignDyna niHSDIO_Initiate niHSDIO_WaitUntilDo niHSDIO_Abort

niHSDIO_ConfigureA( niHSDIO_ConfigureD; niHSDIO_ReadWavef niHSDIO_FetchWavef niHSDIO_ReadWavef niHSDIO_FetchWavef niHSDIO_ReadWavef niHSDIO_FetchWavef niHSDIO_FetchWavef niHSDIO_FetchWavef

Read Multi Record (2D U16) Fetch Multi Record (2D U16) Read Multi Record (2D U8) Fetch Multi Record (2D U8) **Dynamic Generation** Write Named Waveform (1D U32) Write Named Waveform (1D U16) Write Named Waveform (1D U8) Write Named Waveform (WDT) Write Named Waveform From File (HWS) Initial/Idle States Configure Idle State (String) Configure Idle State (U32) Configure Initial State (String) Configure Initial State (U32) Waveform Control **Configure Repeat Mode Configure Waveform To Generate** Allocate Named Waveform Set Named Waveform Next Write Position **Delete Named Waveform** Scripting **Configure Generation Mode** Write Script **Configure Script To Generate** Timing & Triggering Timing **Configure Sample Clock Configure Data Position** 

niHSDIO_ReadMultiR niHSDIO_FetchMultiR niHSDIO_ReadMultiR niHSDIO_FetchMultiR

niHSDIO_WriteName( niHSDIO_WriteName( niHSDIO_WriteName( niHSDIO_WriteName( niHSDIO_WriteName(

niHSDIO_ConfigureId niHSDIO_ConfigureId niHSDIO_ConfigureIn niHSDIO_ConfigureIn

niHSDIO_ConfigureG niHSDIO_ConfigureW niHSDIO_AllocateNar niHSDIO_SetNamedV niHSDIO_DeleteName

niHSDIO_ConfigureG niHSDIO_WriteScript niHSDIO_ConfigureS(

niHSDIO_ConfigureS; niHSDIO_ConfigureD; Configure Data Position Delay

#### Advanced

**Configure Ref Clock** Adjust Sample Clock Relative Delay

#### Triggers

#### Start Trigger

Configure Start Trigger (Digital Edge) Configure Start Trigger (Digital Pattern Match) Configure Start Trigger (Software) **Disable Start Trigger** 

#### **Ref Trigger**

Configure Ref Trigger (Digital Edge) Configure Ref Trigger (Digital Pattern Match) Configure Ref Trigger (Software) **Disable Ref Trigger** 

niHSDIO ConfigureDa

niHSDIO ConfigureRe niHSDIO AdjustSamp

niHSDIO ConfigureDi niHSDIO ConfigurePa niHSDIO ConfigureSc niHSDIO DisableStar

niHSDIO ConfigureDi niHSDIO ConfigurePa niHSDIO ConfigureSc niHSDIO DisableRef1

#### **Advance Trigger**

Configure Advance Trigger (Digital Edge) Configure Advance Trigger (Digital Pattern Match) niHSDIO ConfigurePa Configure Advance Trigger (Software) Disable Advance Trigger

#### **Script Trigger**

Configure Script Trigger (Digital Edge) Configure Script Trigger (Digital Level) Configure Script Trigger (Software) **Disable Script Trigger** 

#### **Pause Trigger**

Configure Pause Trigger (Digital Level) Configure Pause Trigger (Digital Pattern Match) **Disable Pause Trigger** 

niHSDIO ConfigureDi niHSDIO ConfigureSc niHSDIO DisableAdva

niHSDIO ConfigureDi niHSDIO ConfigureDi niHSDIO ConfigureSc niHSDIO DisableScrit

niHSDIO ConfigureDi niHSDIO ConfigurePa niHSDIO DisablePau

Send Software Trigger (Edge Trigger) **Events** Export Signal Static I/O **Assign Static Channels Read Static Channels** Write Static Channels Calibration Self Calibrate **Change External Calibration Password** Initialize External Calibration Session Adjust Channel Voltage **Close External Calibration Utility Functions** Self-Test Is Done? **Device Control Tristate Channels Commit Dynamic** Commit Static Reset **Reset Device Error Handling** Clear Error Error Message Get Error Locking Lock Session **Unlock Session** 

niHSDIO_SendSoftwa

niHSDIO_ExportSigna

niHSDIO_AssignStatic niHSDIO_ReadStaticL niHSDIO_WriteStaticL

niHSDIO_SelfCal niHSDIO_ChangeExt( niHSDIO_InitExtCal niHSDIO_CalAdjustCl niHSDIO_CloseExtCa

niHSDIO_self_test niHSDIO_IsDone

niHSDIO_TristateCha niHSDIO_CommitDyn niHSDIO_CommitStat niHSDIO_reset niHSDIO_ResetDevic

niHSDIO_ClearError niHSDIO_error_messa niHSDIO_GetError

niHSDIO_LockSessio niHSDIO_UnlockSess

#### Set/Get Attribute Set Attribute

Set Attribute ViBoolean Set Attribute ViInt32 Set Attribute ViReal64 Set Attribute ViSession Set Attribute ViString **Get Attribute** ViBoolean Get Attribute ViBoolean Get Attribute ViReal64 Get Attribute ViSession Get Attribute ViSession

niHSDIO_SetAttribute niHSDIO_SetAttribute niHSDIO_SetAttribute niHSDIO_SetAttribute niHSDIO_SetAttribute

niHSDIO_GetAttribute niHSDIO_GetAttribute niHSDIO_GetAttribute niHSDIO_GetAttribute niHSDIO_GetAttribute

# niHSDIO_InitAcquisitionSession

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_InitAcquisitionSession (ViRsrc resourceName, ViBoolean idQuery, ViBoolean resetInstrument, ViConstString optionString, ViSession* vi);

### Purpose

Call this function to create a new acquisition session. You can perform static and dynamic acquisition operations with this session.

Creating a new session does not automatically tristate your front panel terminals or channels that might have been left driving voltages from previous sessions (refer to <u>niHSDIO_close</u> for more information about leaving lines driving after closing a session).

Pass VI_TRUE into the **resetInstrument** parameter to place your device in a known start-up state when creating a new session. This action is equivalent to calling <u>niHSDIO_reset</u>, and it tristates the front panel terminals and channels.

### Parameters

Name	Туре	Description
resourceName	ViRsrc	Specifies the device name, for example "Dev1" where "Dev1" is an device name assigned by Measurement & Automation Explorer.
idQuery	ViBoolean	Specifies whether the driver performs an ID query on the device. When this parameter is set to VI_TRUE, NI- HSDIO ensures compatibility between the device and the driver.
		Defined Values
		VI_TRUE (1)—Perform ID query. VI_FALSE (0)—Skip ID query.
		Default Value: VI_FALSE
resetInstrument	ViBoolean	Specifies whether the driver resets the device during initialization of the session. Refer to <u>niHSDIO_reset</u> for more information about what happens during a device reset.
		Defined Values
		VI_FALSE (0)—Do not reset device. VI_TRUE (1)—Reset device.
		Default Value: VI_FALSE
		Note Resetting your device resets the <i>entire</i> device. Acquisition or generation operations in progress are aborted and cleared.

**optionString** ViConstString Currently unused. Set this string to "".

vi ViSession	Returns a ViSession handle. Use this handle to identify the device in all subsequent instrument driver function calls related to your acquisition operation.
--------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Return Value

# niHSDIO_InitGenerationSession

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_InitGenerationSession (ViRsrc resourceName, ViBoolean idQuery, ViBoolean resetInstrument, ViConstString optionString, ViSession* vi);
Call this function to create a new generation session. You can perform static and dynamic generation operations with this session.

Creating a new session does not automatically tristate your front panel terminals or channels that might have been left driving voltages from previous sessions (refer to <u>niHSDIO_close</u> for more information about leaving lines driving after closing a session).

Pass VI_TRUE into the **resetInstrument** parameter to place your device in a known start-up state when creating a new session. This action is equivalent to calling <u>niHSDIO_reset</u>, and it tristates the front panel terminals and channels.

Name	Туре	Description
resourceName	ViRsrc	Specifies the device name, for example "Dev1" where "Dev1" is a device name assigned by Measurement & Automation Explorer.
idQuery	ViBoolean	Specifies whether the driver performs an ID query upon the device. When this parameter is set to VI_TRUE, NI- HSDIO ensures compatibility between the device and driver.
		Defined Values
		VI_TRUE (1)—Perform ID query. VI_FALSE (0)—Skip ID query.
		Default Value: VI_FALSE
resetInstrument	ViBoolean	Specifies whether the driver resets the device during initialization of the session. Refer to <u>niHSDIO_reset</u> for more information about what happens during a device reset.
		Defined Values
		VI_FALSE (0)—Do not reset device. VI_TRUE (1)—Reset device.
		Default Value: VI_FALSE
		Note Resetting your device resets the <i>entire</i> device. Acquisition or generation operations in progress are aborted and cleared.

optionString	ViConstString	Currently unused. Set this string to "".
vi	ViSession	Returns a VISession handle. Use this handle to identify the device in all subsequent instrument driver function calls related to your generation operation.

Return Value

# niHSDIO_close

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_close (ViSession vi);

Closes the session and frees resources that it has reserved. If the session is running, it is first aborted.

To prevent generating unwanted signal glitches between sessions, no front panel terminals or channels are tristated by calling <u>niHSDIO_close</u>—they are all left driving whatever voltage they would have been driving had you simply called <u>niHSDIO_Abort</u>. Call <u>niHSDIO_reset</u> before calling <u>niHSDIO_close</u> if you want to tristate your terminals and channels before closing your session.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

Return Value

# niHSDIO_ConfigureDataVoltageLogicFamily

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureDataVoltageLogicFamily (ViSession vi, ViConstString channelList, ViInt32 logicFamily);

This function configures the voltage levels for the data channels using a logic family.



**Notes** Refer to <u>Logic Families</u> for links to the applicable logic families for your device and possible voltage restrictions.

NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this function when programming those devices.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the function used to initialize the session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
logicFamily	Vilnt32	Specifies the logic family for the data voltage levels.
		Defined Values
		NIHSDIO_VAL_1_8V_LOGIC (8)—Uses 1.8 V logic family.
		NIHSDIO_VAL_2_5V_LOGIC (7)—Uses 2.5
		NIHSDIO_VAL_3_3V_LOGIC (6)—Uses 3.3
		V logic family. NIHSDIO_VAL_5_0V_LOGIC (5)—Uses 5.0 V logic family.

Return Value

# niHSDIO_ConfigureDataVoltageCustomLevels

### **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureDataVoltageCustomLevels (ViSession vi, ViConstString channelList, ViReal64 lowLevel, ViReal64 highLevel);

This function configures the <u>voltage levels</u> of the data channels using the specified high and low levels.



**Notes** Refer to the device documentation for possible voltage restrictions.

If you are using an NI 654*x* device for generation sessions, you must set **highLevel** to the appropriate logic family value, and you must set **lowLevel** to 0. For acquisition sessions with the NI 654*x*, select the same value for **highLevel** and **lowLevel** from the following list: 0.9 V, 1.25 V, or 1.65 V.

NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this function when programming those devices.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the function used to initialize the session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
lowLevel	ViReal64	Specifies what voltage identifies low level.
highLevel	ViReal64	Specifies what voltage identifies high level.
<u>Return Value</u>		

# niHSDIO_ConfigureTriggerVoltageLogicFamily

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureTriggerVoltageLogicFamily (ViSession vi, ViInt32 logicFamily);

This function configures the voltage levels for the trigger channels using a logic family.



**Note** Refer to the device documentation for descriptions of logic families and possible voltage restrictions.

NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this function when programming those devices.

Name	Туре	Description	
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the function used to initialize the session.	
logicFamily	Vilnt32	Specifies the logic family for the trigger voltage levels.	
		Defined Values	
		NIHSDIO_VAL_1_8V_LOGIC (8)—Uses 1.8 V	
		NIHSDIO_VAL_2_5V_LOGIC (7)—Uses 2.5 V logic family.	
		NIHSDIO_VAL_3_3V_LOGIC (6)—Uses 3.3 V	
		NIHSDIO_VAL_5_0V_LOGIC (5)—Uses 5.0 V logic family.	

Return Value

# niHSDIO_ConfigureTriggerVoltageCustomLevels

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_ConfigureTriggerVoltageCustomLevels (ViSession vi, ViReal64 lowLevel, ViReal64 highLevel);

This function configures the voltage levels of the trigger channels using user-defined high and low levels.



**Notes** Refer to the device documentation for possible voltage restrictions.

If you are using an NI 654*x* device for generation sessions, you must set **highLevel** to the appropriate logic family value, and you must set **lowLevel** to 0. For acquisition sessions with the NI 654*x*, select the same value for **highLevel** and **lowLevel** from the following list: 0.9 V, 1.25 V, or 1.65 V.

NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this function when programming those devices.

NameTypeDescriptionviViSessionThis handle identifies your instrument session. vi<br/>was obtained from the function used to initialize<br/>the session.IowLevelViReal64Specifies what voltage identifies low level.highLevelViReal64Specifies what voltage identifies high level.Return Value

# niHSDIO_ConfigureEventVoltageLogicFamily

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureEventVoltageLogicFamily (ViSession vi, ViInt32 logicFamily);

This function configures the voltage levels for the event channels using a logic family.



**Notes** Refer to the device documentation for descriptions of <u>logic</u> <u>families</u> and possible voltage restrictions.

NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this function when programming those devices.

Name	Туре	Description	
vi	ViSession	This handle identifies your instrument session. $\mathbf{vi}$ was obtained from the function used to initialize the session.	
logicFamily	Vilnt32	Specifies the logic family for the event voltage levels.	
		Defined Values	
		NIHSDIO_VAL_1_8V_LOGIC (8)—Uses 1.8 V	
		NIHSDIO_VAL_2_5V_LOGIC (7)—Uses 2.5 V logic family.	
		NIHSDIO_VAL_3_3V_LOGIC (6)—Uses 3.3 V logic family.	
		NIHSDIO_VAL_5_0V_LOGIC (5)—Uses 5.0 V logic family.	

Return Value

# niHSDIO_ConfigureEventVoltageCustomLevels

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_ConfigureEventVoltageCustomLevels (ViSession vi, ViReal64 lowLevel, ViReal64 highLevel);

This function configures the voltage levels of the event channels using user-defined high and low levels.



**Notes** Refer to the device documentation for possible voltage restrictions.

If you are using an NI 654*x* device for generation sessions, you must set **highLevel** to the appropriate logic family value, and you must set **lowLevel** to 0. For acquisition sessions with the NI 654*x*, select the same value for **highLevel** and **lowLevel** from the following list: 0.9 V, 1.25 V, or 1.65 V.

NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this function when programming those devices.

NameTypeDescriptionviViSessionThis handle identifies your instrument session. vi<br/>was obtained from the function used to initialize<br/>the session.IowLevelViReal64Specifies what voltage identifies low level.highLevelViReal64Specifies what voltage identifies high level.Return Value

# niHSDIO_AssignDynamicChannels

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_AssignDynamicChannels (ViSession vi, ViConstString channelList);

Configures channels for dynamic acquisition (if **vi** is an acquisition session) or dynamic generation (if **vi** is a generation session).



**Note** A channel cannot be assigned to static generation and dynamic generation at the same time.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	This string identifies which channels are reserved for dynamic operation.
		Valid Syntax
		"0-19" or "0-15,16-19" or "0-18,19" "" (empty string) or VI_NULL to specify all channels "none" to unassign all channels
		Channels cannot be configured for both static generation and dynamic generation.
<u>Return Value</u>		

# niHSDIO_Initiate

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_Initiate (ViSession vi);

Commits any pending attributes to hardware and starts the dynamic operation (refer to the <u>niHSDIO_CommitDynamic</u> function for more information about committing).

For a generation operation with a Start trigger configured, calling <u>niHSDIO_Initiate</u> causes the channels to go to their <u>Initial states</u>.

This function is only valid for dynamic operations (acquisition or generation). It is not valid for static operations.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

Return Value

# niHSDIO_WaitUntilDone

# **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_WaitUntilDone (ViSession vi, ViInt32 maxTimeMilliseconds);

Call this function to pause execution of your program until the dynamic data operation is completed or the function returns a timeout error. <u>niHSDIO_WaitUntilDone</u> is a blocking function that periodically checks the operation status. It returns control to the calling program if the operation completes successfully or an error occurs (including a timeout error).

This function is most useful for finite data operations that you expect to complete within a certain time.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
maxTimeMilliseconds	Vilnt32	This parameter specifies the number of milliseconds to allow the function to complete before returning. If the specified time elapses before the data operation has completed, the function returns a timeout error.
		Setting a value of 0 causes the function to return immediately. This setting can be useful to manually poll for hardware errors after a data operation has been initiated. If no other error has occurred and the data operation is still not complete, the function returns a timeout error.
		Setting a value of -1 causes the function to never timeout. Be careful not to use this value during a continuous operation, as it will never return unless a hardware error occurs. Perform a manual device reset from Measurement & Automation Explorer if you get stuck in this state or use <u>niHSDIO_reset</u> or <u>niHSDIO_ResetDevice</u> from the other session of the device.

Default Value: 10000

Return Value

# niHSDIO_Abort

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_Abort (ViSession vi);

Stops a running dynamic session. This function is generally not required on finite data operations, as they complete on their own after the last data point is generated or acquired. This function is generally required for continuous operations or if you wish to interrupt a finite operation before it has completed.

This function is valid for dynamic operations (acquisition or generation) only. It is not valid for static operations.

Note To avoid receiving hardware clocking errors when reconfiguring an external clock, explicitly call the niHSDIO_Abort function after your finite operation has completed before performing any clocking reconfiguration. An external clock that stops sending pulses to the device (even after a finite operation has completed) may cause NI-HSDIO to return an error, stating that the clock became unlocked, if the device has not implicitly aborted yet.
Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_ConfigureAcquisitionSize

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureAcquisitionSize (ViSession vi, ViInt32 samplesPerRecord, ViInt32 numberOfRecords);

Configures the acquisition size, including the number of acquired records and the minimum record size.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
samplesPerRecord	Vilnt32	Sets the number of samples to be acquired per record. If you need pretrigger and post-rigger points, configure a <u>Reference trigger</u> and specify the number of pretrigger points.
		Default Value: 1000
numberOfRecords	Vilnt32	Sets how many records are acquired.
		Default Value: 1
<u>Return Value</u>		

# niHSDIO_ConfigureDataInterpretation

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureDataInterpretation (ViSession vi, ViConstString channelList, ViInt32 dataInterpretation);

Selects between high/low data or valid/invalid <u>data interpretation</u> during a static or dynamic acquisition operation.

Select NIHSDIO_VAL_HIGH_OR_LOW to get logic high or logic low values. Select NIHSDIO_VAL_VALID_OR_INVALID to determine if the signal is within the specified voltage range (above <u>NIHSDIO_ATTR_DATA_VOLTAGE_LOW_LEVEL</u> but below <u>NIHSDIO_ATTR_DATA_VOLTAGE_HIGH_LEVEL</u>) or outside the range (below <u>NIHSDIO_ATTR_DATA_VOLTAGE_HIGH_LEVEL</u>) or above <u>NIHSDIO_ATTR_DATA_VOLTAGE_HIGH_LEVEL</u>).



**Note** NI 654*x*/656*x* devices only support the high/low mode of data interpretation. NI-HSDIO returns an error if you select valid/invalid mode for an acquisition with these devices.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
dataInterpretation	Vilnt32	Selects the data interpretation mode.
		Defined Values
		NIHSDIO_VAL_HIGH_OR_LOW (3) —Data read represents logical values (logic high or low level)
		NIHSDIO_VAL_VALID_OR_INVALID (4)—Data read represents whether channel data is within the specified voltage range.
		<b>Default Value:</b> NIHSDIO_VAL_HIGH_OR_LOW
<u>Return Value</u>		

# niHSDIO_ReadWaveformU32

### **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ReadWaveformU32 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32* numberOfSamplesRead, ViUInt32[] data);

Initiates a waveform acquisition on channels enabled for dynamic acquisition, waits until it acquires the number of samples in **samplesToRead**, and returns the acquired binary data.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. vi was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and a timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>maxTimeMilliseconds</b> . A value of -1 causes the function to never time out.
numberOfSamplesRead	Vilnt32	Returns the number of samples that were successfully fetched and transferred into <b>data</b> [].
data	ViUInt32[]	Returns the preallocated array where samples are written.

# niHSDIO_FetchWaveformU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchWaveformU32 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32* numberOfSamplesRead, ViUInt32[] data);

Transfers acquired binary data from <u>onboard memory</u> to PC memory. The data was acquired to onboard memory previously by calling the <u>niHSDIO_Initiate</u> function.

If the number of samples specified in **samplesToRead** is still not available after the number of milliseconds specified in **maxTimeMilliseconds**, this function returns no data with a timeout error.

The fetch position can be modified by using <u>niHSDIO_SetAttributeViInt32</u> and the <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> attribute or <u>NIHSDIO_ATTR_FETCH_OFFSET</u> attributes. The default value for <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> is <u>NIHSDIO_VAL_CURRENT_READ_POSITION_The default value for</u>

NIHSDIO_VAL_CURRENT_READ_POSITION. The default value for <u>NISHDIO_ATTR_FETCH_OFFSET</u> is 0.

Calling this function is not necessary if you are using the <u>niHSDIO_ReadWaveformU32</u> function, as the fetch is performed as part of that function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from niHSDIO InitAcquisitionSession.
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
		Default Value: 1000
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>max</b> <b>time milliseconds</b> . A value of –1 causes the function to never timeout.
	) (1+0.0	Default Value: 10000
numberOfSamplesRead	Viint32	Returns the number of samples that were successfully fetched and transferred into <b>data</b> [].

data

# niHSDIO_ReadWaveformU16

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ReadWaveformU16 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32* numberOfSamplesRead, ViUInt16[] data);

Initiates a waveform acquisition on channels enabled for dynamic acquisition, waits until it acquires the number of samples in **samplesToRead**, and returns the acquired binary data.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>maxTimeMilliseconds</b> . A value of -1 causes the function to never time out.
numberOfSamplesRead	Vilnt32	Returns the number of samples that were successfully fetched and transferred into <b>data</b> [].
data	ViUInt16[]	Returns the preallocated array where samples are written.
Deturne Malure		

# niHSDIO_FetchWaveformU16

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchWaveformU16 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32* numberOfSamplesRead, ViUInt32[] data);

Transfers acquired binary data from onboard memory to PC memory. The data was acquired to onboard memory previously by calling the <u>niHSDIO_Initiate</u> function.

If the number of samples specified in **samplesToRead** is still not available after the number of milliseconds specified in **maxTimeMilliseconds**, this function returns no data with a timeout error.

The fetch position can be modified by using <u>niHSDIO_SetAttributeViInt32</u> and the <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> or <u>NIHSDIO_ATTR_FETCH_OFFSET</u> attributes. The default value for <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> is NIHSDIO_VAL_CURRENT_READ_POSITION. The default value for

<u>NIHSDIO_ATTR_FETCH_OFFSET</u> is 0.

Calling this function is not necessary if you use the <u>niHSDIO_ReadWaveformU16</u> function, as the fetch is performed as part of that function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and a timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>maxTimeMilliseconds</b> . A value of -1 causes the function to never time out.
numberOfSamplesRead	Vilnt32	Returns the number of samples that were successfully fetched and transferred into <b>data</b> [].
data	ViUInt32[]	Returns the preallocated array where samples are written.

# niHSDIO_ReadWaveformU8

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ReadWaveformU8 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32* numberOfSamplesRead, ViUInt8[] data);

Initiates a waveform acquisition on channels enabled for dynamic acquisition, waits until it acquires the number of samples in **samplesToRead**, and returns the acquired binary data.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and a timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>maxTimeMilliseconds</b> . A value of -1 causes the function to never time out.
numberOfSamplesRead	Vilnt32	Returns the number of samples that were successfully fetched and transferred into <b>data</b> [].
data	ViUInt8[]	Returns the preallocated array where samples are written.

# niHSDIO_FetchWaveformU8

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchWaveformU8 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32* numberOfSamplesRead, ViUInt8[] data);

Transfers acquired binary data from onboard memory to PC memory. The data was acquired to onboard memory previously by calling the <u>niHSDIO_Initiate</u> function.

If the number of samples specified in **samplesToRead** is still not available after the number of milliseconds specified in **maxTimeMilliseconds**, this function returns no data with a timeout error.

The fetch position can be modified by using <u>niHSDIO_SetAttributeViInt32</u> and the <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> or <u>NIHSDIO_ATTR_FETCH_OFFSET</u> attributes. The default value for <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> is NIHSDIO_VAL_CURRENT_READ_POSITION. The default value for <u>NIHSDIO_ATTR_FETCH_OFFSET</u> is 0.

Calling this function is not necessary if you use the <u>niHSDIO_ReadWaveformU8</u> function, as the fetch is performed as part of that function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>maxTimeMilliseconds</b> . A value of -1 causes the function to never time out.
numberOfSamplesRead	Vilnt32	Returns the number of samples that were successfully fetched and transferred into <b>data</b> [].
data	ViUInt8[]	Returns the preallocated array where samples are written.

# niHSDIO_FetchWaveformDirectDMA

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchWaveformDirectDMA (ViSession vi, ViInt32 maxTimeMilliseconds, ViInt32 samplesToRead, ViUint32 bufferSize, void bufferAddress, niHSDIO_wfmInfo waveformInfo, ViUInt32* offsetToFirstSample;

Transfers acquired waveform data from device memory directly to PC memory allocated by a Direct DMA-compatible device. The size of the sample that is transferred is determined by the the data width for your device.

If the number of samples specified in **samplesToRead** is still not available after the number of milliseconds specified in **maxTimeMilliseconds**, this function returns no data with a timeout error.

The fetch position can be modified by using <u>niHSDIO_SetAttributeViInt32</u> and the <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> or <u>NIHSDIO_ATTR_FETCH_OFFSET</u> attributes. The default value for <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> is NIHSDIO_VAL_CURRENT_READ_POSITION. The default value for <u>NIHSDIO_ATTR_FETCH_OFFSET</u> is 0.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSessio</u>
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and timeou occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> tha is greater than the number of samples in the device memor NI-HSDIO returns the available samples after <b>maxTimeMilliseconds</b> . A value of -1 causes the function to never time out.
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
bufferSize	ViUInt32	Specifies the size (in bytes) of the buffer in memory at which to transfer acquired data.
bufferAddress	void	Specifies the location for the buffer in memory at which to

		transfer acquired data.
waveformInfo	niHSDIO_wfmInfo	Returns the absolute and relative timestamp for the operation, the dt, and the actual number of samples read.
offsetToFirstSample	ViUInt32*	Returns the offset of the first sample acquired within the specified buffer. Data is transfered from device memory in 128 byte increments, so the first sample of the acquired data typically occurs at some offset from the start of the buffer when using Reference trigger.

# niHSDIO_ReadMultiRecordU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ReadMultiRecordU32 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32 startingRecord, ViInt32 recordsToRead, ViUInt32[] waveformData, niHSDIO_wfmInfo * waveformInfo);

Initiates a multirecord acquisition and returns the acquired waveform as a two-dimensional array of unsigned 32-bit data.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSessio</u>
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of -1 causes the function to never time out.
startingRecord	Vilnt32	Specifies the first record you want to fetch.
recordsToRead	Vilnt32	Specifies the number of records you want to fetch.
waveformData	ViUInt32[]	Returns the array of waveforn data that contains the records to fetch.
waveformInfo	niHSDIO_wfmInfo *	Returns information about the records. <b>waveformInfo</b> includes an absolute timestamp, relative timestamp the number of samples fetched, and the dT of the waveform.

# niHSDIO_FetchMultiRecordU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchMultiRecordU32 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32 startingRecord, ViInt32 recordsToRead, ViUInt32[] waveformData, niHSDIO_wfmInfo * waveformInfo);

Fetches the data as a two-dimensional array of unsigned 32-bit integers and returns the number of samples read.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSessio</u>
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of -1 causes the function to never time out.
startingRecord	Vilnt32	Specifies the the first record you want to fetch.
recordsToRead	Vilnt32	Specifies the number of records you want to fetch.
waveformData	ViUInt32[]	Returns the array of waveforn data that contains the records to fetch.
waveformInfo	niHSDIO_wfmInfo *	Returns information about the records. <b>waveformInfo</b> includes an absolute timestamp, relative timestamp the number of samples fetched, and the dT of the waveform.
# niHSDIO_ReadMultiRecordU16

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ReadMultiRecordU16 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32 startingRecord, ViInt32 recordsToRead, ViUInt16[] waveformData, niHSDIO_wfmInfo * waveformInfo);

This function initiates a multirecord acquisition, and returns the acquired waveform as a two-dimensional array of unsigned 16-bit data.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u>
samplesToRead	Vilnt32	Number of samples to fetch.
maxTimeMilliseconds	Vilnt32	This attribute specifies in milliseconds how long to allow the function to complete befor returning a timeout error.
		A value of -1 causes the function to never time out.
startingRecord	Vilnt32	The first record you want to fetch.
recordsToRead	Vilnt32	The number of records you want to fetch.
waveformData	ViUInt16[]	The array of waveform data that contains the records to fetch.
waveformInfo	niHSDIO_wfmInfo *	Returns information about the records. <b>waveformInfo</b> includes an absolute timestamp, relative timestamp the number of samples fetched, and the dT of the waveform.

# niHSDIO_ReadMultiRecordU8

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ReadMultiRecordU8 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32 startingRecord, ViInt32 recordsToRead, ViUInt8[] waveformData, niHSDIO_wfmInfo * waveformInfo);

Initiates a multirecord acquisition and returns the acquired waveform as a two-dimensional array of unsigned 8-bit data.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSessio</u>
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of -1 causes the function to never time out.
startingRecord	Vilnt32	Specifies the first record you want to fetch.
recordsToRead	Vilnt32	Specifies the number of records you want to fetch.
waveformData	ViUInt8[]	Returns the array of waveforn data that contains the records to fetch.
waveformInfo	niHSDIO_wfmInfo *	Returns information about the records. <b>waveformInfo</b> includes an absolute timestamp, relative timestamp the number of samples fetched, and the dT of the waveform.

# niHSDIO_FetchMultiRecordU16

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchMultiRecordU16 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32 startingRecord, ViInt32 recordsToRead, ViUInt16[] waveformData, niHSDIO_wfmInfo * waveformInfo);

Fetches the data as a two-dimensional array of unsigned 16-bit integers and returns the number of samples read.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u>
samplesToRead	Vilnt32	Specifies the Number of samples to fetch.
		Default Value: 1000
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of -1 causes the function to never timeout.
		Default Value: 10000
startingRecord	Vilnt32	Specifies the first record you want to fetch.
recordsToRead	Vilnt32	Specifies the number of records you want to fetch.
waveformData	ViUInt16[]	Returns the array of waveforn data that contains the records to fetch.
waveformInfo	niHSDIO_wfmInfo *	Returns information about the records. <b>waveformInfo</b> includes an absolute timestamp, relative timestamp the number of samples fetched, and the dT of the waveform.

# niHSDIO_FetchMultiRecordU8

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_FetchMultiRecordU8 (ViSession vi, ViInt32 samplesToRead, ViInt32 maxTimeMilliseconds, ViInt32 startingRecord, ViInt32 recordsToRead, ViUInt8[] waveformData, niHSDIO_wfmInfo * waveformInfo);

Fetches the data as a two-dimensional array of unsigned 8-bit integers and returns the number of samples read.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u>
samplesToRead	Vilnt32	Specifies the number of samples to fetch.
		Default Value: 1000
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of -1 causes the function to never timeout.
		Default Value: 10000
startingRecord	Vilnt32	Specifies the first record you want to fetch.
recordsToRead	Vilnt32	Specifies the number of records you want to fetch.
waveformData	ViUInt8[]	Returns the array of waveforn data that contains the records to fetch.
waveformInfo	niHSDIO_wfmInfo *	Returns information about the records. <b>waveformInfo</b> includes an absolute timestamp, relative timestamp the number of samples fetched, and the dT of the waveform.

# niHSDIO_HWC_FetchSampleErrors

### **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_HWC_FetchSampleErrors (ViSession vi, ViInt32* numSampleErrorsToRead, ViInt32 maxTimeMilliseconds, ViReal64[] sampleNumber, ViUInt32[] errorBits, ViInt32[] errorRepeatCount, ViUInt32* reserved1, ViUInt32* reserved2);

Returns the sample error information from a hardware comparison operation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
numSampleErrorsToRead	Vilnt32*	Specifies the number of sample errors to fetch.
maxTimeMilliseconds	Vilnt32	Specifies in milliseconds how long to allow the function to complete before returning a timeout error.
		A value of 0 causes the function to return immediately with up to the number of samples specified in <b>samplesToRead</b> . If you set <b>maxTimeMilliseconds</b> to a value other than 0, and timeout occurs before all the samples are acquired, you receive a timeout error. If you specify a value for <b>samplesToRead</b> that is greater than the number of samples in the device memory, NI-HSDIO returns the available samples after <b>max time milliseconds</b> . A value of –1 causes the function to never timeout.
sampleNumber	ViReal64[]	Returns the number of samples with errors.
errorBits	ViUInt32[]	Returns the bit numbers that were in error for each sample that has an error.

errorRepeatCount	Vilnt32[]	Returns the number of times that error was repeated.
reserved1	ViUInt32*	Reserved.
reserved2	ViUInt32*	Reserved.
<u>Return Value</u>		

# niHSDIO_WriteNamedWaveformU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_WriteNamedWaveformU32 (ViSession vi, ViConstString waveformName, ViInt32 samplesToWrite, ViUInt32[] data);

Transfers waveform data from PC memory to <u>onboard memory</u>. Supported devices for this function depend on the <u>data width</u> for your device, not on the number of assigned dynamic channels. This function may be used when the data width is 4.

If you specify a **waveformName** not already allocated on the device, the appropriate amount of onboard memory is allocated (if available), and the data is stored in that new location.

Data is always written to memory starting at the current write position of the waveform. A new waveform write position is the start of the allocated memory. Calling this function moves the next write position to the end of the data just written, so subsequent calls to this function append data to the end of previously written data. You can manually change the write position by calling <u>niHSDIO_SetNamedWaveformNextWritePosition</u>. If you try to write past the end of the allocated space, NI-HSDIO returns an error.

Waveforms are stored contiguously in onboard memory. You cannot resize an existing named waveform. Instead, delete the existing waveform using <u>niHSDIO_DeleteNamedWaveform</u> and then recreate it with the new size using the same name.

This function calls <u>niHSDIO_CommitDynamic</u>— all pending attributes are committed to hardware.

When you explicitly call <u>niHSDIO_AllocateNamedWaveform</u> and write waveforms using multiple niHSDIO_WriteNamedWaveformU32 calls, each waveform block written must be a multiple of 32 samples for the NI 654*x*/655*x* devices or a multiple of 64 samples for the NI 656*x* devices (128 samples if the NI 656*x* is in DDR mode).

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies a string representing the name to associate with the allocated waveform memory.
		Default Value: "" (empty string)
samplesToWrite	Vilnt32	Specifies the number of samples in <b>data</b> to be written to onboard memory.
		If <b>samples to write</b> is less than the size of <b>data</b> , only the number of samples indicated in <b>samplesToWrite</b> are written.
		Default Value: 1000
data	ViUInt32[]	Specifies the waveform data.
		If you want to use Direct DMA to write your waveform from onboard memory, pass the memory address (pointer value) of the region so that you write within the direct DMA window.
<u>Return Value</u>		

# niHSDIO_WriteNamedWaveformU16

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_WriteNamedWaveformU16 (ViSession vi, ViConstString waveformName, ViInt32 samplesToWrite, ViUInt16[] data);

Transfers waveform data from PC memory to onboard memory. Supported devices for this function depend on the <u>data width</u> for your device, not on the number of assigned dynamic channels. This function may be used when the data width is 2.

If you specify a **waveformName** not already allocated on the device, the appropriate amount of onboard memory is allocated (if available) and the data is stored in that new location.

Data is always written to memory starting at the current write position of the waveform. A new waveform's write position is the start of the allocated memory. Calling this function moves the next write position to the end of the data just written. Thus, subsequent calls to this function append data to the end of previously written data. You may also manually change the write position by calling

<u>niHSDIO_SetNamedWaveformNextWritePosition</u>. If you try to write past the end of the allocated space, an error will be returned.

Waveforms are stored contiguously in onboard memory. You cannot resize an existing named waveform. Instead, delete the existing waveform using <u>niHSDIO_DeleteNamedWaveform</u> and then re-create it with the new size using the same name.

This function calls <u>niHSDIO_CommitDynamic</u>—all pending attributes are committed to hardware.

When you explicitly call <u>niHSDIO_AllocateNamedWaveform</u> and write waveforms using multiple niHSDIO_WriteNamedWaveformU16 calls, each waveform block written must be a multiple of 32 samples for the NI 654*x*/655*x* devices or a multiple of 64 samples for the NI 656*x* devices (128 samples if the NI 656*x* is in DDR mode).

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. vi was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies a string representing the name to associate with the allocated waveform memory.
samplesToWrite	Vilnt32	Specifies the number of samples in <b>data</b> to be written to onboard memory.
		If <b>samples to write</b> is less than the size of <b>data</b> , only the number of samples indicated in <b>samplesToWrite</b> are written.
data	ViUInt16[]	Specifies the waveform data.
		If you want to use Direct DMA to write your waveform from onboard memory, pass the memory address (pointer value) of the region so that you write within the direct DMA window.
<u>Return Value</u>		

# niHSDIO_WriteNamedWaveformU8

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_WriteNamedWaveformU8 (ViSession vi, ViConstString waveformName, ViInt32 samplesToWrite, ViUInt8[] data);

Transfers waveform data from PC memory to onboard memory. Supported devices for this function depend on the <u>data width</u> for your device, not on the number of assigned dynamic channels. This function may be used when the data width is 1.

If you specify a **waveformName** not already allocated on the device, the appropriate amount of onboard memory is allocated (if available) and the data is stored in that new location.

Data is always written to memory starting at the current write position of the waveform. A new waveform's write position is the start of the allocated memory. Calling this function moves the next write position to the end of the data just written. Thus, subsequent calls to this function append data to the end of previously written data. You may also manually change the write position by calling

<u>niHSDIO_SetNamedWaveformNextWritePosition</u>. If you try to write past the end of the allocated space, an error will be returned.

Waveforms are stored contiguously in onboard memory. You cannot resize an existing named waveform. Instead, delete the existing waveform using <u>niHSDIO_DeleteNamedWaveform</u> and then recreate it with the new size using the same name.

This function calls <u>niHSDIO_CommitDynamic</u> - all pending attributes are committed to hardware.

When you explicitly call <u>niHSDIO_AllocateNamedWaveform</u> and write waveforms using multiple niHSDIO_WriteNamedWaveformU8 calls, each waveform block written must be a multiple of 32 samples for the NI 654*x*/655*x* devices or a multiple of 64 samples for the NI 656*x* devices (128 samples if the NI 656*x* is in DDR mode).

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. vi was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies a string representing the name to associate with the allocated waveform memory.
samplesToWrite	Vilnt32	Specifies the number of samples in <b>data</b> to be written to onboard memory.
		If <b>samples to write</b> is less than the size of <b>data</b> , only the number of samples indicated in <b>samplesToWrite</b> are written.
data	ViUInt8[]	Specifies the waveform data.
		If you want to use Direct DMA to write your waveform from onboard memory, pass the memory address (pointer value) of the region so that you write within the direct DMA window.
<u>Return Value</u>		

# niHSDIO_WriteNamedWaveformWDT

### **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_WriteNamedWaveformWDT (ViSession vi, ViConstString waveformName, ViInt32 samplesToWrite, ViInt32 dataLayout, ViUInt8[] data);

Transfers multistate digital waveforms from PC memory to <u>onboard</u> <u>memory</u>. Each element of **data**[] uses one byte per channel per sample. The supported values are defined in niHSDIO.h.

If you specify a **waveformName** not already <u>allocated</u> on the device, the appropriate amount of onboard memory is allocated (if available), and the data is stored in that new location.

Data is always written to memory starting at the current write position of the waveform. A new waveform's write position is the start of the allocated memory. Calling this function moves the next write position to the end of the data just written. Thus, subsequent calls to this function append data to the end of previously written data. You can manually change the write position by calling

<u>niHSDIO_SetNamedWaveformNextWritePosition</u>. If you try to write past the end of the allocated space, NI-HSDIO returns an error.

Waveforms are stored contiguously in onboard memory. You cannot resize an existing named waveform. Instead, delete the existing waveform using <u>niHSDIO_DeleteNamedWaveform</u> and then recreate it with the new size using the same name.

This function calls <u>niHSDIO_CommitDynamic</u>— all pending attributes are committed to hardware.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies a string representing the name to associate with the allocated waveform memory.
samplesToWrite	Vilnt32	Specifies the number of samples in data to be written to onboard memory. This number is not equal to the length of the <b>data</b> [] array, since its size is the number of samples to write times the number of channels.
dataLayout	Vilnt32	Describes the layout of the waveform contained in <b>data</b> [].
		Defined Values
		NIHSDIO_VAL_GROUP_BY_SAMPLE— Specifies that consecutive samples in <b>data</b> [] are such that the array contains th first sample from every signal in the operation, then the second sample from every signal, up to the last sample from every signal. NIHSDIO_VAL_GROUP_BY_CHANNEL- Specifies that consecutive samples in <b>data</b> [] are such that the array contains a the samples from the first signal in the operation, then all the samples from the second signal, up to all samples from the last signal.
data	ViUInt8[]	Specifies the digital waveform data. Eac value on this array defines the state of one channel of one sample. Supported

states are defined in niHSDIO.h

# niHSDIO_WriteNamedWaveformFromFileHWS

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_WriteNamedWaveformFromFileHWS (ViSession vi, ViConstString waveformName, ViConstString filePath, ViBoolean useRateFromWaveform, ViInt32* waveformSize);

Reads and transfers data from a digital .hws file to <u>onboard memory</u>.

If you specify a **waveformName** not already <u>allocated</u> on the device, the appropriate amount of onboard memory is allocated (if available), and the data is stored in that new location.

Data is always written to memory starting at the current write position of the waveform. A new waveform's write position is the start of the allocated memory. Calling this function moves the next write position to the end of the data just written. Thus, subsequent calls to this function append data to the end of previously written data. You can manually change the write position by calling

<u>niHSDIO_SetNamedWaveformNextWritePosition</u>. If you try to write past the end of the allocated space, NI-HSDIO returns an error.

Waveforms are stored contiguously in onboard memory. You cannot resize an existing named waveform. Instead, delete the existing waveform using <u>niHSDIO_DeleteNamedWaveform</u> and then re-create it with the new size using the same name.

This function calls <u>niHSDIO_CommitDynamic</u>— all pending attributes are committed to hardware.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies a string representing the name to associate with the allocated waveform memory.
filePath	ViConstString	Specifies the path and file name of the digital .hws file to open. The .hws extension is typically used for .hws files, although using this extension is optional.
useRateFromWaveform	ViBoolean	Controls how the sample rate is computed.
		Setting this value to TRUE computes the generation rate from the WDT value. If the sample rate has been configured using <u>niHSDIO_ConfigureSampleCloc</u> function, <b>useRateFromWaveform</b> overrides the sample rate.
waveformSize	Vilnt32	Returns the number of samples contained in the waveform.

# niHSDIO_ConfigureIdleState

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureIdleState (ViSession vi, ViConstString channelList, ViConstString idleState);

Sets the <u>Idle state</u> for a dynamic generation operation. The Idle state may be active in a variety of conditions:

- The generation operation completes normally.
- The generation operation pauses from an active Pause trigger.
- The generation operation terminates due to an underflow error.

#### Valid Syntax:

Both of these examples are valid and do the same thing. The order of **channelList** determines the order of the pattern string.

niHSDIO_ConfigureIdleState(vi, "19-0", "0000 0XXX XX11 111Z ZZZZ");

niHSDIO_ConfigureIdleState(vi, "0-19", "ZZZZ Z111 11XX XXX0 0000");

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
channelList	ViConstString	Specifies which channels will have their idle value set using the <b>idleState</b> string. The order of channels in <b>channelList</b> determines the order of the <b>idleState</b> string.
idleState	ViConstString	<ul> <li>Describes the Idle state of a dynamic generation operation. This expression is composed of the following characters:</li> <li>'X' or 'x': keeps the previous value</li> <li>'1': sets the channel to logic high</li> <li>'0': sets the channel to logic low</li> <li>'Z' or 'z': disables the channel (sets it to high-impedance)</li> </ul>
		<ul> <li>Note NI 656x devices do not support the high-impedance (Z) Idle state.</li> <li>The first character in the expression corresponds to the first channel in channelList. The number of characters in pattern must equal the number of channels specified in channelList, or NI-HSDIO returns an error.</li> <li>The default state of a channel is to keep the previous value.</li> </ul>

# niHSDIO_ConfigureIdleStateU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureIdleStateU32 (ViSession vi, ViUInt32 idleState);
Sets the <u>Idle state</u> for a dynamic generation operation. The Idle state may be active in a variety of conditions:

- The generation operation completes normally.
- The generation operation pauses from an active Pause trigger.
- The generation operation terminates due to an underflow error.

Unlike <u>niHSDIO_ConfigureIdleState</u> which uses a string, this function uses a binary format to only represent high and low. If you require more choices for your Idle state, use the <u>niHSDIO_ConfigureIdleState</u> function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
idleState	Vilnt32	Specifies the bit mask representing the Idle state. High is specified with a 1, and low is specified with a 0.
		The first character in <b>idleState</b> corresponds to the first channel in <b>channelList</b> . The number of characters in the pattern must equal the number of channels specified in <b>channelList</b> , or NI-HSDIO returns an error.

# niHSDIO_ConfigureInitialState

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureInitialState (ViSession vi, ViConstString channelList, ViConstString initialState);

Sets the <u>Initial state</u> for a dynamic generation operation. The Initial state of each channel is driven after the session is initiated using <u>niHSDIO_Initiate</u>. Channels remain unchanged until the first waveform sample is generated.

#### Valid Syntax:

Both of these examples are valid and do the same thing. The order of **channelList** determines the order of the pattern string.

- <u>niHSDIO_ConfigureInitialState</u> (vi, "19-0", "0000 0XXX XX11 111Z ZZZZ");
- <u>niHSDIO_ConfigureInitialState</u> (vi, "0-19", "ZZZZ Z111 11XX XXX0 0000");

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
channelList	ViConstString	Specifies which channels have their initial value set using the <b>initialState</b> string. The order of channels in <b>channelList</b> determines the order of the <b>initialState</b> string.
initialState	ViConstString	<ul> <li>Describes the Initial state of a dynamic generation operation. This expression is composed of characters: <ul> <li>'X' or 'x': keeps the previous value</li> <li>'1': sets the channel to logic high</li> <li>'0': sets the channel to logic low</li> <li>'Z' or 'z': disables the channel or sets it to high-impedance</li> </ul> </li> <li>Note NI 656x devices do not support the high-impedance (Z) Initial state.</li> <li>The first character in the expression corresponds to the first characters in pattern must equal the number of characters in pattern must equal the number of channels specified in channelList or an error is returned.</li> </ul>
		<ul> <li>generation operation. This expression is composed of characters: <ul> <li>'X' or 'x': keeps the previous value</li> <li>'1': sets the channel to logic high</li> <li>'0': sets the channel to logic low</li> <li>'Z' or 'z': disables the channel or set it to high-impedance</li> </ul> </li> <li>Note NI 656x devices do not support the high-impedance (Z) Initial state. The first character in the expression corresponds to the first channel in channelList. The number of characters in pattern must equal the number of channel specified in channelList or an error is returned.</li> </ul>

previous value.

# niHSDIO_ConfigureInitialStateU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureInitialStateU32 (ViSession vi, ViUInt32 initialState);

Sets the <u>Initial state</u> for a dynamic generation operation. The Initial state of each channel is driven after the session is initiated using <u>niHSDIO_Initiate</u>. Channels remain unchanged until the first waveform sample is generated.

Unlike <u>niHSDIO_ConfigureInitialState</u> which uses a string, this function uses a binary format to only represent high and low. If you require more choices for your Initial state, use the <u>niHSDIO_ConfigureInitialState</u> function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
initialState	ViUInt32	Specifies the bit mask representing the Initial state. High is specified with a 1, and low is specified with a 0.
		The first character in <b>initialState</b> corresponds to the first channel in <b>channelList</b> . The number of characters in the pattern must equal the number of channels specified in <b>channelList</b> or NI- HSDIO returns an error.

# niHSDIO_ConfigureGenerationRepeat

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureGenerationRepeat (ViSession vi, ViInt32 repeatMode, ViInt32 repeatCount);

Specifies the number of times to generate a waveform or whether to generate it continuously. This function is only valid when the **generationMode** parameter of the <u>niHSDIO_ConfigureGenerationMode</u> function is set to NIHSDIO_VAL_WAVEFORM.

Type Description Name ViSession This handle identifies your instrument session. vi **vi** was obtained from niHSDIO InitGenerationSession. repeatMode Vilnt32 Specifies the repeat mode to configure. **Defined Values** NIHSDIO VAL FINITE (16)—Calling the niHSDIO Initiate function generates the named waveform a finite number of times. The number of times is specified by the repeatCount parameter. NIHSDIO VAL CONTINUOUS (17)—Calling the niHSDIO Initiate function generates the named waveform continuously (until the niHSDIO Abort function is called). repeatCount is ignored. **Default Value:** NIHSDIO VAL FINITE Specifies the number of times to generate the repeatCount Vilnt32 waveform. This parameter is ignored if repeatMode is NIHSDIO VAL CONTINUOUS. Default Value: 1

# niHSDIO_ConfigureWaveformToGenerate

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureWaveformToGenerate (ViSession vi, ViConstString waveformName);

Sets the waveform to be generated upon a call to the <u>niHSDIO_Initiate</u> function when <u>NIHSDIO_ATTR_GENERATION_MODE</u> equals NIHSDIO_VAL_WAVEFORM. This function need only be called if multiple waveforms are present in onboard memory (refer to <u>NIHSDIO_ATTR_WAVEFORM_TO_GENERATE</u> for more information).

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies which waveform to generate upon calling <u>niHSDIO_Initiate</u> .

# niHSDIO_AllocateNamedWaveform

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_AllocateNamedWaveform (ViSession vi, ViConstString waveformName, ViInt32 sizeInSamples);

Reserves waveform space in <u>onboard memory</u> and associates a waveform name with it. Individual waveforms are stored contiguously in onboard memory.



**Notes** The <u>niHSDIO_AllocateNamedWaveform</u> function sets aside onboard memory space and associates a string name with that space. The name given to the waveform is the same name used in the <u>write named waveform</u> functions, as well as the name used in scripts.

If not enough space is available to accommodate a waveform of size **sizeInSamples**, an error is returned and no memory space is created.

This function does not change any data on the device itself, but rather adds the named reference in software. Use the <u>write named</u> <u>waveform</u> functions to fill the onboard memory with waveform data to be generated.

Name	Туре	Descript	tion	
vi	ViSession	This han session. <u>niHSDIO</u>	dle identifies your <b>vi</b> was obtained fr <u>InitGenerationSess</u>	instrument om <u>ion</u> .
waveformName	ViConstString	Specifies name to waveforr	s the string represe associate with the n memory.	enting the allocated
		Default '	Value: ""	
sizeInSamples	Vilnt32	Number named v	of samples to alloc vaveform.	ate for the
		The num samples device ye	ber of bits in the a differs depending ou are using.	llocated on the
		Device	Bits Per Sample	
		NI 654 <i>x</i>	32	
		NI 655 <i>x</i>	32	
		NI 656 <i>x</i>	16 in SDR mode 8 in DDR mode	
		Default '	Value: 10000	

<u>Return Value</u>

## niHSDIO_SetNamedWaveformNextWritePosition

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_SetNamedWaveformNextWritePosition (ViSession vi, ViConstString waveformName, ViInt32 position, ViInt32 offset);

Modifies where within a named waveform to next write data. <u>Write named</u> <u>waveform</u> functions always begins writing at the current write position. Existing data in the waveform is overwritten.

The **position** and **offset** parameters are used together to determine the next write position. **position** describes an absolute (NIHSDIO_VAL_START_OF_WAVEFORM) or relative (NIHSDIO_VAL_CURRENT_READ_POSITION) move. **offset** is the number of samples to shift the next write position. You must always set the write position at a position that is a multiple of 32 samples for the NI 654*x*/655*x* devices or 64 samples for the NI 656*x* devices (128 samples if the NI 656*x* is in DDR mode).

Examples of combinations of position and offset Position:

NIHSDIO_VAL_START_OF_WAVEFORM Offset: 0 Effect: Write location becomes the start of waveform.

Position: NIHSDIO_VAL_START_OF_WAVEFORM Offset: 5 Effect: Write location becomes the sixth sample in waveform.

Position: NIHSDIO_VAL_START_OF_WAVEFORM Offset: -1 Effect: ERROR—The device would try to place the write position before start of waveform.

Position: NIHSDIO_VAL_CURRENT_READ_POSITION Offset: 0 Effect: No effect—leaves next write position unchanged.

Position: NIHSDIO_VAL_CURRENT_POSITION Offset: 10 Effect: Shift write position 10 samples ahead from current write location. This position setting is only valid if the current write position plus this offset is in the waveform.

Position: NIHSDIO_VAL_CURRENT_POSITION

Offset: -10

Effect: Shift write position 10 samples back from current location. This position setting is only valid if the current write position is greater than 10.

The write position is moved to the end of the most recently written data after each call to <u>niHSDIO_WriteNamedWaveformU32</u>. Thus you do not need to explicitly call <u>niHSDIO_SetNamedWaveformNextWritePosition</u> unless you want to.

Attempting to set the write position past the end of the allocated space results in an error.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
waveformName	ViConstString	Specifies a string representing the name to associate with the allocated waveform memory.
		Default Value: ""
position	Vilnt32	Specifies where to place the write position (in conjunction with offset):
		Defined Values
		NIHSDIO_VAL_START_OF_WAVEFORM (44)—Offset is relative to the beginning of the waveform.
		NIHSDIO_VAL_CURRENT_POSITION (45)—Offset is relative to the current write position in the waveform.
		<b>Default Value:</b> NIHSDIO_VAL_CURRENT_POSITION
offset	Vilnt32	Specifies the write position of the name waveform in conjunction with the mode attribute. <b>offset</b> is in samples.
		Default Value: 0
<u>Return Value</u>		

# niHSDIO_DeleteNamedWaveform

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_DeleteNamedWaveform (ViSession vi, ViConstString waveformName);

Frees a named waveform space in onboard memory.

Note This function releases onboard memory space previously allocated by either the <u>niHSDIO_AllocateNamedWaveform</u> or <u>Write</u> <u>Named Waveform</u> functions. Any future reference to the deleted waveform results in an error. However, previously written scripts that still reference the deleted waveform do not generate an error at initiation.

An error is generated if the waveform name is not allocated in onboard memory.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
waveformName	ViConstString	Specifies the name of the waveform to delete.
		Default Value: ""

# niHSDIO_ConfigureGenerationMode

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureGenerationMode (ViSession vi, ViInt32 generationMode);

Configures whether to generate the waveform specified in <u>NIHSDIO_ATTR_WAVEFORM_TO_GENERATE</u> or the script specified in <u>NIHSDIO_ATTR_SCRIPT_TO_GENERATE</u> upon calling the <u>niHSDIO_Initiate</u> function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessivity vi was obtained from <u>niHSDIO_InitGenerationSession</u> .
generationMode	Vilnt32	Specifies the generation mode to configure.
		Defined Values
		NIHSDIO_VAL_WAVEFORM (14)—Calling niHSDIO_Initiate generates the named waveform represented by <u>NIHSDIO_ATTR_WAVEFORM_TO_GENERA</u> NIHSDIO_VAL_SCRIPTED (15)—Calling niHSDIO_Initiate generates the script represented by <u>NIHSDIO_ATTR_SCRIPT_TO_GENERATE</u> .
		<b>Default Value:</b> NIHSDIO_VAL_WAVEFORM

# niHSDIO_WriteScript

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_WriteScript (ViSession vi, ViConstString script);

Writes a string containing scripts that govern the generation of waveforms. If this function is called repeatedly, previously written scripts with unique names remain loaded. Previously written scripts with identical names to those being written are replaced.

If multiple scripts are loaded when the <u>niHSDIO_Initiate</u> function is called, then one of the scripts must be designated as the script to generate by setting <u>NIHSDIO_ATTR_SCRIPT_TO_GENERATE</u> to the desired script name. If only one script is in memory, then you do not need to designate the script to generate. All waveforms referenced in the scripts must be written before the script is written.

An error is returned if the script uses incorrect syntax. This function calls the <u>niHSDIO_CommitDynamic</u> function. All pending attributes are committed to hardware.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
script	ViConstString	Specifies a string containing a syntactically correct script.
<u>Return</u>	<u>Value</u>	

# niHSDIO_ConfigureScriptToGenerate

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureScriptToGenerate (ViSession vi, ViConstString scriptName);

Sets the script to be generated upon a call to the <u>niHSDIO_Initiate</u> function when in <u>NIHSDIO_ATTR_GENERATION_MODE</u> equals NIHSDIO_VAL_SCRIPTED. If there are multiple scripts loaded when <u>niHSDIO_Initiate</u> is called, then one script must be designated as the script to generate or you receive an error. This function need only be called if multiple scripts are present in onboard memory (refer to <u>NIHSDIO_ATTR_SCRIPT_TO_GENERATE</u> for more information).

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
scriptName	ViConstString	Specifies which script to generate upon calling <u>niHSDIO_Initiate</u> .

# niHSDIO_ConfigureSampleClock

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureSampleClock (ViSession vi, ViConstString clockSource, ViReal64 clockRate);

Configures the Sample clock. This function allows you to specify the clock source and rate for the Sample clock.

If clockSource is set to NIHSDIO_VAL_ON_BOARD_CLOCK_STR, NI-HSDIO coerces the rate to a value that is supported by the hardware. Use the <u>niHSDIO_GetAttributeViReal64</u> function to get the value for <u>NIHSDIO_ATTR_SAMPLE_CLOCK_RATE</u> to see to what value NI-HSDIO has coerced the Sample clock rate.

**clockSource** can be set to NIHSDIO_VAL_STROBE_STR for acquisition only.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
clockSource	ViConstString	Specifies the Sample clock source. Refer to <u>NIHSDIO_ATTR_SAMPLE_CLOCK_SOURC</u> for details.
		Defined Values
		NIHSDIO_VAL_ON_BOARD_CLOCK_STR ("OnBoardClock")—The device uses the On Board Clock as the Sample clock source. NIHSDIO_VAL_STROBE_STR ("STROBE")— The device uses the signal present on the STROBE channel as the Sample clock source. This choice is only valid for acquisition operations. NIHSDIO_VAL_CLK_IN_STR ("ClkIn")—The device uses the signal present on the front panel CLK IN SMB jack connector as the Sample clock source. NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")—The device uses the signal present on the PXI_STAR line as the Sample clock source. This choice is valid for devices in slots that support PXI_STAR.
		<b>Default Value</b> : NIHSDIO_VAL_ON_BOARD_CLOCK_STR
clockRate	ViReal64	Specifies the Sample clock rate, expressed in Hz. You must set this property even when you supply an external clock because NI- HSDIO uses this property for a number of reasons, including optimal error checking
and certain pulse width selections.

Default Value: 5000000

# niHSDIO_ConfigureDataPosition

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureDataPosition (ViSession vi, ViConstString channelList, ViInt32 position);

Configures channels to be clocked in various ways by the Sample clock edges. You have three options for data position: rising edge, falling edge, or delayed.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_Ir</u>
channelList	ViConstString	Identifies channels to apply settings. Use "" o channels.
position	Vilnt32	Specifies which edge of the Sample clock sig also configure the device to generate data at rising edge of the Sample clock.
		Defined Values
		NIHSDIO_VAL_SAMPLE_CLOCK_RISING_E or generates data on the rising edge of the S NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_ samples or generates data on the falling edge NIHSDIO_VAL_DELAY_FROM_SAMPLE_CL device samples or generates data with delay clock. Specify the delay using NIHSDIO_ATT the niHSDIO_ConfigureDataPositionDelay func
		Notes NIHSDIO_VAL_DELAY_FROM_S has more jitter than the rising or falling e
		Certain devices have sample clock frequ delay is used. Refer to the device docur
		To configure a delay on NI 656 <i>x</i> devices, the device. NI-HSDIO returns an error if partial channel list.
		Default Value: NIHSDIO_VAL_SAMPLE_CL(

# niHSDIO_ConfigureDataPositionDelay

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureDataPositionDelay (ViSession vi, ViConstString channelList, ViReal64 delay);

Sets up the data delay with respect to the Sample clock. To configure the data delay, select

NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE as the value for the **position** parameter of the <u>niHSDIO_ConfigureDataPosition</u> function.



**Note** To configure a delay on NI 656*x* devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels. Default Value: ""
delay	ViReal64	Specifies the delay after the Sample clock rising edge when the device generates or acquires a new data sample. Data delay is expressed as a fraction of the clock period, that is, a fraction of 1/ <u>NIHSDIO_ATTR_SAMPLE_CLOCK_RATE</u> . All the channels in the session that use delayed Sample clock to position data must have the same delay value. <b>Default Value:</b> 0.00

# niHSDIO_ConfigureRefClock

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureRefClock (ViSession vi, ViConstString clockSource, ViReal64 clockRate);

Configures the <u>Reference clock</u>. Use this function when you are using the On Board Clock as a Sample clock, and you want the Sample clock to be phase-locked to a reference signal. Phase-locking the Sample clock to a Reference clock prevents the Sample clock from "drifting" relative to the Reference clock.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
clockSource	ViConstString	Specifies the PLL Reference clock source. Refer to <u>NIHSDIO_ATTR_REF_CLOCK_SOURCE</u> for details.
		Defined Values
		NIHSDIO_VAL_NONE_STR ("None")—The device does not use a Reference clock. NIHSDIO_VAL_CLK_IN_STR ("ClkIn")— The device uses the signal present on the front panel CLK IN SMB jack connector as the Reference clock source. NIHSDIO_VAL_PXI_CLK10_STR ("PXI_CLK10")—The device uses the 10 MHz PXI backplane clock as the Reference clock source. This source is only available for PXI devices. NIHSDIO_VAL_RTSI7_STR ("RTSI7")—The device uses the signal on RTSI 7 as the Reference clock source. This source is only available for PCI devices.
		Default Value: NIHSDIO_VAL_NONE_STR
clockRate	ViReal64	Specifies the Reference clock rate, expressed in Hz. Refer to <u>NIHSDIO_ATTR_REF_CLOCK_RATE</u> for details.
		Default Value: 10000000

# niHSDIO_AdjustSampleClockRelativeDelay

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_AdjustSampleClockRelativeDelay (ViSession vi, ViReal64 adjustmentTime);

Delays the Sample clock relative to the Reference clock. Use this function to align the Sample clock of your device to the Sample clock of another device in your system.

Only call this function after your session is committed. The effect of this function is immediate.

This function generates an error if <u>NIHSDIO_ATTR_REF_CLOCK_SOURCE</u> is set to NIHSDIO_VAL_NONE_STR.

This function can only align the device Sample clock to another Sample clock if the other device is using the same Reference clock source.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the function used to initialize the session.
adjustmentTime	ViReal64	Specifies the time in seconds to delay the Sample clock. Values range between 0 and the Sample clock period (1/ <u>NIHSDIO_ATTR_SAMPLE_CLOCK_RATF</u>
Poturn Valuo		

<u>Return Value</u>

# niHSDIO_ConfigureDigitalEdgeStartTrigger

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureDigitalEdgeStartTrigger (ViSession vi, ViConstString source, ViInt32 edge);

Configures the <u>Start trigger</u> for edge triggering.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> v obtained from the <u>niHSDIO_InitAcquisitionSession</u> ( <u>niHSDIO_InitGenerationSession</u> function.
source	ViConstString	You may specify any valid source terminal for this Trigger voltages and positions are only relevant if of the trigger is from the front panel connectors.
edge	Vilnt32	Refer to <u>NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGI</u> for possible values. Specifies the edge to detect.
		Defined Values
		NIHSDIO_VAL_RISING_EDGE (12)—Rising edge NIHSDIO_VAL_FALLING_EDGE (13)—Falling edç
		Default Value: NIHSDIO_VAL_RISING_EDGE
Return V	/alue	

# niHSDIO_ConfigurePatternMatchStartTrigger

## **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchStartTrigger (ViSession vi, ViConstString channelList, ViConstString pattern, ViInt32 triggerWhen);

Configures the <u>Start trigger</u> for pattern-match triggering. This function is only valid for acquisition operations.

#### Valid Syntax:

Both of the following examples are valid and do the same thing. The order of **channelList** determines the order of the pattern string.

- niHSDIO_ConfigurePatternMatchStartTrigger (vi, "19-0", "0000 0XXX XX11 1111 1111");
- niHSDIO_ConfigurePatternMatchStartTrigger (vi, "0-19", "1111 1111 11XX XXX0 0000");

Note The logic levels seen by pattern matching are affected by data interpretation.

Name	Туре	Description	
vi	ViSession	This handle identifies your instrument sessic <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .	
channelList	ViConstString	This string specifies which channels will be configured for pattern matching using the pattern string. The order of channels in the li determines the order of the <b>pattern</b> string.	
pattern	ViConstString	<ul> <li>This string expression describes the pattern be matched. This expression is composed o the following characters:</li> <li>X or x: Ignore the channel</li> <li>1: Match on a logic 1</li> <li>0: Match on a logic 0</li> <li>R or r: Match on a rising edge</li> <li>F or f: Match on a falling edge</li> <li>E or e: Match on either edge</li> </ul> The first character in the expression corresponds to the first channel in channelList. The number of characters in pattern must correspond to the number of channels specified in channelList.	
triggerWhen	Vilnt32	Specifies the when the trigger asserts.	
		Defined Values	
		NIHSDIO_VAL_PATTERN_MATCHES (36)— The trigger asserts when the pattern matche NIHSDIO_VAL_PATTERN_DOES_NOT_MAT (37)—The trigger asserts when the pattern does not match.	

#### **Default Value:**

#### NIHSDIO_VAL_PATTERN_MATCHES

# niHSDIO_ConfigurePatternMatchStartTriggerU32

### **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchStartTriggerU32 (ViSession vi, ViConstString channelList, ViUInt32 pattern, ViInt32 triggerWhen);

Configures the <u>Start trigger</u> for pattern-match triggering.

Unlike <u>niHSDIO_ConfigurePatternMatchStartTrigger</u> which uses a string, this function uses a binary format to only represent high and low. If you require more choices for your pattern, use the <u>niHSDIO_ConfigurePatternMatchStartTrigger</u> function.

This function is only valid for acquisition operations.



**Note** The logic levels seen by pattern matching are affected by data interpretation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels will be configured for pattern matching using the pattern string. The order of channels in the li determines the order of the <b>pattern</b> string.
pattern	ViUInt32	Specifies the binary pattern that activates the pattern match trigger under the conditions specified in <b>triggerWhen</b> .
		Bits on channels not specified in <b>channelLis</b> are ignored.
triggerWhen	Vilnt32	Specifies the when the trigger asserts.
		Defined Values
		NIHSDIO_VAL_PATTERN_MATCHES (36)— The trigger asserts when the pattern matche NIHSDIO_VAL_PATTERN_DOES_NOT_MAT (37)—The trigger asserts when the pattern does not match.
		<b>Default Value:</b> NIHSDIO_VAL_PATTERN_MATCHES

# niHSDIO_ConfigureSoftwareStartTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureSoftwareStartTrigger (ViSession vi);

Configures the <u>Start trigger</u> for software triggering.

Refer to <u>niHSDIO_SendSoftwareEdgeTrigger</u> for more information about using the software Start trigger.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_DisableStartTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_DisableStartTrigger (ViSession vi);

Configures the device to not wait for a <u>Start trigger</u> after the <u>niHSDIO_Initiate</u> function is called. Calling this function is only necessary if you have configured a Start trigger and now want to disable it.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_ConfigureDigitalEdgeRefTrigger

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigureDigitalEdgeRefTrigger (ViSession vi, ViConstString source, ViInt32 edge, ViInt32 pretriggerSamples);

Configures the <u>Reference trigger</u> for edge triggering in an acquisition. If the Reference trigger asserts before all the pretrigger samples are acquired, then it is ignored. This function is only valid for acquisition operations.

Туре	Description
ViSession	This handle identifies your instrument obtained from <u>niHSDIO_InitAcquisitior</u>
ViConstString	You may specify any valid source terr Trigger voltages and positions are on source of the trigger is from the front
	Refer to <u>NIHSDIO_ATTR_DIGITAL_EDGE_RE</u> for possible values.
Vilnt32	Specifies the edge to detect.
	Defined Values
	NIHSDIO_VAL_RISING_EDGE (12)— NIHSDIO_VAL_FALLING_EDGE (13)
	Default Value: NIHSDIO_VAL_RISIN
Vilnt32	Specifies the number of necessary pr before the Reference trigger is ackno
	Type ViSession ViConstString ViInt32 ViInt32

# niHSDIO_ConfigurePatternMatchRefTrigger

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchRefTrigger (ViSession vi, ViConstString channelList, ViConstString pattern, ViInt32 triggerWhen, ViInt32 pretriggerSamples);

Configures the <u>Reference trigger</u> for pattern-match triggering. If the Reference trigger asserts before all the pretrigger samples are acquired, then it is ignored. This function is only valid for acquisition sessions.

#### Valid Syntax:

Both of the following examples are valid and do the same thing. The order of **channelList** determines the order of the pattern string.

- niHSDIO_ConfigurePatternMatchAdvanceTrigger (vi, "19-0", "0000 0XXX XX11 1111 1111");
- niHSDIO_ConfigurePatternMatchAdvanceTrigger (vi, "0-19", "1111 1111 11XX XXX0 0000");



Note The logic levels seen by pattern matching are affected by data interpretation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels <i>a</i> configured for pattern matching using <b>pattern</b> string. The order of channels determines the order of the <b>pattern</b> s
pattern	ViConstString	<ul> <li>This string expression describes the p be matched. This expression is comp the following characters:</li> <li>X or x: Ignore the channel</li> <li>1: Match on a logic 1</li> <li>0: Match on a logic 0</li> <li>R or r: Match on a rising edge</li> <li>F or f: Match on a falling edge</li> <li>E or e: Match on either edge</li> </ul> The first character in the expression corresponds to the first channel in channelList. The number of characte pattern must correspond to the numb channels specified in channelList, or is returned.
triggerWhen	Vilnt32	Specifies when the trigger asserts.
		Defined Values
		NIHSDIO_VAL_PATTERN_MATCHES The trigger asserts when the pattern I NIHSDIO_VAL_PATTERN_DOES_NO' (37)—The trigger asserts when the pa does not match.
pretriggerSamples Vilnt32

**Default Value:** 

NIHSDIO_VAL_PATTERN_MATCHES

Specifies the number of necessary pr samples before the Reference trigger acknowledged.

# niHSDIO_ConfigurePatternMatchRefTriggerU32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchRefTriggerU32 (ViSession vi, ViConstString channelList, ViUInt32 pattern, ViInt32 triggerWhen, ViInt32 pretriggerSamples);

Configures the <u>Reference trigger</u> for pattern-match triggering. If the Reference trigger asserts before all the pretrigger samples are acquired, then it is ignored.

Unlike <u>niHSDIO_ConfigurePatternMatchRefTrigger</u> which uses a string, this function uses a binary format to only represent high and low. If you require more choices for your pattern, use the <u>niHSDIO_ConfigurePatternMatchRefTrigger</u> function.

This function is only valid for acquisition sessions.



**Note** The logic levels seen by pattern matching are affected by data interpretation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels <i>a</i> configured for pattern matching using <b>pattern</b> string. The order of channels determines the order of the <b>pattern</b> s
pattern	ViUInt32	Specifies the binary pattern that active pattern match trigger under the condition specified in <b>triggerWhen</b> .
		Bits on channels not specified in <b>cha</b> i are ignored.
triggerWhen	Vilnt32	Specifies when the trigger asserts.
		Defined Values
		NIHSDIO_VAL_PATTERN_MATCHES The trigger asserts when the pattern I NIHSDIO_VAL_PATTERN_DOES_NO' (37)—The trigger asserts when the pa does not match.
		<b>Default Value:</b> NIHSDIO_VAL_PATTERN_MATCHES
pretriggerSamples	Vilnt32	Specifies the number of necessary pr samples before the Reference trigger acknowledged.
<u>Return Value</u>		

# niHSDIO_ConfigureSoftwareRefTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureSoftwareRefTrigger (ViSession vi, ViInt32 pretriggerSamples);

Configures the <u>Reference trigger</u> for software triggering. If the Reference trigger asserts before all the pretrigger samples are acquired, then it is ignored. This function is valid only for acquisition sessions.

Refer to <u>niHSDIO_SendSoftwareEdgeTrigger</u> for more information about the software Reference trigger.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
pretriggerSamples	Vilnt32	Specifies the number of necessary pretrigger samples before the Reference trigger is acknowledged.
		Default Value: 500

# niHSDIO_DisableRefTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_DisableRefTrigger (ViSession vi);

Configures the acquisition operation to have no <u>Reference trigger</u>. Calling this function is only necessary if you have configured a Reference trigger and now want to disable it. This function is valid only for acquisition sessions.

Name Type Description

**vi** ViSession This handle identifies your instrument session. **vi** was obtained from <u>niHSDIO_InitAcquisitionSession</u>.

# niHSDIO_ConfigureDigitalEdgeAdvanceTrigger

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureDigitalEdgeAdvanceTrigger (ViSession vi, ViConstString source, ViInt32 edge);

Configures the <u>Advance trigger</u> for edge triggering.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> v from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
source	ViConstString	You may specify any valid source terminal for this voltages and positions are only relevant if the sour is from the front panel connectors.
edge	Vilnt32	Refer to <u>NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRI</u> for possible values. Specifies the edge to detect.
		Defined Values
		NIHSDIO_VAL_RISING_EDGE (12)—Rising edge NIHSDIO_VAL_FALLING_EDGE (13)—Falling edç
		Default Value: NIHSDIO_VAL_RISING_EDGE
Return V	/alue	

# niHSDIO_ConfigurePatternMatchAdvanceTrigger

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchAdvanceTrigger (ViSession vi, ViConstString channelList, ViConstString pattern, ViInt32 triggerWhen);

Configures the <u>Advance trigger</u> for pattern-match triggering. This function is only valid for acquisition operations.

### Valid Syntax:

Both of the following examples are valid and do the same thing. The order of **channelList** determines the order of the pattern string.

- niHSDIO_ConfigurePatternMatchAdvanceTrigger (vi, "19-0", "0000 0XXX XX11 1111 1111");
- niHSDIO_ConfigurePatternMatchAdvanceTrigger (vi, "0-19", "1111 1111 11XX XXX0 0000");

Note The logic levels seen by pattern matching are affected by data interpretation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels will be configured for pattern matching using the pattern string. The order of channels in the li determines the order of the <b>pattern</b> string.
pattern	ViConstString	<ul> <li>This string expression describes the pattern be matched. This expression is composed o the following characters:</li> <li>X or x: Ignore the channel</li> <li>1: Match on a logic 1</li> <li>0: Match on a logic 0</li> <li>R or r: Match on a rising edge</li> <li>F or f: Match on a falling edge</li> <li>E or e: Match on either edge</li> </ul> The first character in the expression corresponds to the first channel in channelList. The number of characters in pattern must correspond to the number of channels specified in channelList.
triggerWhen	Vilnt32	Specifies the when the trigger asserts.
		Defined Values
		NIHSDIO_VAL_PATTERN_MATCHES (36)— The trigger asserts when the pattern matche NIHSDIO_VAL_PATTERN_DOES_NOT_MAT (37)—The trigger asserts when the pattern does not match.

### **Default Value:**

### NIHSDIO_VAL_PATTERN_MATCHES

# niHSDIO_ConfigurePatternMatchAdvanceTrigger

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchAdvanceTriggerU32 (ViSession vi, ViConstString channelList, ViUInt32 pattern, ViInt32 triggerWhen);

Configures the <u>Advance trigger</u> for pattern-match triggering.

Unlike <u>niHSDIO_ConfigurePatternMatchAdvanceTrigger</u> which uses a string, this function uses a binary format to only represent high and low. If you require more choices for your pattern, use the <u>niHSDIO_ConfigurePatternMatchAdvanceTrigger</u> function.

This function is only valid for acquisition operations.



**Note** The logic levels seen by pattern matching are affected by data interpretation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels will be configured for pattern matching using the pattern string. The order of channels in the li determines the order of the <b>pattern</b> string.
pattern	ViUInt32	Specifies the binary pattern that activates the pattern match trigger under the conditions specified in <b>triggerWhen</b> .
		Bits on channels not specified in <b>channelLis</b> are ignored.
triggerWhen	Vilnt32	Specifies the when the trigger asserts.
		Defined Values
		NIHSDIO_VAL_PATTERN_MATCHES (36)— The trigger asserts when the pattern matche NIHSDIO_VAL_PATTERN_DOES_NOT_MAT (37)—The trigger asserts when the pattern does not match.
		<b>Default Value:</b> NIHSDIO_VAL_PATTERN_MATCHES

# niHSDIO_ConfigureSoftwareAdvanceTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureSoftwareAdvanceTrigger (ViSession vi);

Configures the <u>Advance trigger</u> for software triggering.

Refer to <u>niHSDIO_SendSoftwareEdgeTrigger</u> for more information about using the software Advance trigger.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_DisableAdvanceTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_DisableAdvanceTrigger (ViSession vi);

Configures the device to not wait for a <u>Advance trigger</u> after the <u>niHSDIO_Initiate</u> function is called. Calling this function is only necessary if you have configured a Advance trigger and now want to disable it.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_ConfigureDigitalLevelPauseTrigger

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureDigitalLevelPauseTrigger (ViSession vi, ViConstString source, ViInt32 triggerWhen);

Configures the <u>Pause trigger</u> for level triggering. The operation is paused when the trigger is active.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
source	ViConstString	You may specify any valid source terminal fc Trigger voltages and positions are only relev the trigger is from the front panel connectors
triggerWhen	Vilnt32	Refer to <u>NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_'</u> for possible values. Specifies the active level for the desired trig(
		Defined Values
		NIHSDIO_VAL_HIGH (34)—Trigger is active high level. NIHSDIO_VAL_LOW (35)—Trigger is active low level.
		Default Value: NIHSDIO_VAL_HIGH
<u>Return Value</u>		

# niHSDIO_ConfigurePatternMatchPauseTrigger

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchPauseTrigger (ViSession vi, ViConstString channelList, ViConstString pattern, ViInt32 triggerWhen);

Configures the <u>Pause trigger</u> for pattern-match triggering. This function is valid only for acquisition sessions.

#### Valid Syntax:

Both of the following examples are valid and do the same thing. The order of **channelList** determines the order of the pattern string.

- niHSDIO_ConfigurePatternMatchPauseTrigger (vi, "19-0", "0000 0XXX XX11 1111 1111");
- niHSDIO_ConfigurePatternMatchPauseTrigger (vi, "0-19", "1111 1111 11XX XXX0 0000");

Note The values seen by pattern matching are affected by <u>data</u> <u>interpretation</u>.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels are configured for pattern matching using the pattern string. The order of channels in the li determines the order of the <b>pattern</b> string. E "0-19" and "19-0" are reverse of one anothe
pattern	ViConstString	<ul> <li>This string expression describes the pattern be matched. The pattern specifies an edge of level transition that must occur before the trigger is recognized. The first sample acquir will be the first full sample following the configured transition.</li> <li>This expression is composed of characters: <ul> <li>X or x: Ignore the channel</li> <li>1: Match on a logic 1</li> <li>0: Match on a logic 0</li> <li>R or r: Match on a rising edge</li> <li>F or f: Match on a falling edge</li> <li>E or e: Match on either edge</li> </ul> </li> <li>The first character in the expression corresponds to the first channel in channelList. The number of characters in pattern must correspond to the number of channels specified in channelList or an errc</li> </ul>
triggerWhen	Vilnt32	IS returned.
unggerwiten	VIIIIIJZ	כאברוובא אוובוו נווב נווטטבו מאשרונא.

### **Defined Values**

NIHSDIO_VAL_PATTERN_MATCHES (36)— The trigger asserts when the pattern matche NIHSDIO_VAL_PATTERN_DOES_NOT_MAT (37)—The trigger asserts when the pattern does not match.

### Default Value:

NIHSDIO_VAL_PATTERN_MATCHES

# niHSDIO_ConfigurePatternMatchPauseTriggerU3

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_ConfigurePatternMatchPauseTriggerU32 (ViSession vi, ViConstString channelList, ViUInt32 pattern, ViInt32 triggerWhen);

Configures the <u>Pause trigger</u> for pattern-match triggering.

Unlike <u>niHSDIO_ConfigurePatternMatchPauseTrigger</u> which uses a string, this function uses a binary format to only represent high and low. If you require more choices for your pattern, use the <u>niHSDIO_ConfigurePatternMatchPauseTrigger</u> function.

This function is valid only for acquisition sessions.



**Note** The values seen by pattern matching are affected by <u>data</u> <u>interpretation</u>.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
channelList	ViConstString	This string specifies which channels are configured for pattern matching using the pattern string. The order of channels in the li determines the order of the <b>pattern</b> string. E "0-19" and "19-0" are reverse of one anothe
pattern	ViUInt32	Specifies the binary pattern that activates the pattern match trigger under the conditions specified in <b>triggerWhen</b> .
		Bits on channels not specified in <b>channelLi</b> : are ignored.
triggerWhen	Vilnt32	Specifies when the trigger asserts.
		Defined Values
		NIHSDIO_VAL_PATTERN_MATCHES (36)— The trigger asserts when the pattern matche NIHSDIO_VAL_PATTERN_DOES_NOT_MAT (37)—The trigger asserts when the pattern does not match.
		<b>Default Value:</b> NIHSDIO_VAL_PATTERN_MATCHES
# niHSDIO_DisablePauseTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_DisablePauseTrigger (ViSession vi);

Sets the data operation to have no <u>Pause trigger</u>. Calling this function is only necessary if you have configured a Pause trigger and now want to disable it.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_ConfigureDigitalEdgeScriptTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureDigitalEdgeScriptTrigger (ViSession vi, ViConstString triggerID, ViConstString source, ViInt32 edge);

Configures the <u>Script trigger</u> for edge triggering. This function is only valid for generation sessions that use <u>scripting</u>.

Description Name Type

vi ViSession

This handle identifies your instrument session. v

from niHSDIO InitGenerationSession.

triggerID ViConstString Identifies which Script trigger this function confic

#### **Defined Values**

"ScriptTrigger0" "ScriptTrigger1" "ScriptTrigger2" "ScriptTrigger3"

**Default Value:** ScriptTrigger0

ViConstString You may specify any valid source terminal for th source Trigger voltages and positions are only relevant the trigger is from the front panel connectors.

#### Refer to

NIHSDIO ATTR DIGITAL EDGE SCRIPT TRIC for possible values.

#### edge Specifies the edge to detect. Vilnt32

#### **Defined Values**

NIHSDIO_VAL_RISING_EDGE (12)—Rising edg NIHSDIO_VAL_FALLING_EDGE (13)—Falling e

Default Value: NIHSDIO VAL RISING EDGE

# niHSDIO_ConfigureDigitalLevelScriptTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureDigitalLevelScriptTrigger (ViSession vi, ViConstString triggerID, ViConstString source, ViInt32 triggerWhen);

Configures the <u>Script trigger</u> for level triggering. This function is only valid for generation sessions that use <u>scripting</u>.

Parameters		
Name	Туре	Description
vi	ViSession	This handle identifies your instrument sessic from <u>niHSDIO_InitGenerationSession</u> .
triggerID	ViConstString	Identifies which script trigger this function co
		Defined Values
		"ScriptTrigger0" "ScriptTrigger1" "ScriptTrigger2" "ScriptTrigger3"
		Default Value: "ScriptTrigger0"
source	ViConstString	You may specify any valid source terminal fc Trigger voltages and positions are only relev the trigger is from the front panel connectors
trigger/A/ben	\/ilet22	Refer to <u>NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT</u> for possible values. Specifies the pative level for the desired trigg
triggerwnen	VIINt32	Specifies the active level for the desired trig(
		Defined Values
		NIHSDIO_VAL_HIGH (34)—Trigger is active high level. NIHSDIO_VAL_LOW (35)—Trigger is active low level.
		Default Value: NIHSDIO_VAL_HIGH
<u>Return Value</u>		

# niHSDIO_ConfigureSoftwareScriptTrigger

### **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ConfigureSoftwareScriptTrigger (ViSession vi, ViConstString triggerID);

Configures the <u>Script trigger</u> for software triggering. This function is only valid for generation sessions that use <u>scripting</u>.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
triggerID	ViConstString	Identifies which script trigger this function configures.
		Defined Values
		"ScriptTrigger0"
		"ScriptTrigger1"
		"ScriptTrigger2"
		"ScriptTrigger3"

Default Value: "ScriptTrigger0"

# niHSDIO_DisableScriptTrigger

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_DisableScriptTrigger (ViSession vi, ViConstString triggerID);

Sets the data operation to not have a <u>Script trigger</u>. Calling this function is only necessary if you have configured a particular Script trigger and now want to disable it. This function is only valid for generation sessions.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
triggerID	ViConstString	Identifies which Script trigger this function will configure.
		Defined Values
		"ScriptTrigger0"
		"ScriptTrigger1"
		"ScriptTrigger2"
		"ScriptTrigger3"

Default Value: "ScriptTrigger0"

# niHSDIO_SendSoftwareEdgeTrigger

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_SendSoftwareEdgeTrigger (ViSession vi, ViInt32 trigger, ViConstString triggerIdentifier);

Use this function to force a particular edge-based trigger to occur. This function applies to the Start, Reference, Advance, and Script triggers, and is valid if the particular trigger is configured for edge, pattern match, or software triggering (for edge or pattern-match triggers you can use <u>niHSDIO_SendSoftwareEdgeTrigger</u> as a software override).

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
trigger	Vilnt32	The trigger to assert.
		Defined Values
		NIHSDIO_VAL_START_TRIGGER (53) —Start trigger for dynamic acquisition or generation. NIHSDIO_VAL_REF_TRIGGER (54)— Reference trigger for dynamic acquisition. NIHSDIO_VAL_SCRIPT_TRIGGER (58) —Script trigger for dynamic generation. NIHSDIO_VAL_ADVANCE_TRIGGER (61)—Advance trigger for dynamic acquisition.
		<b>Default Value:</b> NIHSDIO_VAL_START_TRIGGER
triggerldentifier	ViConstString	Describes the software trigger. For example, "ScriptTrigger0" could be the identifier for the Script trigger, or you could have an empty string for the Start and Reference triggers.
		Defined Values
		"ScriptTrigger0" "ScriptTrigger1" "ScriptTrigger2" "ScriptTrigger3" "" (empty string) or VI_NULL

Default Value: "" (empty string)

# niHSDIO_ExportSignal

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ExportSignal (ViSession vi, ViInt32 signal, ViConstString signalIdentifier, ViConstString outputTerminal);

Use this function to route signals (clocks, triggers, and events) to the output terminal you specify. Refer to your device documentation for valid signal destinations.

Any routes created within a session persist after the session closes to prevent signal glitching. To unconfigure signal routes created in previous sessions, set the **resetInstrument** parameter in <u>niHSDIO_InitGenerationSession</u> or <u>niHSDIO_InitAcquisitionSession</u> to VI_TRUE or use <u>niHSDIO_reset</u>.

If you export a signal with this function and commit the session, the signal is routed to the output terminal you specify. If you then reconfigure the signal to have a different output terminal, the previous output terminal is tristated after the session is committed. If you change the output terminal to NIHSDIO_VAL_DO_NOT_EXPORT_STR or an empty string when you commit the operation, the previous output terminal is tristated.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument ses obtained from the <u>niHSDIO_InitAcquisition</u> <u>niHSDIO_InitGenerationSession</u> function.
signal	Vilnt32	Signal (clock, trigger, or event) to export.
		Defined Values
		<ul> <li>NIHSDIO_VAL_SAMPLE_CLOCK (51)—I clock.</li> <li>NIHSDIO_VAL_REF_CLOCK (52)—Devic NIHSDIO_VAL_START_TRIGGER (53)—I NIHSDIO_VAL_REF_TRIGGER (54)—Devingger (dynamic acquisition only).</li> <li>NIHSDIO_VAL_ADVANCE_TRIGGER (61 trigger (dynamic acquisition only)</li> <li>NIHSDIO_VAL_PAUSE_TRIGGER (57)—I trigger (dynamic generation only).</li> <li>NIHSDIO_VAL_SCRIPT_TRIGGER (58)—trigger (dynamic generation only—require to describe a particular Script trigger).</li> <li>NIHSDIO_VAL_DATA_ACTIVE_EVENT (event (dynamic generation only).</li> <li>NIHSDIO_VAL_MARKER_EVENT (59)—(dynamic generation only—requires sign describe a particular marker).</li> <li>NIHSDIO_VAL_READY_FOR_START_EV for Start event.</li> <li>NIHSDIO_VAL_READY_FOR_ADVANCE Ready for Advance event (dynamic acquisition only).</li> <li>NIHSDIO_VAL_END_OF_RECORD_EVER Record event (dynamic acquisition only).</li> <li>NIHSDIO_VAL_ONBOARD_REF_CLOCK onboard Reference clock (PCI devices or Default Value: NIHSDIO_VAL_SAMPLE</li> </ul>
		Delauit value: NIHSDIO_VAL_SAMPLE_

signalIdentifier ViConstString Describes the signal being exported.

### **Defined Values**

- "ScriptTrigger0"
- "ScriptTrigger1"
- "ScriptTrigger2"
- "ScriptTrigger3"
- "Marker0"
- "Marker1"
- "Marker2"
- "Marker3"
- "" (empty String)
- VI_NULL

### **Default Value:** "" (empty String)

outputTerminal ViConstString Output terminal where the signal is expor

- NIHSDIO_VAL_PFI0_STR—NIHSD PFI connectors
- NIHSDIO_VAL_PXI_TRIG0_STR-NIHSDIO_VAL_PXI_TRIG6_STR : t backplane (for PXI devices only)
- NIHSDIO_VAL_PXI_TRIG7/NIHSD —NIHSDIO_VAL_RTSI6_STR : the (for PCI devices only)
- NIHSDIO_VAL_RTSI7_STR : RTSI designated for the Onboard Refere
  - Note NI-HSDIO returns an e any signal other than the Onl PXI Trigger Line 7/RTSI 7. TI Clock is only available on PC
- NIHSDIO_VAL_CLK_OUT_STR—C connector on the front panel
- NIHSDIO_VAL_DDC_CLK_OUT_S' OUT terminal in the DDC connecto

 "" (empty string) or VI_NULL—the exported

Trigger and event voltages and pos relevant if the destination of the eve front panel connectors.

# niHSDIO_AssignStaticChannels

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_AssignStaticChannels (ViSession vi, ViConstString channelList);

Use this function to configure channels for static acquisition (if **vi** is an acquisition session) or static generation (if **vi** is a generation session). A channel cannot be simultaneously assigned to a static generation and dynamic generation.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	This string identifies which channels will be configured as static.
		Valid Syntax
		"0-19" or "0-15,16-19" or "0-18,19" "" (empty string) or VI_NULL to specify all channels "none" to unassign all channels Channels cannot be configured for both
		static generation and dynamic generation.
<u>Return Value</u>		

# niHSDIO_ReadStaticU32

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ReadStaticU32 (ViSession vi, ViUInt32* readData);

This function immediately reads the digital value on channels configured for static acquisition. Configure a channel for static acquisition using the <u>niHSDIO_AssignStaticChannels</u> function. Channels not configured for static acquisition return a zero.

Values obtained from static read operations are affected by <u>data</u> <u>interpretation</u>.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitAcquisitionSession</u> .
readData	ViUInt32	Bit-value of data read from channels configured for static acquisition.
		The least significant bit of <b>readData</b> corresponds to the lowest physical channel number (for example, <b>readData</b> of 0x00F0 means channels 4- 7 are logic one, while the remaining channels are logic zero or are not configured for static acquisition).

# niHSDIO_WriteStaticU32

## **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_WriteStaticU32 (ViSession vi, ViUInt32 writeData, ViUInt32 channelMask);

This function writes to channels configured for static generation. You can configure a channel for static generation using the <u>niHSDIO_AssignStaticChannels</u> function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
writeData	ViUInt32	Bit-value of data to drive on channels configured for static generation. 1 corresponds to logic high, 0 corresponds to logic low.
		The least significant bit of <b>writeData</b> corresponds to the lowest physical channel number (for example, <b>writeData</b> of 0xFF00 means set the lower eight channels to 0, while setting the upper eight channels to logic high.
		Data values in <b>writeData</b> corresponding to channels not configured for static generation are ignored.
		Static channels explicitly disabled with the <u>niHSDIO_TristateChannels</u> function remain disabled, but the channel data value changes internally. Re-enabling a channel with <u>niHSDIO_TristateChannels</u> causes the channel to drive any value that you have written to it, even while the channel was disabled.
channelMask	ViUInt32	Bit-value of channels to leave unchanged. 1 means to change the channel to whatever is reflected by <b>writeData</b> . 0 means do not alter the channel, regardless of <b>writeData</b> .
		The least significant bit of <b>channelMask</b> corresponds to the lowest physical channel number (for example, <b>writeData</b> of 0xFFFF and <b>channelMask</b> of 0x0080 means set only channel 7 to 1: all other channels remain

unchanged).

### Default Value: -1

# niHSDIO_SelfCal

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_SelfCal (ViSession vi);

This function self-calibrates the device. During self-calibration, the VCXO oscillator phase D/A converters are recalibrated.
Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the function used to initialize the session.

# niHSDIO_ChangeExtCalPassword

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_ChangeExtCalPassword (ViSession vi, ViConstString oldPassword, ViConstString newPassword);

Changes the password that is required to initialize an external calibration session. The password may be up to four characters long.

You can call this function from an acquisition, generation, or calibration session.

Name	Туре	Description
vi	ViSession	The session handle returned from <u>niHSDIO_InitAcquisitionSession</u> , <u>niHSDIO_InitGenerationSession</u> , or <u>niHSDIO_InitExtCal</u> .
oldPassword	ViConstString	The old (current) external calibration password.
		Default Value: ""
newPassword	ViConstString	The new (desired) external calibration password.
		Default Value: ""

# niHSDIO_InitExtCal

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_InitExtCal (ViRsrc resourceName, ViConstString password, ViSession* vi);

Creates and initializes a special NI-HSDIO external calibration session. The ViSession returned is an NI-HSDIO session that can be used during the calibration session.

Multiple calls to this function return the same session ID. Calibration sessions are mutually exclusive with acquisition and generation sessions.

Name	Туре	Description
resourceName	ViRsrc	Specifies the device name, for example "Dev1" where "Dev1" is a device name assigned by Measurement & Automation Explorer.
		Default Value: "PXI1Slot3"
password	ViConstString	The calibration password required to open an external calibration session to the device.
		Default Value: ""
vi	ViSession	Returns a session handle that you use to identify the instrument in all subsequent NI-HSDIO function calls.
<u>Return Value</u>		

# niHSDIO_CalAdjustChannelVoltage

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_CalAdjustVoltage (ViSession vi, ViConstString channelList);

Adjusts the voltage of the selected channel(s). The only errors that can be returned are actual calibration process errors.



**Notes** This function is not supported for the NI 654*x*/656*x* devices.

This function runs a static loopback test before adjusting the voltage. You must disconnect the cable from your device to run this function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitExtCal</u> .
channelList	ViConstString	Identifies channels on which voltage will be

adjusted.

# niHSDIO_CloseExtCal

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_CloseExtCal (ViSession vi, ViInt32 action);

Closes an NI-HSDIO external calibration session and, if specified, stores the new calibration constants and calibration data in the onboard EEPROM.



**Note** Whether you commit or cancel, the device is reset and the FPGA is reloaded afterwards.

Name	Туре	Description
vi	ViSession	The session handle that you obtain from <u>niHSDIO_InitExtCal</u> . The handle identifies a particular instrument session.
action	Vilnt32	The action to perform upon closing.
		Defined Values
		NIHSDIO_VAL_EXT_CAL_COMMIT (62)—The new calibration constants and data determined during the external calibration session are stored in the onboard EEPROM, given that the calibration was complete and passed successfully. NIHSDIO_VAL_EXT_CAL_CANCEL (63)—No changes are made to the calibration constants and data in the EEPROM.

# niHSDIO_TristateChannels

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_TristateChannels (ViSession vi, ViConstString channelList, ViBoolean tristate);

Use this function to force a channel into a high-impedance state. The effect is immediate—it does not require the session be committed. The channel will remain tristated regardless of what other software commands are called. Call this function again and pass VI_FALSE into the **tristate** parameter to allow other software commands to control the channel normally.

Channels are kept in a high-impedance state while the session remains open. Closing the session does not affect the high-impedance state of the channel, but future sessions can now control it.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from <u>niHSDIO_InitGenerationSession</u> .
channelList	ViConstString	This string identifies which channels will be tristated. Channels not specified in this list are unaffected.
		Syntax examples: "2-15" or "0-3, 5, 8-15" or "0, 3, 10"
tristate	ViBoolean	Specifies whether the channels specified in <b>channelList</b> remain tristated, ignoring future software commands.
		Defined Values
		<ul> <li>VI_TRUE (1)—The channels specified in channelList remain tristated, ignoring future software commands.</li> <li>VI_FALSE (0)—The channels specified in channelList are untristated by future software commands.</li> </ul>
		Default Value: VI_TRUE

# niHSDIO_CommitDynamic

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_CommitDynamic (ViSession vi);

Programs the hardware for the dynamic data operation using the attributes you select. Before entering the committed state, most attribute values are stored in software only; these values have not yet been programmed to the hardware. Once the session is committed, the hardware is configured.

For many operations it is not necessary to explicitly call this function because the following functions implicitly commit:

- <u>niHSDIO_Initiate</u>
- <u>niHSDIO_WriteScript</u>
- Read Waveform functions
- Write Named Waveform functions

Start the operation with <u>niHSDIO_Initiate</u>. Running this function while a dynamic operation is in progress returns an error. Committing only programs attributes changed since previous commits.

Note Committing some attributes may have immediate effects seen on external instrument connectors. Voltage levels are an example of an attribute with an immediate effect when committed.

Before committing a session that requires an external clock, ensure the external clock is available. Otherwise you receive an error that the device could not find or lock to the external clock.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the function used to initialize the session.

# niHSDIO_CommitStatic

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_CommitStatic (ViSession vi);

Programs the hardware for the static data operation using the attributes you select. Before entering the committed state, most attribute values are stored in software only; these values have not yet been programmed to the hardware. Once the session is committed, the hardware is configured.

For most static operations it is not necessary to explicitly call <u>niHSDIO_CommitStatic</u> because the following functions implicitly commit:

- <u>niHSDIO_ReadStaticU32</u>
- <u>niHSDIO_WriteStaticU32</u>

Committing only programs attributes changed since previous commits.



**Note** Committing some attributes may have immediate effects seen on external instrument connectors. Voltage levels are an example of an attribute with an immediate effect when committed.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the function used to initialize the session.

# niHSDIO_reset

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_reset (ViSession vi);

Call this function to reset the session to its Initial state. All channels and front panel terminals are put into a high-impedance state. All software attributes are reset to their initial values.

During a reset, routes of signals between this and other devices are released, regardless of which device created the route. For instance, a trigger signal being exported to a PXI trigger line and used by another device will no longer be exported.

<u>niHSDIO_reset</u> is applied to the *entire* device. If you have both a generation and an acquisition session active, niHSDIO_reset resets the current session, including attributes, and invalidates the other session if it is committed or running. The other session must be closed.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_ResetDevice

## **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_ResetDevice (ViSession vi);

Call this function to reset the device to its Initial state and reload its FPGA. All channels and front panel terminals are put into a highimpedance state. All software attributes are reset to their initial values. The entire contents of the FPGA and EEPROM files are reloaded. Use this function to re-enable your device if it has disabled itself because the device temperature has risen above its optimal operating temperature.

During a device reset, routes of signals between this and other devices are released, regardless of which device created the route. For instance, a trigger signal being exported to a PXI trigger line and used by another device will no longer be exported.

<u>niHSDIO_ResetDevice</u> is applied to the *entire* device. If you have both a generation and an acquisition session active, niHSDIO_ResetDevice resets the current session, including attributes, and invalidates the other session if it is committed or running. The other session must be closed.

Generally, calling <u>niHSDIO_reset</u> is acceptable instead of calling <u>niHSDIO_ResetDevice</u>. <u>niHSDIO_reset</u> executes more quickly.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_ClearError

## **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_ClearError (ViSession vi);

Clears the error information for the current execution thread and the IVI session you specify. If you pass VI_NULL for **vi**, this function clears the error information only for the current execution thread.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

# niHSDIO_error_message

### **Specific Function**

### **C** Function Prototype

ViStatus niHSDIO_error_message (ViSession vi, ViStatus errorCode, ViChar[] errorMessage);

Takes the error code returned by NI-HSDIO functions, interprets it, and returns it as a user readable string.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
errorCode	ViStatus	The error code returned from the device.
errorMessage	ViChar[]	Message string
		Note The string must contain at least 256 characters.

# niHSDIO_GetError

### **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_GetError (ViSession vi, ViStatus* errorCode, ViInt32 errorDescriptionBufferSize, ViChar[] errorDescription);

Returns the error information associated with the instrument handle. This function retrieves and then clears the error information for the session. If **vi** is VI_NULL, this function retrieves and then clears the error information for the current thread.
Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
errorCode	ViStatus	Returns the error code for the session or execution thread.
errorDescriptionBufferSize	Vilnt32	Passes the number of bytes in the ViChar array you specify for the <b>errorDescription</b> parameter.
		If the error description, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies BufferSize - 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the buffer size is 4, the function places "123" into the buffer and returns 7.
		If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.
errorDescription	ViChar[]	Returns the error description

for the IVI session or execution thread.

If there is no description, the function returns an empty string. The buffer must contain at least as many elements as the value you specify with the buffer size parameter.

If you pass 0 for the **errorDescriptionBufferSize**, you can pass VI_NULL for this parameter.

# niHSDIO_IsDone

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_IsDone (ViSession vi, ViBoolean* done);

Call this function to check the hardware to determine if your dynamic data operation has completed. You can also use this function for continuous dynamic data operations to poll for error conditions.

Name Type Description

vi ViSession This handle identifies your instrument session. vi was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.

done ViBoolean Returns the status of your data operation.

#### **Defined Values**

VI_TRUE (1)—Indicates that the data operation is complete or an error has occurred. VI_FALSE (0)—Indicates that the data operation has not completed.

# niHSDIO_LockSession

# **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_LockSession (ViSession vi, ViBoolean* callerHasLock);

This function obtains the multithreaded lock on the instrument session. Before doing so, the function waits until all other execution threads have released the lock on the instrument session. Other threads might have obtained the lock on this session in the following ways:

- Your application called niHSDIO_LockSession
- A call to the instrument driver locked the session
- A call to the IVI engine locked the session

After the call to <u>niHSDIO_LockSession</u> returns successfully, no other threads can access the instrument session until you call <u>niHSDIO_UnlockSession</u>. Use <u>niHSDIO_LockSession</u> and <u>niHSDIO_UnlockSession</u> around a sequence of calls to instrument driver functions if you require exclusive access through the end of the sequence.

You can safely make nested calls to <u>niHSDIO_LockSession</u> within the same thread. To completely unlock the session, you must balance each call to <u>niHSDIO_LockSession</u> with a call to <u>niHSDIO_UnlockSession</u>. If, however, you use the callerHasLock parameter in all calls to <u>niHSDIO_LockSession</u> and <u>niHSDIO_UnlockSession</u> within a function, the IVI Library locks the session only once within the function, regardless of the number of calls you make to <u>niHSDIO_LockSession</u>. This functionality allows you to call <u>niHSDIO_UnlockSession</u> just once at the end of the function.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
callerHasLock	ViBoolean	This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL. You can use this parameter in complex functions to track lock status and the need to unlock the session. Pass the address of a local ViBoolean variable in the declaration of the local variable and initialize it to VI_FALSE. Also, pass the address of the same local variable to any other calls you make to <u>niHSDIO_LockSession</u> or <u>niHSDIO_UnlockSession</u> in the same function.

# niHSDIO_UnlockSession

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_UnlockSession (ViSession vi, ViBoolean* callerHasLock);

This function releases a lock that you acquired on an instrument session using <u>niHSDIO_LockSession</u>.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
callerHasLock	ViBoolean	This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL. You can use this parameter in complex functions to track lock status and the need to unlock the session. Pass the address of a local ViBoolean variable in the declaration of the local variable and initialize it to VI_FALSE. Also, pass the address of the same local variable to any other calls you make to <u>niHSDIO_LockSession</u> or <u>niHSDIO_UnlockSession</u> in the same function.

# niHSDIO_self_test

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_self_test (ViSession vi, ViInt16* selfTestResult, ViChar[] selfTestMessage);

This function performs a self-test on the device and returns the test results. The self-test function performs a simple series of tests that ensure the device is powered up and responding. Complete functional testing and calibration are not performed by this function.

This function is internal and does not affect external I/O connections or connections between devices.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
selfTestResult	Vilnt16	This control contains the value returned from the device self-test.
		Self-test Code Description: 0—Self-test passed Anything else—Self-test failed
selfTestMessage	ViChar[ ]	Returns the self-test response string from the device; you must pass a ViChar array at least IVI_MAX_MESSAGE_BUF_SIZE bytes in length
<u>Return Value</u>		

# niHSDIO_GetAttributeViBoolean

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_GetAttributeViBoolean (ViSession vi, ViConstString channelList, ViAttr attributeID, ViBoolean* value);

This function queries the value of a ViBoolean attribute. You can use this function to get the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance Based, pass VI_NULL or an empty string.
attributeID	ViAttr	The ID of an attribute.
value	ViBoolean	Returns the current value of the attribute; pass the address of a ViBoolean variable.

# niHSDIO_GetAttributeViInt32

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_GetAttributeViInt32 (ViSession vi, ViConstString channelList, ViAttr attributeID, ViInt32* value);

This function queries the value of a ViInt32 attribute. You can use this function to get the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
attributeID	ViAttr	The ID of an attribute.
value	Vilnt32	Returns the current value of the attribute; pass the address of a ViInt32 variable.

# niHSDIO_GetAttributeViReal64

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_GetAttributeViReal64 (ViSession vi, ViConstString channelList, ViAttr attributeID, ViReal64* value);

This function queries the value of a ViReal64 attribute. You can use this function to get the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
attributeID	ViAttr	The ID of an attribute.
value	ViReal64	Returns the current value of the attribute; pass the address of a ViReal64 variable.

# niHSDIO_GetAttributeViSession

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_GetAttributeViSession (ViSession vi, ViConstString channelList, ViAttr attributeID, ViSession* value);

This function queries the value of a ViSession attribute. You can use this function to get the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
attributeID	ViAttr	The ID of an attribute.
value	ViSession	Returns the current value of the attribute; pass the address of a ViSession variable.

# niHSDIO_GetAttributeViString

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_GetAttributeViString (ViSession vi, ViConstString channelList, ViAttr attributeID, ViInt32 bufSize, ViChar[] value);

This function queries the value of a ViString attribute. You can use this function to get the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
attributeID	ViAttr	The ID of an attribute.
bufSize	Vilnt32	Pass the number of bytes in the ViChar array you specify for the value parameter.
		If the current value of the attribute, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies Array Size-1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the array size you must pass to get the entire value. For example, if the value is "123456", and the Array Size is 4, the function places "123" into the buffer and returns 7.
		If you pass 0, you can pass VI_NULL for the value buffer parameter.
value	ViChar[ ]	Returns the current value of the attribute; pass the address of a ViChar array.
Return Value		

# niHSDIO_SetAttributeViBoolean

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_SetAttributeViBoolean (ViSession vi, ViConstString channelList, ViAttr attributeID, ViBoolean value);

This function sets the value of a ViBoolean attribute. This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
		You can pass in multiple channels to this function.
attributeID	ViAttr	The ID of an attribute.
value	ViBoolean	The value to which you want to set the attribute; some of the values might not be valid depending on the current settings of the instrument session.
<u>Return Value</u>		

# niHSDIO_SetAttributeViInt32

## **Specific Function**

# **C** Function Prototype

ViStatus niHSDIO_SetAttributeViInt32 (ViSession vi, ViConstString channelList, ViAttr attributeID, ViInt32 value);

This function sets the value of a ViInt32 attribute. This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes.

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
		You can pass in multiple channels to this function.
attributeID	ViAttr	The ID of an attribute.
value	Vilnt32	The value to which you want to set the attribute; some of the values might not be valid depending on the current settings of the instrument session.
<u>Return Value</u>		

# niHSDIO_SetAttributeViReal64

## **Specific Function**

## **C** Function Prototype

ViStatus niHSDIO_SetAttributeViReal64 (ViSession vi, ViConstString channelList, ViAttr attributeID, ViReal64 value);
#### Purpose

This function sets the value of a ViReal64 attribute. This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes.

#### **Parameters**

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
		You can pass in multiple channels to this function.
attributeID	ViAttr	The ID of an attribute.
value	ViReal64	The value to which you want to set the attribute; some of the values might not be valid depending on the current settings of the instrument session.
Return Value		

# niHSDIO_SetAttributeViSession

#### **Specific Function**

#### **C** Function Prototype

ViStatus niHSDIO_SetAttributeViSession (ViSession vi, ViConstString channelList, ViAttr attributeID, ViSession value);

#### Purpose

This function sets the value of a ViSession attribute. This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes.

#### **Parameters**

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
		You can pass in multiple channels to this function.
attributeID	ViAttr	The ID of an attribute.
value	ViSession	The value to which you want to set the attribute; some of the values might not be valid depending on the current settings of the instrument session.
Return Value		

# niHSDIO_SetAttributeViString

#### **Specific Function**

#### **C** Function Prototype

ViStatus niHSDIO_SetAttributeViString (ViSession vi, ViConstString channelList, ViAttr attributeID, ViConstString value);

#### Purpose

This function sets the value of a ViString attribute. This is a low-level function that you can use to set the values of device-specific attributes and inherent IVI attributes.

#### **Parameters**

Name	Туре	Description
vi	ViSession	This handle identifies your instrument session. <b>vi</b> was obtained from the <u>niHSDIO_InitAcquisitionSession</u> Or <u>niHSDIO_InitGenerationSession</u> function.
channelList	ViConstString	If the attribute is channel or instance based, this parameter specifies the name of the channel or instance on which to set the value of the attribute; if the attribute is not channel or instance based, pass VI_NULL or an empty string.
		You can pass in multiple channels to this function.
attributeID	ViAttr	The ID of an attribute.
value	ViConstString	The value to which you want to set the attribute; some of the values might not be valid depending on the current settings of the instrument session.
Return Value		

## **NI-HSDIO** Attributes

#### **Group/Attribute Name**

**Dynamic Channels** 

**Static Channels** 

#### **Voltage Levels**

Data High

Data Low

Trigger High

Trigger Low

Event High

Event Low

#### **Dynamic Acquisition**

Samples Per Record Number Of Records To Acquire Input Impedance Data Interpretation **Fetch** Fetch Relative To Fetch Offset Fetch Backlog Records Done **Dynamic Generation** Initial State Idle State Drive Type

Repeat Mode

Repeat Count

**Generation Mode** 

NIHSDIO_ATTR_DYNAMIC_CF NIHSDIO_ATTR_STATIC_CHAI

NIHSDIO_ATTR_DATA_VOLTA NIHSDIO_ATTR_DATA_VOLTA NIHSDIO_ATTR_TRIGGER_VC NIHSDIO_ATTR_TRIGGER_VC NIHSDIO_ATTR_EVENT_VOLT NIHSDIO_ATTR_EVENT_VOLT

NIHSDIO_ATTR_SAMPLES_PE NIHSDIO_ATTR_NUM_RECOF NIHSDIO_ATTR_INPUT_IMPEI NIHSDIO_ATTR_DATA_INTERI

NIHSDIO_ATTR_FETCH_RELA NIHSDIO_ATTR_FETCH_OFFS NIHSDIO_ATTR_FETCH_BACH NIHSDIO_ATTR_RECORDS_D

NIHSDIO_ATTR_INITIAL_STAT NIHSDIO_ATTR_IDLE_STATE NIHSDIO_ATTR_DRIVE_TYPE NIHSDIO_ATTR_REPEAT_MOI NIHSDIO_ATTR_REPEAT_COI NIHSDIO_ATTR_GENERATION

Waveform To Generate Script To Generate Timing Sample Clock Rate Source Impedance Exported Sample Clock Output Terminal NIHSDIO ATTR EXPORTED 5 Exported Sample Clock Mode Exported Sample Clock Delay **Ref Clock** Rate Source Impedance **Export Output Terminal Onboard Ref Clock Export Output Terminal Data Position** Position Delay Advanced **Oscillator Phase DAC Value** Exported Sample Clock Offset Triggers **Start Trigger** Type **Digital Edge Source Digital Edge Edge** Position

NIHSDIO ATTR WAVEFORM NIHSDIO ATTR SCRIPT TO

NIHSDIO ATTR SAMPLE CL( NIHSDIO ATTR SAMPLE CL( NIHSDIO ATTR SAMPLE CL( NIHSDIO_ATTR_EXPORTED_S NIHSDIO ATTR EXPORTED

NIHSDIO ATTR REF CLOCK NIHSDIO ATTR REF CLOCK NIHSDIO ATTR REF CLOCK NIHSDIO ATTR EXPORTED I

NIHSDIO ATTR EXPORTED (

NIHSDIO ATTR DATA POSITI NIHSDIO ATTR DATA POSITI

NIHSDIO ATTR OSCILLATOR NIHSDIO ATTR EXPORTED S

NIHSDIO ATTR START TRIG NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO ATTR DIGITAL EDG NIHSDIO ATTR DIGITAL EDG

**Digital Edge Terminal Configuration Digital Edge Impedance** Pattern Match Pattern Pattern Match Trigger When **Export Output Terminal Export Terminal Configuration Ref Trigger Ref Trigger Type** Pretrigger Samples Per Record **Digital Edge Source Digital Edge Edge** Position **Digital Edge Impedance Digital Edge Terminal Configuration** Pattern Match Pattern Pattern Match Trigger When **Export Output Terminal Export Terminal Configuration** Start to Reference Trigger Holdoff Reference to Reference Trigger Holdoff **Advance Trigger** 

Type Digital Edge Source Digital Edge Edge Digital Edge Position Digital Edge Terminal Configuration Digital Edge Impedance Pattern Match Pattern Pattern Match Trigger When NIHSDIO_ATTR_DIGITAL_EDG NIHSDIO_ATTR_DIGITAL_EDG NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_EXPORTED_ NIHSDIO_ATTR_EXPORTED_

NIHSDIO_ATTR_REF_TRIGGE NIHSDIO_ATTR_REF_TRIGGE NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_EXPORTED_I NIHSDIO_ATTR_EXPORTED_I NIHSDIO_ATTR_START_TO_R NIHSDIO_ATTR_REF_TO_REF

NIHSDIO_ATTR_ADVANCE_TF NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_PATTERN_M/ **Export Output Terminal Export Terminal Configuration Script Trigger** Type **Digital Edge Source Digital Edge Edge Digital Edge Terminal Configuration Digital Edge Impedance Digital Level Source Digital Level Level Digital Level Terminal Configuration Digital Level Impedance Export Output Terminal Export Terminal Configuration Pause Trigger** Type **Digital Level Source Digital Level Level Digital Level Position Digital Level Terminal Configuration Digital Level Impedance** Pattern Match Pattern Pattern Match Trigger When **Export Output Terminal Export Terminal Configuration Events Ready For Start Event** 

Output Terminal Active Level NIHSDIO_ATTR_EXPORTED_/ NIHSDIO_ATTR_EXPORTED_/

NIHSDIO_ATTR_SCRIPT_TRIC NIHSDIO_ATTR_SCRIPT_TRIC NIHSDIO_ATTR_SCRIPT_TRIC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_EDC NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV

NIHSDIO_ATTR_PAUSE_TRIG NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_DIGITAL_LEV NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_PATTERN_M/ NIHSDIO_ATTR_EXPORTED_I NIHSDIO_ATTR_EXPORTED_I

NIHSDIO_ATTR_READY_FOR NIHSDIO_ATTR_READY_FOR

**Terminal Configuration Ready For Advance Event Output Terminal** Active Level **Terminal Configuration** End Of Record Event **Output Terminal Pulse Polarity Terminal Configuration Data Active Event Output Terminal** Active Level Position **Terminal Configuration Marker Event Output Terminal Pulse Polarity** Position **Terminal Configuration Device Characteristics** Total Acquisition Memory Size **Total Generation Memory Size** Serial Number Advanced Data Width Data Rate Multiplier Data Active Internal Route Delay

NIHSDIO_ATTR_READY_FOR_

NIHSDIO_ATTR_READY_FOR NIHSDIO_ATTR_READY_FOR NIHSDIO_ATTR_READY_FOR

NIHSDIO_ATTR_END_OF_RE( NIHSDIO_ATTR_END_OF_RE( NIHSDIO_ATTR_END_OF_RE(

NIHSDIO_ATTR_DATA_ACTIVE NIHSDIO_ATTR_DATA_ACTIVE NIHSDIO_ATTR_DATA_ACTIVE NIHSDIO_ATTR_DATA_ACTIVE

NIHSDIO_ATTR_MARKER_EV NIHSDIO_ATTR_MARKER_EV NIHSDIO_ATTR_MARKER_EV NIHSDIO_ATTR_MARKER_EV

NIHSDIO_ATTR_TOTAL_ACQL NIHSDIO_ATTR_TOTAL_GENE NIHSDIO_ATTR_SERIAL_NUM

NIHSDIO_ATTR_DATA_WIDTH NIHSDIO_ATTR_DATA_RATE_ NIHSDIO_ATTR_DATA_ACTIVE

# NIHSDIO_ATTR_DYNAMIC_CHANNELS

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_AssignDynamicChannels

Assigns channels for dynamic operation. The group of dynamic channels is changed each time this attribute is set. For example, setting this attribute to 0-10 and then setting it to 5-8 results in only channels 5-8 being assigned to dynamic. Channels 0-4 and 9-10 are no longer assigned to the dynamic operation by the second configuration.

Setting an empty string to this attribute configures *all* channels for dynamic operation. Writing None removes all channels from the dynamic operation. The session must be committed before changes to this attribute will take affect (refer to <u>niHSDIO_CommitDynamic</u> for more information about committing a session).

You can configure a channel for more than one simultaneous operation. A channel can be simultaneously configured for the following operations:

- Dynamic generation and any (static and/or dynamic) acquisition
- Static generation and any (static and/or dynamic) acquisition
- Both static and dynamic acquisition



Unconfiguring a dynamic generation channel frees that channel to be reconfigured for static generation. Unconfiguring a dynamic generation channel does not stop the channel from driving its current valueany value already written to the channel continues to be driven.

#### Syntax examples:

- 2-15 or 15-2 set channels 2 through 15 to dynamic
- 0, 3, 10 or 3, 10, 0 set channels 0, 3, and 10 to dynamic
- "" (empty string) sets all channels for dynamic
- None removes all channels from dynamic

# NIHSDIO_ATTR_STATIC_CHANNELS

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_AssignStaticChannels</u>

Assigns channels for static operation. The group of static channels is changed each time this attribute is set. For example, setting this attribute to 0-10 and then setting it to 5-8 results in only channels 5-8 being assigned to static. Channels 0-4 and 9-10 are unconfigured by the second configuration.

Writing an empty string to this attribute configures *all* channels for static operation. Writing the value None removes all channels from static operation. The channel is not changed until a call to <u>niHSDIO_ReadStaticU32</u> (for acquisition sessions) or <u>niHSDIO_WriteStaticU32</u> (for generation sessions).

You can configure a channel for more than one simultaneous operation. A channel can be simultaneously configured for the following operations:

- Dynamic generation and any (static and/or dynamic) acquisition
- Static generation and any (static and/or dynamic) acquisition
- Both static and dynamic acquisition



Unconfiguring a static generation channel frees that channel to be reconfigured for dynamic generation. Unconfiguring a static generation channel does not stop the channel from driving its current value any static value already written to the channel continues to be driven.

#### Syntax examples:

- 2-15 or 15-2 set channels 2 through 15 to static
- 0, 3, 10 or 3, 10, 0 set channels 0, 3, and 10 to static
- "" (empty string) sets all channels for static
- None removes all channels from static operation.

# NIHSDIO_ATTR_DATA_VOLTAGE_HIGH_LEVEL

Data type	Access	Applies to	Coercion	High-Lev	el Functions
ViReal64	R/W	Channel	None	niHSDIO_	<u>ConfigureDataVoltageLog</u>
				<u>niHSDIO</u>	ConfigureDataVoltageCust

This attribute sets the high data <u>voltage level</u> for the session. For an acquisition session, this sets the Acquisition Voltage High Level. For a generation session, this sets the Generation Voltage High Level.

This property applies to static and dynamic data operations.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this attribute when programming those devices.

# NIHSDIO_ATTR_DATA_VOLTAGE_LOW_LEVEL

Data type	Access	Applies to	Coercion	High-Lev	el Functions
ViReal64	R/W	Channel	None	niHSDIO_	<u>ConfigureDataVoltageLog</u>
				<u>niHSDIO</u>	ConfigureDataVoltageCust

This attribute sets the low data <u>voltage level</u> for the session. For an acquisition session, this sets the Acquisition Voltage Low Level. For a generation session, this sets the Generation Voltage Low Level.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this attribute when programming those devices.

# NIHSDIO_ATTR_EVENT_VOLTAGE_HIGH_LEVEL

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	N/A	None	niHSDIO_ConfigureEventVoltageLog
				niHSDIO ConfigureEventVoltageCus

This attribute sets the high event voltage level for the session. If you do not explicitly set this attribute, NI-HSDIO assumes the same value as <u>NIHSDIO_ATTR_DATA_VOLTAGE_HIGH_LEVEL</u> for the generation session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this attribute when programming those devices.

# NIHSDIO_ATTR_EVENT_VOLTAGE_LOW_LEVEL

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	N/A	None	niHSDIO_ConfigureEventVoltageLog
				niHSDIO ConfigureEventVoltageCus

This attribute sets the low event <u>voltage level</u> for the session. If you do not explicitly set this attribute, NI-HSDIO assumes the same value as <u>NIHSDIO_ATTR_DATA_VOLTAGE_LOW_LEVEL</u> for the generation session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this attribute when programming those devices.

# NIHSDIO_ATTR_TRIGGER_VOLTAGE_LOW_LEV

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	N/A	None	niHSDIO_ConfigureTriggerVoltageL niHSDIO_ConfigureTriggerVoltageC

This attribute sets the low trigger voltage level for the session. If you do not explicitly set this attribute, NI-HSDIO assumes the same value as <u>NIHSDIO_ATTR_DATA_VOLTAGE_LOW_LEVEL</u> for the acquisition session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this attribute when programming those devices.

# NIHSDIO_ATTR_TRIGGER_VOLTAGE_HIGH_LEV

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	N/A	None	niHSDIO_ConfigureTriggerVoltageL
				niHSDIO ConfigureTriggerVoltageC

This attribute sets the high trigger voltage level for the session. If you do not explicitly set this attribute, NI-HSDIO assumes the same value as <u>NIHSDIO_ATTR_DATA_VOLTAGE_HIGH_LEVEL</u> for the acquisition session.



**Note** NI 656*x* devices do *not* support configuring voltage levels. NI-HSDIO returns an error if you use this attribute when programming those devices.

# NIHSDIO_ATTR_SAMPLES_PER_RECORD

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies the number of samples to be acquired per record. If you are using a Reference trigger, this attribute includes both pretrigger and posttrigger samples.

This attribute is valid only for acquisition sessions.

# NIHSDIO_ATTR_INPUT_IMPEDANCE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W Channel None None

Use this attribute to change input impedance for the data channels. Refer to the device documentation for more information about the input impedance.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI	High impedance
654 <i>x</i>	
	Refer to the <u>specifications</u> document for your device for more information on the supported high-impedance values.
NI 655 <i>x</i>	50 or high impedance
	Refer to the <u>specifications</u> document for your device for more information on the supported high-impedance values.
NI 656 <i>x</i>	100 in <u>LVDS</u> terminal configuration 10,000 in single-ended terminal configuration

Refer to the <u>Termination</u> section for acquisition with your device for more information about choosing the input impedance.

Units: ohms

# NIHSDIO_ATTR_DATA_INTERPRETATION

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	Channel	None	niHSDIO_ConfigureDataInterpretation

Use this attribute to select between acquiring using the high/low data or valid/invalid data mode of <u>data interpretation</u> during a static or dynamic acquisition operation.

Select high/low mode to get logic high or logic low values. Use valid/invalid mode to tell if the signal is within the specified voltage range (above data voltage low level but below data voltage high level) or outside the range (below data voltage low level or above data voltage high level).



**Note** NI 654*x*/656*x* devices only support the high/low mode of data interpretation. NI-HSDIO returns an error if you select valid/invalid mode for an acquisition with these devices.

Refer to your specific hardware documentation to understand how data is returned to you. This attribute is only valid for acquisition sessions.

### **Defined Values:**

NIHSDIO_VAL_HIGH_OR_LOW (3)	Data read represents logical values (high level or low level).
NIHSDIO_VAL_VALID_OR_INVALID (4)	Data read represents the channel data state: tristate, high level, or low level.
# NIHSDIO_ATTR_FETCH_BACKLOG

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 RO N/A None None

Use this attribute to query how many acquired samples remain in onboard memory. This attribute is used with

<u>NIHSDIO_ATTR_FETCH_OFFSET</u> and <u>NIHSDIO_FETCH_RELATIVE_TO</u>. This attribute returns the number of samples available from the specified relativeTo and offset.

This attribute is valid only for acquisition sessions.

# NIHSDIO_ATTR_FETCH_RELATIVE_TO

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViInt32 R/W N/A None None

Specifies the absolute location within the <u>acquired record</u> from which to begin fetching. The default value is

NIHSDIO_VAL_CURRENT_READ_POSITION. If the <u>Reference trigger</u> is enabled, the read position is initially set to the first pretrigger sample. If the Reference trigger is disabled, the read position is initially set to the first sample acquired. After every fetch, the read position moves to the sample immediately after the last fetched sample. Thus, if you call fetch multiple times and <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> is set to NIHSDIO_VAL_CURRENT_READ_POSITION, each fetch retrieves a different part of the record. If <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> is set to any other value, you must modify

<u>NIHSDIO_ATTR_FETCH_OFFSET</u> between fetches to fetch different parts of the record.

### **Defined Values:**

NIHSDIO_VAL_MOST_RECENT_SAMPLE (46)	Specifies that fetchir relative to the most r acquired data. The function ( <u>NIHSDIO_ATTR_FF</u> must be negative.
NIHSDIO_VAL_FIRST_SAMPLE (47)	Specifies that fetchir the first sample acqu device. If the device buffer, then the first s longer available. In t HSDIO returns an er offset is in the overw
NIHSDIO_VAL_REFERENCE_TRIGGER (48)	Specifies that fetchir relative to the Refere This value behaves   NIHSDIO_VAL_FIRS no Reference trigger
NIHSDIO_VAL_FIRST_PRETRIGGER_SAMPLE (49)	Specifies that fetchir relative to the first pr sample acquired. Th behaves like NIHSDIO_VAL_FIRS no Reference trigger
NIHSDIO_VAL_CURRENT_READ_POSITION (50)	Specifies that fetchir the last fetched sam

# NIHSDIO_ATTR_FETCH_OFFSET

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViInt32 R/W N/A None None

Offset in samples to start fetching acquired waveform data. The offset is applied relative to the <u>NIHSDIO_ATTR_FETCH_RELATIVE_TO</u> position. Offset can be a positive or negative value.

If the specified offset would cause the fetch to exceed the end of the waveform, NI-HSDIO returns a data overwrite error. If the selected offset would cause the fetch location to occur before the start of the waveform, the fetch location is coerced to the beginning of the waveform.

This attribute is valid only for acquisition sessions.

# NIHSDIO_ATTR_RECORDS_DONE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

Vilnt32 RO N/A None None

Returns the number of records that have been acquired.

## NIHSDIO_ATTR_NUM_RECORDS

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies the number of records you want to acquire.

# NIHSDIO_ATTR_INITIAL_STATE

#### **Specific Attribute**

# Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W Channel

None

niHSDIO_ConfigureInitialState niHSDIO_ConfigureInitialStateU32

Specifies the channel state after the session is initiated but before the first waveform sample is generated. The channel changes to the <u>Initial</u> <u>state</u> once the data operation has been initiated. When the <u>Start trigger</u> is asserted, the Initial state is replaced by the first sample in the waveform.

This attribute is valid only for generation sessions.

Note NI 656x devices do not support the tristate Initial state.

### **Defined Values:**

NIHSDIO_VAL_TRISTATE (24)	Sets the channel to a high- impedance state.
NIHSDIO_VAL_LOGIC_HIGH (1)	Sets the channel to a logic- high (high level) state.
NIHSDIO_VAL_LOGIC_LOW (0)	Sets the channel to a logic- low (low level) state.
NIHSDIO_VAL_HOLD_LAST_VALUE (27)	The channel retains its previous value.

# NIHSDIO_ATTR_IDLE_STATE

#### **Specific Attribute**

# Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W Channel

el None

niHSDIO_ConfigureIdleState niHSDIO_ConfigureIdleStateU32

Specifies the <u>Idle state</u> for a channel.

A dynamic generation operation may be idle when the operation completes normally, when the operation pauses, or when it terminates because of an underflow error.

This attribute is valid only for generation sessions.



**Note** NI 656*x* devices do not support the tristate Idle state.

### **Defined Values:**

NIHSDIO_VAL_TRISTATE (24)	Sets the channel to a high- impedance state.
NIHSDIO_VAL_LOGIC_HIGH (1)	Sets the channel to a logic- high (high level) state.
NIHSDIO_VAL_LOGIC_LOW (0)	Sets the channel to a logic- low (low level) state.
NIHSDIO_VAL_HOLD_LAST_VALUE (27)	The channel retains its previous value.

# NIHSDIO_ATTR_DRIVE_TYPE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W Channel None N/A

Specifies what the data channels generate when set to logic 1. Using the <u>open collector</u> setting to generate a Z is useful for wired logic buses, such as I²C or SMBus.



**Notes** <u>NI 656x</u> devices only support the active drive setting. NI-HSDIO returns an error if you try to configure the channels on these devices for open collector generation.

<u>NI 654x</u> devices support open collector generation only for <u>static</u> <u>generation</u>. NI-HSDIO returns an error if you try to configure the channels on these devices for open collector dynamic generation.

This property is only valid for generation sessions.

### **Defined Values:**

Active drive (75)	The Generation Voltage High Level for the device is produced at the channel electronics when the Pattern Generation Engine generates a binary 1.
<b>Open</b> collector (76)	The channel electronics assume a high-impedance state when the Pattern Generation Engine generates a binary 1.

### Remarks

The following table lists the characteristics of this property.

Data Type	Vilnt32
Permissions	R/W
Channel Based	Yes

# NIHSDIO_ATTR_REPEAT_MODE

### Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureGenerationRepeat

Use this attribute to specify whether or not to generate a single waveform continuously. This attribute is valid only when

NIHSDIO_ATTR_GENERATION_MODE is set to

NIHSDIO_VAL_WAVEFORM; it is not used when in scripted mode. If this attribute is set to NIHSDIO_VAL_FINITE, then use

<u>NIHSDIO_ATTR_REPEAT_COUNT</u> to specify how many times the named waveform is generated.

This attribute is valid only for generation sessions.

### **Defined Values:**

NIHSDIO_VAL_FINITE (16)	Calling <u>niHSDIO_Initiate</u> generates the named waveform a finite number of times. The number to repeat is defined by <u>NIHSDIO_ATTR_REPEAT_COUNT</u> .
NIHSDIO_VAL_CONTINUOUS (17)	Calling <u>niHSDIO_InitGenerationSession</u> generates the named waveform continuously (until <u>niHSDIO_Abort</u> is called).

# NIHSDIO_ATTR_REPEAT_COUNT

### Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureGenerationRepeat

Use this attribute to specify how many times to generate the waveform specified by <u>NIHSDIO_ATTR_WAVEFORM_TO_GENERATE</u>. This attribute is valid only when <u>NIHSDIO_ATTR_REPEAT_MODE</u> is set to NIHSDIO_VAL_FINITE; it is not used when the device is in continuous mode. This attribute is valid only when <u>NIHSDIO_ATTR_GENERATION_MODE</u> is set to <u>NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NIHSDIO_NI</u>

NIHSDIO_VAL_WAVEFORM; it is ignored when in scripted mode.

This attribute is only valid for generation sessions.

# NIHSDIO_ATTR_GENERATION_MODE

### Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureGenerationMode

Use this attribute to specify whether to generate the waveform specified by <u>NIHSDIO_ATTR_WAVEFORM_TO_GENERATE</u> or the script specified by <u>NIHSDIO_ATTR_SCRIPT_TO_GENERATE</u> upon calling <u>niHSDIO_Initiate</u>.

This attribute is valid only for generation sessions.

### **Defined Values:**

NIHSDIO_VAL_WAVEFORM (14)	Calling <u>niHSDIO_Initiate</u> generates the waveform represented by <u>NIHSDIO_ATTR_WAVEFORM_TO_GEN</u>
NIHSDIO_VAL_SCRIPTED (15)	Calling niHSDIO_Initiate generates the represented by <u>NIHSDIO_ATTR_SCRIPT_TO_GENERA</u>

# NIHSDIO_ATTR_WAVEFORM_TO_GENERATE

### Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigureWaveformToGene

Use this attribute to specify which named waveform in <u>onboard memory</u> is generated upon calling <u>niHSDIO_Initiate</u> when

NIHSDIO_ATTR_GENERATION_MODE is NIHSDIO_VAL_WAVEFORM. If this attribute is not set to a valid waveform name and more than one waveform is in onboard memory, you receive an error when calling <u>niHSDIO_Initiate</u>. If only one waveform is in onboard memory and this attribute is set to empty string, then that waveform is generated upon calling <u>niHSDIO_Initiate</u>.

This attribute is ignored when <u>NIHSDIO_ATTR_GENERATION_MODE</u> is set to NIHSDIO_VAL_SCRIPTED, since

<u>NIHSDIO_ATTR_SCRIPT_TO_GENERATE</u> defines the sequence of waveforms to generate.

This attribute is valid only for generation sessions.

# NIHSDIO_ATTR_SCRIPT_TO_GENERATE

### Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigureScriptToGenerate

Use this attribute to specify which <u>script</u> in <u>onboard memory</u> is generated upon calling <u>niHSDIO_Initiate</u> when

<u>NIHSDIO_ATTR_GENERATION_MODE</u> is NIHSDIO_VAL_SCRIPTED. If this attribute is not set to a valid script and more than one script is in onboard memory, you receive an error upon calling <u>niHSDIO_Initiate</u>. If only one script is in onboard memory and this attribute is set to empty string, you receive an error upon calling <u>niHSDIO_Initiate</u>.

This attribute is ignored when <u>NIHSDIO_ATTR_GENERATION_MODE</u> is set to NIHSDIO_VAL_WAVEFORM, since

<u>NIHSDIO_ATTR_WAVEFORM_TO_GENERATE</u> defines which waveform to generate.

This attribute is valid only for generation sessions.

# NIHSDIO_ATTR_STREAMING_ENABLED

Data type Access Applies to Coercion High-Level Functions

ViBoolean R/W N/A None None

Enables streaming of data from host memory to the device. This property is valid only for dynamic generation sessions.

## NIHSDIO_ATTR_STREAMING_WAVEFORM_NAM

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None None
Specifies the name of the waveform for streaming. Use this attribute in conjunction with the <u>NIHSDIO_ATTR_STREAMING_ENABLED</u> attribute.

Note You cannot stream an unnamed waveform.

# NIHSDIO_ATTR_SPACE_AVAILABLE_IN_STREAM

Data type Access Applies to Coercion High-Level Functions

Vilnt32 RO N/A None None

Specifies the space (in samples) available in the streaming waveform. This property is valid only when streaming.

# NIHSDIO_ATTR_DIRECT_DMA_ENABLED

Data type Access Applies to Coercion High-Level Functions

ViBoolean R/W N/A None None

Enables direct DMA.

This property is valid only for dynamic generation sessions.

# NIHSDIO_ATTR_DIRECT_DMA_WINDOW_SIZE

Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the direct DMA window size (in bytes).

# NIHSDIO_ATTR_DIRECT_DMA_WINDOW_ADDRE

Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the start address for the direct DMA window.

# NIHSDIO_ATTR_DATA_TRANSFER_BLOCK_SIZE

Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the number of samples to download to onboard memory at one time.

# NIHSDIO_ATTR_SAMPLE_CLOCK_RATE

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	N/A	None	niHSDIO_ConfigureSampleClock

Use this attribute to specify the <u>Sample clock</u> rate.

You must set this attribute even when you supply an external clock because NI-HSDIO uses this attribute for a number of reasons, including optimal error checking and certain pulse width selections.

If you are using the On Board Clock source, <u>getting</u> this value shows how NI-HSDIO coerced the value.

Units: hertz

# NIHSDIO_ATTR_SAMPLE_CLOCK_SOURCE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ConfigureSampleClock</u>

Use this attribute to specify the <u>Sample clock</u> source. STROBE is valid only for acquisition operations.

### **Defined Values:**

NIHSDIO_VAL_ON_BOARD_CLOCK_STR ("OnBoardClock")	The device will use the onboard oscillator.
NIHSDIO_VAL_CLK_IN_STR ("ClkIn")	The device will use the clock present at the front panel CLK IN SMB jack connector.
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.
NIHSDIO_VAL_STROBE_STR ("STROBE")	The device will use the clock present at

t ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	the STROBE channel of the DDC connector. This is valid only for acquisition
	acquisition sessions.

# NIHSDIO_ATTR_SAMPLE_CLOCK_IMPEDANCE

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None None

Use this attribute to program the input impedance of the CLK IN SMB jack connector when the <u>Sample clock</u> is supplied through the front panel.

Units: ohms

# NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_C

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Use this attribute to export the <u>Sample clock</u> to the specified terminal.

### **Defined Values:**

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_CLK_OUT_STR ("ClkOut")	The devices will use the signal present at the front panel CLK OUT SMB jack connector.
NIHSDIO_VAL_DDC_CLK_OUT_STR ("DDC_ClkOut")	DDC CLK OUT channel in the front panel DDC connector.

# NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_N

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Use this attribute to specify the position of the exported <u>Sample clock</u> relative to the Sample clock used by the device. When the Sample clock rate is set to less than 25 MS/s, this attribute must not be set to NIHSDIO_VAL_DELAYED.

### **Defined Values:**

NIHSDIO_VAL_NONINVERTED (21)	The device exports the Sample cloc modifications.
NIHSDIO_VAL_INVERTED (22)	The device inverts the Sample clock
NIHSDIO_VAL_DELAYED (23)	The device delays the Sample clock Use <u>NIHSDIO_ATTR_EXPORTED_SAMI</u> to specify the delay.

# NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_C

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None None

Use this attribute to specify the delay of the exported <u>Sample clock</u> relative to the Sample clock used by the device. This attribute is relevant only when <u>NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_MODE</u> is set to NIHSDIO_VAL_DELAYED. Otherwise, this attribute is ignored. This attribute is specified as a fraction of the Sample clock period, that is, as a fraction of (1/<u>NIHSDIO_ATTR_SAMPLE_CLOCK_RATE</u>).

# NIHSDIO_ATTR_REF_CLOCK_RATE

### **Specific Attribute**

### Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None <u>niHSDIO_ConfigureRefClock</u>

Use this attribute to specify the rate of the <u>Reference clock</u>. 10 MHz is currently the only valid value for this attribute. This attribute is ignored when <u>NIHSDIO_ATTR_REF_CLOCK_SOURCE</u> is set to None.

Units: hertz

# NIHSDIO_ATTR_REF_CLOCK_SOURCE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ConfigureRefClock</u>

Use this attribute to specify the <u>Reference clock</u> source.

## **Defined Values:**

NIHSDIO_VAL_NONE_STR ("None")	The device will not use a Reference clock.
NIHSDIO_VAL_CLK_IN_STR ("ClkIn")	The device will use the clock present at the front panel CLK IN SMB jack connector.
NIHSDIO_VAL_PXI_CLK10_STR ("PXI_CLK10")	The device will use the PXI_CLK10 signal, which is present on the PXI backplane. This selection is valid only for PXI devices.
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. This selection is valid only for PCI devices.

# NIHSDIO_ATTR_REF_CLOCK_IMPEDANCE

### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None None

Use this attribute to set the input impedance of the <u>Reference clock</u> when it is supplied through the front panel.

Valid values are 50 or 1000.

Units: ohms

# NIHSDIO_ATTR_EXPORTED_REF_CLOCK_OUTP

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Use this attribute to export the <u>Reference clock</u> to the specified terminal.
NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_CLK_OUT_STR ("ClkOut")	The devices will use the signal present at the front panel CLK OUT SMB jack connector.

# NIHSDIO_ATTR_EXPORTED_ONBOARD_REF_CI

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString R/W None None None

Use this attribute to export the <u>Onboard Reference clock</u> to the specified terminal. This attribute is only valid for PCI devices.

NIHSDIO_VAL_NONE_STR ("None")	The device will not export the Onboard Reference clock.
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7.

# NIHSDIO_ATTR_DATA_POSITION

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W Channel None <u>niHSDIO_ConfigureDataPosition</u>

Specifies the <u>data position</u> for the operation, which specifies which edge of the <u>Sample clock</u> is used to time the generation or acquisition. You can also configure the device to generate or acquire data at a configurable delay past each rising edge of the Sample clock. When this attribute is set to NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE, use <u>NIHSDIO_ATTR_DATA_POSITION_DELAY</u> to specify the delay value.



**Note** To configure a delay on NI 656*x* devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	th
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	th
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Τł
	a
	Sţ
	<u>N</u> ]
	Τł
	or
	Sá
	wł
	th

# NIHSDIO_ATTR_DATA_POSITION_DELAY

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViReal64	R/W	Channel	None	niHSDIO_ConfigureDataPositionDel

Specifies the delay after the Sample clock rising edge when the device generates or acquires a new data sample. Data delay is expressed as a fraction of the clock period (for example, a fraction of 1/<u>NIHSDIO_ATTR_SAMPLE_CLOCK_RATE</u>). This attribute is relevant only when <u>NIHSDIO_ATTR_DATA_POSITION</u> is set to NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE.



**Note** To configure a delay on NI 656*x* devices, you must delay all channels on the device. NI-HSDIO returns an error if you apply a delay to only a partial channel list.

# NIHSDIO_ATTR_EXPORTED_SAMPLE_CLOCK_C

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None None

Use this attribute to offset the exported <u>Sample clock</u> by a fixed time. Refer to <u>Dynamic Generation Timing Diagrams</u> for more information about changing this value.

Valid values for this ViReal64 are 2.5e-9 and 0 for the  $\frac{NI 654x}{655x}$  devices and 1.6e-9 for the  $\frac{NI 656x}{656x}$  devices.

Units: seconds

# NIHSDIO_ATTR_OSCILLATOR_PHASE_DAC_VAL

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Use this attribute to phase shift the PLL circuit of the onboard clock source. You can use this attribute to align the <u>Sample clock</u> of this device with another device that shares the same <u>Reference clock</u>. This attribute is valid if <u>NIHSDIO_ATTR_REF_CLOCK_SOURCE</u> source is not set to NIHSDIO_VAL_NONE_STR.

The valid range for this attribute is 0 to 4,095.

# NIHSDIO_ATTR_START_TRIGGER_TYPE

### **Specific Attribute**

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeStartTr
				niHSDIO_ConfigurePatternMatchStart1
				niUSDIO ConfigureSoftwareStartTrigg

<u>Start</u>  $\underline{niHSDIO_ConfigureSoftwareStartTrigg}$ niHSDIO_DisableStartTrigger

Use this attribute to specify whether you want the <u>Start trigger</u> to be a <u>digital edge</u>, <u>pattern match</u>, or <u>software trigger</u>. You can also choose NIHSDIO_VAL_NONE as the value for this attribute.

NIHSDIO_VAL_NONE (28)	The data operation starts immedi <u>niHSDIO_Initiate</u> .
NIHSDIO_VAL_DIGITAL_EDGE (29)	The data operation does not start The source of the digital edge is : <u>NIHSDIO_ATTR_DIGITAL_EDGF</u> and the active edge is specified v <u>NIHSDIO_ATTR_DIGITAL_EDGF</u>
NIHSDIO_VAL_SOFTWARE (32)	The data operation does not star You can assert the software trigg <u>niHSDIO_SendSoftwareEdgeTrigge</u> NIHSDIO_VAL_START_TRIGGEF
NIHSDIO_VAL_PATTERN_MATCH (31)	The data operation does not take pattern matching condition is met setting <u>NIHSDIO_ATTR_PATTERN_MAT</u> and <u>NIHSDIO_ATTR_PATTERN_MAT</u> This is valid only for acquisition s

# NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeStartT

Specifies the source terminal for the <u>Start trigger</u>. This attribute is used only when <u>NIHSDIO_ATTR_START_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7")	PXI trigger line 7. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)
NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3") NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4") NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5") NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6") NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7") NIHSDIO_VAL_RTSI0_STR ("RTSI0") NIHSDIO_VAL_RTSI1_STR ("RTSI1") NIHSDIO_VAL_RTSI2_STR ("RTSI2") NIHSDIO_VAL_RTSI3_STR ("RTSI3")	<ul> <li>PXI trigger line 3. (PXI devices)</li> <li>PXI trigger line 4. (PXI devices)</li> <li>PXI trigger line 5. (PXI devices)</li> <li>PXI trigger line 6. (PXI devices)</li> <li>PXI trigger line 7. (PXI devices)</li> <li>RTSI trigger line 0. (PCI devices)</li> <li>RTSI trigger line 1. (PCI devices)</li> <li>RTSI trigger line 2. (PCI devices)</li> <li>RTSI trigger line 3. (PCI devices)</li> </ul>

NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. (PCI devices)
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

# NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGE

Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeStartTr

Specifies the active edge for the <u>Start trigger</u>. This attribute is used only when <u>NIHSDIO_ATTR_START_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_RISING_EDGE (12)	Rising-edge trigger.
NIHSDIO_VAL_FALLING_EDGE (13)	Falling-edge trigger.

# NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the position where the <u>Start trigger</u> is asserted, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a front panel connector.

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	th
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	th
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Tł
	th
	CS
	ra
	นร
	<u>N</u> ]
	Tł
	or
	Sć
	w
	th

# NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal R/W N/A None

Specifies the impedance on the channel configured for the <u>digital edge</u> <u>Start trigger</u>.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in <u>LVDS terminal configuration</u> 10000 in single-ended <u>terminal configuration</u>

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

# NIHSDIO_ATTR_DIGITAL_EDGE_START_TRIGGE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies whether the <u>Start trigger</u> terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_PATTERN_MATCH_START_TRIG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigurePatternMatchStart

Sets the pattern match mask for the <u>Start trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_START_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_PATTERN_MATCH. The pattern is a string of characters representing the entire pattern to be matched on. Each character corresponds to a particular channel.

- 'X': Match on any value
- '1': Match on a logic 1
- '0': Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

Spaces are ignored, and are useful for readability to segment long patterns. The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are '0' and channels 2-7 are '1'. The values seen by pattern matching are affected by <u>NIHSDIO ATTR DATA INTERPRETATION</u>.

# NIHSDIO_ATTR_PATTERN_MATCH_START_TRIG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigurePatternMatchStart1

Specifies whether a <u>pattern match</u> <u>Start trigger</u> asserts when a particular pattern is matched or not matched. This attribute is valid only for acquisition tasks.

NIHSDIO_VAL_PATTERN_MATCHES (36)	The trigger asserts when the pattern matches.
NIHSDIO_VAL_PATTERN_DOES_NOT_MATCH (37)	The trigger asserts when the pattern does not match.

# NIHSDIO_ATTR_EXPORTED_START_TRIGGER_(

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>
Specifies the destination terminal for exporting the <u>Start trigger</u>. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_EXPORTED_START_TRIGGER_1

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies the terminal configuration for the exported <u>Start trigger</u> terminal. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_REF_TRIGGER_TYPE

## **Specific Attribute**

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalEdge niHSDIO_ConfigurePatternMate

niHSDIO_ConfigureDigitalEdgeRefTri niHSDIO_ConfigurePatternMatchRefTr niHSDIO_ConfigureSoftwareRefTrigge niHSDIO_DisableRefTrigger

Specifies the <u>Reference trigger</u> type. Depending on this attribute value, you may need to set more attributes to fully configure the trigger.

This attribute is valid only for acquisition sessions.

NIHSDIO_VAL_NONE (28)	The acquisition operation does not the data operation starts immedi <u>niHSDIO_Initiate</u> and after the States asserts.
NIHSDIO_VAL_DIGITAL_EDGE (29)	The Reference trigger asserts wh The source of the digital edge is a <u>NIHSDIO_ATTR_DIGITAL_EDGE</u> and the active edge is specified v <u>NIHSDIO_ATTR_DIGITAL_EDGE</u>
NIHSDIO_VAL_SOFTWARE (32)	The data operation does not star occurs. You can assert the softwa <u>niHSDIO_SendSoftwareEdgeTrigge</u> NIHSDIO_VAL_REF_TRIGGER a
NIHSDIO_VAL_PATTERN_MATCH (31)	The data operation does not take pattern matching condition is met setting <u>NIHSDIO_ATTR_PATTERN_MAT</u> and <u>NIHSDIO_ATTR_PATTERN_MAT</u> This is valid only for acquisition s

# NIHSDIO_ATTR_REF_TRIGGER_PRETRIGGER_S

## **Specific Attribute**

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeRefTri
				<pre>niHSDIO_ConfigurePatternMatchRefTr</pre>
				<pre>niHSDIO_ConfigureSoftwareRefTrigge</pre>

Specifies the number of pretrigger samples to be acquired per record (for example, the samples acquired before the <u>Reference trigger</u> is received). The number of pretrigger samples cannot be greater than <u>NIHSDIO_ATTR_SAMPLES_PER_RECORD</u>.

This attribute is valid only for acquisition sessions.

# NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeRefTr

Specifies the source terminal for the <u>Reference trigger</u>. This attribute is used only when <u>NIHSDIO_ATTR_REF_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7")	PXI trigger line 7. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)
NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3") NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4") NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5") NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6") NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7") NIHSDIO_VAL_RTSI0_STR ("RTSI0") NIHSDIO_VAL_RTSI1_STR ("RTSI1") NIHSDIO_VAL_RTSI2_STR ("RTSI2") NIHSDIO_VAL_RTSI3_STR ("RTSI3")	<ul> <li>PXI trigger line 3. (PXI devices)</li> <li>PXI trigger line 4. (PXI devices)</li> <li>PXI trigger line 5. (PXI devices)</li> <li>PXI trigger line 6. (PXI devices)</li> <li>PXI trigger line 7. (PXI devices)</li> <li>RTSI trigger line 0. (PCI devices)</li> <li>RTSI trigger line 1. (PCI devices)</li> <li>RTSI trigger line 2. (PCI devices)</li> <li>RTSI trigger line 3. (PCI devices)</li> </ul>

NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. (PCI devices)
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

# NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeRefTri

Specifies the active edge for the <u>Reference trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_REF_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_RISING_EDGE (12)	Rising-edge trigger.
NIHSDIO_VAL_FALLING_EDGE (13)	Falling-edge trigger.

# NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the position where the <u>Reference trigger</u> is asserted, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a <u>front panel connector</u>.

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	th
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	th
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Tł
	th
	CS
	ra
	นร
	<u>N</u> ]
	Tł
	or
	Sć
	w
	th

# NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal R/W N/A None

Specifies the impedance on the channel configured for the <u>digital edge</u> <u>Reference trigger</u>. Valid values for this attribute vary by device.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in LVDS terminal configuration 10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

# NIHSDIO_ATTR_DIGITAL_EDGE_REF_TRIGGER_

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies whether the <u>Reference trigger</u> terminal is configured for singleended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_PATTERN_MATCH_REF_TRIGGE

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigurePatternMatchRefT

Sets the pattern match mask for the <u>Reference trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_REF_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_PATTERN_MATCH.

The pattern is a string of characters representing the entire pattern to be matched on. Each character corresponds to a particular channel.

- 'X': Match on any value
- '1': Match on a logic 1
- '0': Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

Spaces are ignored, and are useful for readability to segment long patterns. The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are '0' and channels 2-7 are '1'. The values seen by pattern matching are affected by NIHSDIO_ATTR_DATA_INTERPRETATION.

This attribute is valid only for acquisition sessions.

# NIHSDIO_ATTR_PATTERN_MATCH_REF_TRIGGE

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigurePatternMatchRefTr

Specifies whether a <u>pattern match</u> <u>Reference trigger</u> asserts when a particular pattern is matched or not matched. This attribute is valid only for acquisition tasks.

NIHSDIO_VAL_PATTERN_MATCHES (36)	The trigger asserts when the pattern matches.
NIHSDIO_VAL_PATTERN_DOES_NOT_MATCH (37)	The trigger asserts when the pattern does not match.

# NIHSDIO_ATTR_EXPORTED_REF_TRIGGER_OU

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for exporting the <u>Reference trigger</u>. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This attribute is valid only for acquisition sessions.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_EXPORTED_REF_TRIGGER_TEF

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the exported <u>Reference trigger</u> output terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.
NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_START_TO_REF_TRIGGER_HOL

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None None

Use this attribute to specify the amount of time after a <u>Start trigger</u> before a <u>Reference trigger</u> can be recognized. If the pretrigger time (pretrigger samples / frequency) is greater than this attribute, then the holdoff value is ignored.

This attribute is especially useful when you want each device in a multidevice situation to recognize the Reference trigger at the same time, though the Reference trigger is shared among devices and each device has a different pretrigger count.

Units: seconds

# NIHSDIO_ATTR_REF_TO_REF_TRIGGER_HOLD(

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViReal64 R/W N/A None None

Use this attribute to specify the amount of time until the next record's Reference trigger can be recognized. If the posttrigger time of the current record plus the pretrigger time of the next record (posttrigger record n + pretrigger record n+1 / frequency) is greater than this attribute, then the holdoff value is ignored.

This attribute is especially useful when you want each device in a multidevice situation to recognize the Reference trigger at the same time, though the Reference trigger is shared among devices and each device has a different record size.

Units: seconds

# NIHSDIO_ATTR_ADVANCE_TRIGGER_TYPE

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Lev	el Functions
Vilnt32	R/W	N/A	None	<u>niHSDIO</u>	ConfigureDigitalEdgeAdvan
				<u>niHSDIO</u>	<u>ConfigurePatternMatchAdva</u>
				<u>niHSDIO</u>	<u>ConfigureSoftwareAdvanceT</u>
				<u>niHSDIO</u>	<u>DisableAdvanceTrigger</u>

Use this attribute to specify whether you want the <u>Advance trigger</u> to be a <u>digital edge</u>, <u>pattern match</u>, or <u>software trigger</u>. You can also choose NIHSDIO_VAL_NONE as the value for this attribute.

The Advance trigger is used only in acquisition sessions.

NIHSDIO_VAL_NONE (28)	No Advance trigger is configured.
NIHSDIO_VAL_DIGITAL_EDGE (29)	The Advance trigger is not asserted u The source of the digital edge is spec <u>NIHSDIO_ATTR_DIGITAL_EDGE_AD</u> and the active edge is specified with <u>NIHSDIO_ATTR_DIGITAL_EDGE_AD</u>
NIHSDIO_VAL_SOFTWARE (32)	The Advance trigger is not asserted u You can assert the software trigger by <u>niHSDIO_SendSoftwareEdgeTrigger</u> fun NIHSDIO_VAL_START_TRIGGER as 1
NIHSDIO_VAL_PATTERN_MATCH (31)	The Advance trigger is asserted wher matching condition is met. Configure 1 <u>NIHSDIO_ATTR_PATTERN_MATCH_</u> and <u>NIHSDIO_ATTR_PATTERN_MATCH_</u>

# NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIC

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeAdvar

Specifies the source terminal for the <u>Advance trigger</u>. This attribute is used only when <u>NIHSDIO_ATTR_ADVANCE_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7")	PXI trigger line 7. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)
NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3") NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4") NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5") NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6") NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7") NIHSDIO_VAL_RTSI0_STR ("RTSI0") NIHSDIO_VAL_RTSI1_STR ("RTSI1") NIHSDIO_VAL_RTSI2_STR ("RTSI2") NIHSDIO_VAL_RTSI3_STR ("RTSI3")	<ul> <li>PXI trigger line 3. (PXI devices)</li> <li>PXI trigger line 4. (PXI devices)</li> <li>PXI trigger line 5. (PXI devices)</li> <li>PXI trigger line 6. (PXI devices)</li> <li>PXI trigger line 7. (PXI devices)</li> <li>RTSI trigger line 0. (PCI devices)</li> <li>RTSI trigger line 1. (PCI devices)</li> <li>RTSI trigger line 2. (PCI devices)</li> <li>RTSI trigger line 3. (PCI devices)</li> </ul>

NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. (PCI devices)
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

# NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIC

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalEdgeAdvanc

Specifies the active edge for the <u>Advance trigger</u>. This attribute is used only when <u>NIHSDIO_ATTR_ADVANCE_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_RISING_EDGE (12)	Rising-edge trigger.
NIHSDIO_VAL_FALLING_EDGE (13)	Falling-edge trigger.

# NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIC

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the position where the <u>Advance trigger</u> is asserted, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a <u>front panel connector</u>.

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	th
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	th
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Tł
	th
	CS
	ra
	นร
	<u>N</u> ]
	Tł
	or
	Sć
	w
	th

# NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIC

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal R/W N/A None

Specifies the impedance on the channel configured for the <u>digital edge</u> <u>Advance trigger</u>. Valid values for this attribute vary by device.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in LVDS terminal configuration 10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

## NIHSDIO_ATTR_DIGITAL_EDGE_ADVANCE_TRIC

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the terminal configuration for the exported <u>Advance trigger</u>. This attribute is used to switch between single-ended and LVDS configuration.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_PATTERN_MATCH_ADVANCE_T

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigurePatternMatchAdva

Sets the pattern match mask for the <u>Advance trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_ADVANCE_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_PATTERN_MATCH. The pattern is a string of characters representing the entire pattern to be matched on. Each character corresponds to a particular channel.

- 'X': Match on any value
- '1': Match on a logic 1
- '0': Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

Spaces are ignored, and are useful for readability to segment long patterns. The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are '0' and channels 2-7 are '1'. The values seen by pattern matching are affected by <u>NIHSDIO ATTR DATA INTERPRETATION</u>.

# NIHSDIO_ATTR_PATTERN_MATCH_ADVANCE_T

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigurePatternMatchAdva

Specifies whether a <u>pattern match</u> <u>Advance trigger</u> asserts when a particular pattern is matched or not matched. This attribute is valid only for acquisition tasks.

NIHSDIO_VAL_PATTERN_MATCHES (36)	The trigger asserts when the pattern matches.
NIHSDIO_VAL_PATTERN_DOES_NOT_MATCH (37)	The trigger asserts when the pattern does not match.

## NIHSDIO_ATTR_EXPORTED_ADVANCE_TRIGGE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString R/W None None <u>niHSDIO_ExportSignal</u>

Use this attribute to specify the output terminals for the exported <u>Advance</u> <u>trigger</u>. Setting this attribute to an empty string means that when you commit the session, the signal is removed from that terminal and, if possible, the terminal is tristated. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This attribute is valid only for acquisition sessions.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_EXPORTED_ADVANCE_TRIGGE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString R/W None None None

Specifies whether the <u>Advance trigger</u> output terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_SCRIPT_TRIGGER_TYPE

#### **Specific Attribute**

Data type	Access	Applies to	Coercion	High-Level Functions
--------------	--------	---------------	----------	----------------------

Vilnt32 R/W

Script None Trigger niHSDIO_ConfigureDigitalEdgeScriptT niHSDIO_ConfigureDigitalLevelScript_ niHSDIO_ConfigureSoftwareScriptTrig niHSDIO_DisableScriptTrigger

Specifies the <u>Script trigger</u> type. Depending upon the value of this attribute, more attributes may be needed to fully configure the trigger. This attribute is only valid for generation sessions.
NIHSDIO_VAL_NONE (28)	The operation does not use a Scripstarts immediately after you call <u>ni</u>
NIHSDIO_VAL_DIGITAL_EDGE (29)	The Script trigger asserts when a c source of the digital edge is specifi <u>NIHSDIO_ATTR_DIGITAL_EDGE</u> and the active edge is specified wi <u>NIHSDIO_ATTR_DIGITAL_EDGE</u>
NIHSDIO_VAL_DIGITAL_LEVEL (30)	The Script trigger is active when th matches the desired level. The sou specified with <u>NIHSDIO_ATTR_DIGITAL_LEVEL</u> and the desired level is specified w <u>NIHSDIO_ATTR_DIGITAL_LEVEL</u>
NIHSDIO_VAL_SOFTWARE (32)	The Script trigger is not recognized create a software trigger by calling <u>niHSDIO_SendSoftwareEdgeTrigger</u> trigger as the trigger name, for exa can assert the software trigger by ( <u>niHSDIO_SendSoftwareEdgeTrigger</u> NIHSDIO_VAL_SCRIPT_TRIGGER

# NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	Script Trigger	None	<u>niHSDIO_ConfigureDigitalEdgeScript</u>

Specifies the source terminal for the <u>Script trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_SCRIPT_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7")	PXI trigger line 7. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)
NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3") NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4") NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5") NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6") NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7") NIHSDIO_VAL_RTSI0_STR ("RTSI0") NIHSDIO_VAL_RTSI1_STR ("RTSI1") NIHSDIO_VAL_RTSI2_STR ("RTSI2") NIHSDIO_VAL_RTSI3_STR ("RTSI3")	<ul> <li>PXI trigger line 3. (PXI devices)</li> <li>PXI trigger line 4. (PXI devices)</li> <li>PXI trigger line 5. (PXI devices)</li> <li>PXI trigger line 6. (PXI devices)</li> <li>PXI trigger line 7. (PXI devices)</li> <li>RTSI trigger line 0. (PCI devices)</li> <li>RTSI trigger line 1. (PCI devices)</li> <li>RTSI trigger line 2. (PCI devices)</li> <li>RTSI trigger line 3. (PCI devices)</li> </ul>

NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. (PCI devices)
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

# NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	Script Trigger	None	<u>niHSDIO_ConfigureDigitalEdgeScriptT</u>

Specifies the active edge for the <u>Script trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_SCRIPT_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_EDGE.

NIHSDIO_VAL_RISING_EDGE (12)	Rising-edge trigger.
NIHSDIO_VAL_FALLING_EDGE (13)	Falling-edge trigger.

# NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGG

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal R/W N/A None

Specifies the impedance on the channel configured for the <u>digital edge</u> <u>Script trigger</u>.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in <u>LVDS</u> <u>terminal configuration</u> 10000 in single-ended <u>terminal configuration</u>

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

## NIHSDIO_ATTR_DIGITAL_EDGE_SCRIPT_TRIGG

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies whether the <u>Script trigger</u> terminal is configured for singleended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	Script Trigger	None	niHSDIO_ConfigureDigitalLevelScrip

Specifies the source terminal for the <u>Script trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_SCRIPT_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_LEVEL.

NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7")	PXI trigger line 7. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)
NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3") NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4") NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5") NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6") NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7") NIHSDIO_VAL_RTSI0_STR ("RTSI0") NIHSDIO_VAL_RTSI1_STR ("RTSI1") NIHSDIO_VAL_RTSI2_STR ("RTSI2") NIHSDIO_VAL_RTSI3_STR ("RTSI3")	<ul> <li>PXI trigger line 3. (PXI devices)</li> <li>PXI trigger line 4. (PXI devices)</li> <li>PXI trigger line 5. (PXI devices)</li> <li>PXI trigger line 6. (PXI devices)</li> <li>PXI trigger line 7. (PXI devices)</li> <li>RTSI trigger line 0. (PCI devices)</li> <li>RTSI trigger line 1. (PCI devices)</li> <li>RTSI trigger line 2. (PCI devices)</li> <li>RTSI trigger line 3. (PCI devices)</li> </ul>

NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. (PCI devices)
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

# NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGG

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	Script Trigger	None	niHSDIO_ConfigureDigitalLevelScript [*] .

Specifies the active level for the <u>Script trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_SCRIPT_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_DIGITAL_LEVEL.

NIHSDIO_VAL_HIGH (34)	The trigger is asserted when the signal is high level.
NIHSDIO_VAL_LOW (35)	The trigger is asserted when the signal is low level.

# NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGG

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal R/W N/A None

Specifies the impedance on the channel configured for the <u>digital level</u> <u>Script trigger</u>.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in <u>LVDS</u> terminal configuration 10000 in single-ended terminal configuration

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

## NIHSDIO_ATTR_DIGITAL_LEVEL_SCRIPT_TRIGG

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Script trigger</u> terminal is configured for singleended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

## NIHSDIO_ATTR_EXPORTED_SCRIPT_TRIGGER_

#### **Specific Attribute**

Data Access Applies to Coercion High-Level Functions type niHSDIO_ExportSignal

Script Trigger None ViString R/W

Use this attribute to specify the output terminals for the exported <u>Script</u> <u>trigger</u>. Setting this attribute to an empty string means that when you commit the session, the signal is removed from that terminal and, if possible, the terminal is tristated. Event voltages and positions are only relevant if the destination of the event is a front panel connector. This attribute is valid only for generation sessions.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

## NIHSDIO_ATTR_EXPORTED_SCRIPT_TRIGGER_

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Script trigger</u> output terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_PAUSE_TRIGGER_TYPE

## **Specific Attribute**

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalLevelPause
				niHSDIO_ConfigurePatternMatchPause
				niHSDIO_DisablePauseTrigger

Specifies the <u>Pause trigger</u> type. Depending upon the value of this attribute, you may need to set more attributes to fully configure the trigger.

NIHSDIO_VAL_NONE (28)	The Pause trigger does not assemed as a seme trigger does not assemed as the seme trigger does not asseme trigger does not asse
NIHSDIO_VAL_DIGITAL_LEVEL (30)	The level trigger is not active unti the desired level. The source of t <u>NIHSDIO_ATTR_DIGITAL_LEVE</u> and the desired level is specified <u>NIHSDIO_ATTR_DIGITAL_LEVE</u>
NIHSDIO_VAL_PATTERN_MATCH (31)	The data operation is paused wh matching condition is met. Config <u>NIHSDIO_ATTR_PATTERN_MAT</u> and <u>NIHSDIO_ATTR_PATTERN_MAT</u> This is valid only for acquisition s

# NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO_ConfigureDigitalLevelPause
Specifies the source terminal for the <u>Pause trigger</u>. This attribute only applies to acquisition operations.

NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7")	PXI trigger line 7. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)
NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3") NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4") NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5") NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6") NIHSDIO_VAL_PXI_TRIG7_STR ("PXI_Trig7") NIHSDIO_VAL_RTSI0_STR ("RTSI0") NIHSDIO_VAL_RTSI1_STR ("RTSI1") NIHSDIO_VAL_RTSI2_STR ("RTSI2") NIHSDIO_VAL_RTSI3_STR ("RTSI3")	<ul> <li>PXI trigger line 3. (PXI devices)</li> <li>PXI trigger line 4. (PXI devices)</li> <li>PXI trigger line 5. (PXI devices)</li> <li>PXI trigger line 6. (PXI devices)</li> <li>PXI trigger line 7. (PXI devices)</li> <li>RTSI trigger line 0. (PCI devices)</li> <li>RTSI trigger line 1. (PCI devices)</li> <li>RTSI trigger line 2. (PCI devices)</li> <li>RTSI trigger line 3. (PCI devices)</li> </ul>

NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)
NIHSDIO_VAL_RTSI7_STR ("RTSI7")	RTSI trigger line 7. (PCI devices)
NIHSDIO_VAL_PXI_STAR_STR ("PXI_STAR")	The device will use the PXI_STAR signal which is present on the PXI backplane. This selection is valid only for PXI devices in slots other than Slot 2.

# NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGG

## Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigureDigitalLevelPause]

Specifies the active level for pausing the dynamic operation.

NIHSDIO_VAL_HIGH (34)	The trigger is asserted when the signal is high level.
NIHSDIO_VAL_LOW (35)	The trigger is asserted when the signal is low level.

# NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGG

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViInt32 R/W N/A None None

Specifies the position where the digital level Pause trigger is asserted, relative to the Sample clock. Trigger voltages and positions are only relevant if the trigger source is a <u>front panel connector</u>.

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	th
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	th
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Tł
	th
	CS
	ra
	นร
	<u>N</u> ]
	Tł
	or
	Sć
	w
	th

# NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGG

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Pause trigger</u> terminal is configured for singleended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_DIGITAL_LEVEL_PAUSE_TRIGG

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal R/W N/A None

Specifies the impedance on the channel configured for the <u>digital level</u> <u>Pause trigger</u>.

Refer to the following table for the supported settings for your device. NI-HSDIO returns an error if you select an unsupported setting.

Device	Supported Value
NI 654 <i>x</i>	100
NI 655 <i>x</i>	50 or 10000
NI 656 <i>x</i>	100 in <u>LVDS</u> <u>terminal configuration</u> 10000 in single-ended <u>terminal configuration</u>

This attribute is only set if the trigger is configured to use a PFI channel, and it is ignored if the trigger is configured for any other channel.

Units: ohms

# NIHSDIO_ATTR_PATTERN_MATCH_PAUSE_TRIG

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
ViString	R/W	N/A	None	niHSDIO ConfigurePatternMatchPaus

Sets the pattern match mask for the <u>Pause trigger</u>. This attribute is used when <u>NIHSDIO_ATTR_PAUSE_TRIGGER_TYPE</u> is set to NIHSDIO_VAL_PATTERN_MATCH. The pattern is a string of characters representing the entire pattern to be matched on. Each character corresponds to a particular channel.

- 'X': Match on any value
- '1': Match on a logic 1
- '0': Match on a logic 0
- R or r: Match on a rising edge
- F or f: Match on a falling edge
- E or e: Match on either edge

Spaces are ignored, and are useful for readability to segment long patterns. The rightmost character in the expression corresponds to the lowest numbered physical channel. For example, XXXX XXXX XXXX 1111 1100 specifies to match when channels 0 and 1 are '0' and channels 2-7 are '1'. The values seen by pattern matching are affected by <u>NIHSDIO ATTR DATA INTERPRETATION</u>.

This attribute is only valid for acquisition sessions.

# NIHSDIO_ATTR_PATTERN_MATCH_PAUSE_TRIG

# Specific Attribute

Data type	Access	Applies to	Coercion	High-Level Functions
Vilnt32	R/W	N/A	None	niHSDIO_ConfigurePatternMatchPause

Specifies whether a <u>pattern match</u> <u>Pause trigger</u> asserts when a particular pattern is matched or not matched. This attribute is valid only for acquisition sessions.

NIHSDIO_VAL_PATTERN_MATCHES (36)	The trigger asserts when the pattern matches.
NIHSDIO_VAL_PATTERN_DOES_NOT_MATCH (37)	The trigger asserts when the pattern does not match.

# NIHSDIO_ATTR_EXPORTED_PAUSE_TRIGGER_(

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for the exported <u>Pause trigger</u>. Event voltages and positions are only relevant if the destination of the event is one of the front panel connectors.

This attribute is only valid for generation sessions.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_EXPORTED_PAUSE_TRIGGER_1

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Pause trigger</u> output terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

## NIHSDIO_ATTR_READY_FOR_START_EVENT_O

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for the <u>Ready for Start</u> event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_READY_FOR_START_EVENT_LE

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the output polarity of the <u>Ready for Start</u> event.

NIHSDIO_VAL_ACTIVE_HIGH (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. This attribute does not apply to other exported signals.
NIHSDIO_VAL_ACTIVE_LOW (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts. This attribute does not apply to other exported signals.

# NIHSDIO_ATTR_READY_FOR_START_EVENT_TE

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Ready for Start</u> event terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_READY_FOR_ADVANCE_EVENT

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>
Specifies the destination terminal for the <u>Ready for Advance</u> event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_READY_FOR_ADVANCE_EVENT

### **Specific Attribute**

### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the output polarity of the <u>Ready for Advance</u> event.

NIHSDIO_VAL_ACTIVE_HIGH (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. This attribute does not apply to other exported signals.
NIHSDIO_VAL_ACTIVE_LOW (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts. This attribute does not apply to other exported signals.

# NIHSDIO_ATTR_READY_FOR_ADVANCE_EVENT

### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Ready for Advance</u> event terminal is configured for single-ended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_END_OF_RECORD_EVENT_OUT

### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for the <u>End of Record</u> event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_END_OF_RECORD_EVENT_PUL

### **Specific Attribute**

### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies the output polarity of the End of Record event. This attribute is valid only for acquisition sessions.

NIHSDIO_VAL_ACTIVE_HIGH (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. This attribute does not apply to other exported signals.
NIHSDIO_VAL_ACTIVE_LOW (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts. This attribute does not apply to other exported signals.

# NIHSDIO_ATTR_END_OF_RECORD_EVENT_TER

### **Specific Attribute**

### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies the terminal configuration for the operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

## NIHSDIO_ATTR_DATA_ACTIVE_EVENT_OUTPUT

### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for the <u>Data Active</u> event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This attribute is valid only for generation sessions.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.

# NIHSDIO_ATTR_DATA_ACTIVE_EVENT_LEVEL_/

### **Specific Attribute**

### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Specifies the output polarity of the <u>Data Active</u> event.

NIHSDIO_VAL_ACTIVE_HIGH (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. This attribute does not apply to other exported signals.
NIHSDIO_VAL_ACTIVE_LOW (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts. This attribute does not apply to other exported signals.

## NIHSDIO_ATTR_DATA_ACTIVE_EVENT_POSITIO

### **Specific Attribute**

### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None None

Use this attribute to specify the position of the <u>Data Active</u> event relative to the Sample clock. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	Sá
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	Sá
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Τŀ
	de
	Sţ
	<u>N</u> ]
	Tł
	or
	Sć
	W
	th

## NIHSDIO_ATTR_DATA_ACTIVE_EVENT_TERMIN/

### **Specific Attribute**

### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies whether the <u>Data Active</u> event terminal is configured for singleended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

## NIHSDIO_ATTR_MARKER_EVENT_OUTPUT_TER

### **Specific Attribute**

### Data type Access Applies to Coercion High-Level Functions

ViString R/W Marker None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for the <u>Marker event</u>. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This attribute is valid only for generation sessions.

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_MARKER_EVENT_PULSE_POLA

### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W Marker None None
Specifies the output polarity of the <u>Marker event</u>. This attribute is valid only for generation sessions.

# **Defined Values:**

NIHSDIO_VAL_ACTIVE_HIGH (10)	The exported signal is low level while the event is deasserted. A high pulse occurs when the event asserts. This attribute does not apply to other exported signals.
NIHSDIO_VAL_ACTIVE_LOW (11)	The exported signal is high level while the event is deasserted. A low pulse occurs when the event asserts. This attribute does not apply to other exported signals.

# NIHSDIO_ATTR_MARKER_EVENT_POSITION

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W Marker None None

Use this attribute to specify the position of the <u>Marker event</u> relative to the <u>Sample clock</u>. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

This attribute is valid only for generation sessions.

## **Defined Values:**

NIHSDIO_VAL_SAMPLE_CLOCK_RISING_EDGE (18)	Τł
	Sá
NIHSDIO_VAL_SAMPLE_CLOCK_FALLING_EDGE (19)	Tł
	Sá
NIHSDIO_VAL_DELAY_FROM_SAMPLE_CLOCK_RISING_EDGE (20)	Τŀ
	de
	Sţ
	<u>N</u> ]
	Tł
	or
	Sć
	W
	th

# NIHSDIO_ATTR_MARKER_EVENT_TERMINAL_C

#### **Specific Attribute**

Data type Access Applies to Coercion High Level Functions

ViInt32 R/W N/A None None

Specifies whether the <u>Marker</u> event terminal is configured for singleended or LVDS operation. Valid values for this attribute vary by device. Refer to your device documentation to determine if your hardware supports LVDS operation.

# **Defined Values:**

NIHSDIO_VAL_LVDS (64)	The terminal will be configured for LVDS voltage levels.
NIHSDIO_VAL_SINGLE_ENDED (65)	The terminal will be configured for single-ended voltage levels.

# NIHSDIO_ATTR_SAMPLE_ERROR_EVENT_OUTF

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViString R/W N/A None <u>niHSDIO_ExportSignal</u>

Specifies the destination terminal for the Sample Error event. Event voltages and positions are only relevant if the destination of the event is a front panel connector.

# **Defined Values:**

NIHSDIO_VAL_DO_NOT_EXPORT_STR ("None")	The signal is not exported.
NIHSDIO_VAL_PFI0_STR ("PFI0")	PFI 0 on the front panel SMB jack connector.
NIHSDIO_VAL_PFI1_STR ("PFI1")	PFI 1 on the front panel DDC connector.
NIHSDIO_VAL_PFI2_STR ("PFI2")	PFI 2 on the front panel DDC connector.
NIHSDIO_VAL_PFI3_STR ("PFI3")	PFI 3 on the front panel DDC connector.
NIHSDIO_VAL_PXI_TRIG0_STR ("PXI_Trig0")	PXI trigger line 0. (PXI devices)
NIHSDIO_VAL_PXI_TRIG1_STR ("PXI_Trig1")	PXI trigger line 1. (PXI devices)
NIHSDIO_VAL_PXI_TRIG2_STR ("PXI_Trig2")	PXI trigger line 2. (PXI devices)
NIHSDIO_VAL_PXI_TRIG3_STR ("PXI_Trig3")	PXI trigger line 3. (PXI devices)
NIHSDIO_VAL_PXI_TRIG4_STR ("PXI_Trig4")	PXI trigger line 4. (PXI devices)
NIHSDIO_VAL_PXI_TRIG5_STR ("PXI_Trig5")	PXI trigger line 5. (PXI devices)
NIHSDIO_VAL_PXI_TRIG6_STR ("PXI_Trig6")	PXI trigger line 6. (PXI devices)
NIHSDIO_VAL_RTSI0_STR ("RTSI0")	RTSI trigger line 0. (PCI devices)
NIHSDIO_VAL_RTSI1_STR ("RTSI1")	RTSI trigger line 1. (PCI devices)
NIHSDIO_VAL_RTSI2_STR ("RTSI2")	RTSI trigger line 2. (PCI devices)

NIHSDIO_VAL_RTSI3_STR ("RTSI3")	RTSI trigger line 3. (PCI devices)
NIHSDIO_VAL_RTSI4_STR ("RTSI4")	RTSI trigger line 4. (PCI devices)
NIHSDIO_VAL_RTSI5_STR ("RTSI5")	RTSI trigger line 5. (PCI devices)
NIHSDIO_VAL_RTSI6_STR ("RTSI6")	RTSI trigger line 6. (PCI devices)

# NIHSDIO_ATTR_TOTAL_ACQUISITION_MEMORY

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViInt32 RO N/A None None

Returns the total <u>onboard memory</u> size, in samples, for acquiring data. The number of samples is based on the default device data width.

If you configure your device to use a different data width, the total memory size is actually the value returned by this attribute multiplied by the quotient of the default data width divided by the configured data width. For example, if you configure 1-byte data width for a 2-byte device, the total acquisition memory size is twice the number of samples that is returned by this attribute.

Units: samples

# NIHSDIO_ATTR_TOTAL_GENERATION_MEMORY

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

Vilnt32 RO N/A None None

Returns the total <u>onboard memory</u> size, in samples, for generating data. The number of samples is based on the default device data width.

If you configure your device to use a different data width, the total memory size is actually the value returned by this attribute multiplied by the quotient of the default data width divided by the configured data width. For example, if you configure 1-byte data width for a 2-byte device, the total generation memory size is twice the number of samples that is returned by this attribute.

Units: samples

# NIHSDIO_ATTR_SERIAL_NUMBER

## **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViString RO None None None

Returns the device serial number.

# NIHSDIO_ATTR_DATA_WIDTH

# Specific Attribute

Data type	Access	Applies to	Coercion	High Level Functions
Vilnt32	R/W for acquisition sessions, RO for generation sessions	N/A	None	None

Indicates, in bytes, the size of a raw sample from the operation. **Valid Values**: Vary by <u>device</u>.

NI 654x 1, 2, 4 NI 655x 1, 2, 4 NI 656x 1, 2

# NIHSDIO_ATTR_DATA_RATE_MULTIPLIER

#### **Specific Attribute**

#### Data type Access Applies to Coercion High Level Functions

Vilnt32 R/W N/A None None

Specifies whether you want the device to acquire or generate in single data rate (SDR) mode or in double data rate (DDR) mode.

# **Defined Values:**

NIHSDIO_VAL_SINGLE_DATA_RATE (1)	The device will sample or generate data in <u>single data</u> <u>rate (SDR) mode</u> . When in SDR mode, the NI digital waveform generator/analyzer generates or acquires data on a single edge of the Sample clock. Therefore, you can generate or acquire data on the rising or falling edge of every Sample clock pulse or on a delayed version of the rising edge of the Sample clock.
NIHSDIO_VAL_DOUBLE_DATA_RATE (2)	The device will sample or generate data in <u>double data</u> rate (DDR) mode. When in DDR mode, the NI digital waveform generator/analyzer generates or acquires data on both edges of the Sample clock. Therefore, you can generate or acquire data on every rising and falling edge of the Sample clock. Acquisition and generation sessions can be configured in DDR mode to acquire or generate the first data sample on the rising or falling edge of the clock or on a delayed version of the rising edge of the clock.

# NIHSDIO_ATTR_DATA_ACTIVE_INTERNAL_ROU

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

Vilnt32 R/W N/A None N/A

Configures the number of Sample clock cycles to delay the internal <u>Data</u> <u>Active event</u>. Internally routing a delayed version of this event is useful when you want to synchronize an acquisition trigger to the generation operation. Use this coarse delay together with the finer-resolution data delay to compensate for the round trip delay of data in stimulus/response operations.

You can configured the delayed Data Active event as the source for any acquisition trigger by manually entering DelayedDataActiveEvent as the **triggerSource** parameter.

This attribute is only applicable in acquisition sessions.

Valid values for this attribute are 0 to 24.

Units: Sample clock cycles

# NIHSDIO_ATTR_HWC_HARDWARE_COMPARE_N

#### **Specific Attribute**

# Data type Access Applies to Coercion High-Level Functions

Viln32 R/W None None None

Configures the device to compare expected data and actual data in realtime. This attribute must be set to the same value in both sessions.



**Note** To use this feature you must have an acquisition and a generation session running concurrently.

When you set this attribute to either

NIHSDIO_VAL_HWC_STIMULUS_AND_EXPECTED_RESPONSE or NIHSDIO_VAL_HWC_EXPECTED_RESPONSE_ONLY, the generation engine sends expected data to the acquisition session to compare against acquired data.

Use the <u>niHSDIO_WriteNamedWaveformWDT</u> function to write expected data to the device. The device drives any values of 0, 1, or Z in the waveform, while values of H, L, or X are treated as expected data values.

NIHSDIO_VAL_HWC_DISABLED (77)	Co
	enț
	dis
	An
	atte
	wri
	ex
	res
	dat
	de۱
	res
	err
NIHSDIO_VAL_HWC_STIMULUS_AND_EXPECTED_RESPONSE (78)	De
	driv
	100
	dat
	sar
	ses
	Yo
	do۱
	wa
l	1

This property must be set before data is written to the device.

	wit
	and
	100
	dai
	aig
	sta
	SU
	In t
	mc
NIHSDIO_VAL_HWC_EXPECTED_RESPONSE_ONLY (79)	De
	do
	drn
	dai
	acc
	and
	COI
	dov
	driv
	Ser
	dov
	the
	exi
	wa
	Se
	this
	pai
	gei
	lin€
	syr
	the
	acc
	and
	gei
	ses

# NIHSDIO_ATTR_HWC_SAMPLE_ERROR_BACKL

#### **Specific Attribute**

# Data type Access Applies to Coercion High-Level Functions

Viln32 RO None None None

Returns the number of sample errors available you can read using the <u>niHSDIO_HWC_FetchSampleErrors</u> function.

# NIHSDIO_ATTR_HWC_NUM_SAMPLE_ERRORS

## **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

Viln32 RO None None None

Returns the total number of sample errors since the acquisition was initiated. Use this attribute, along with

NIHSDIO_ATTR_HWC_SAMPLES_COMPARED, to calculate the sample error rate.

# NIHSDIO_ATTR_HWC_SAMPLES_COMPARED

## **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViReal64 RO None None None

Returns the total number of samples compared since the acquisition was initiated. Use this attribute, along with

<u>NIHSDIO ATTR HWC NUM SAMPLE ERRORS</u>, to calculate sample error rate.

# NIHSDIO_ATTR_HWC_FILTER_REPEATED_SAMI

#### **Specific Attribute**

#### Data type Access Applies to Coercion High-Level Functions

ViBoolean R/W None None None

Specifies whether the device stores and counts errors when the same error appears in consecutive samples. If this attribute is set to VI_TRUE, the device only counts distinct errors. An error is defined as distinct if the expected response value and the actual sample error do not change over the same number of Sample clock cycles. The

<u>niHSDIO_HWC_FetchSampleErrors</u> function returns the number of clock cycles for which the repeated error occurred.

This attribute is helpful if your NI device clock rate is significantly faster than your DUT clock rate. In this case, one error from the DUT could result in several identical errors on the device.
# NIHSDIO_ATTR_HWC_SAMPLE_ERROR_BUFFEI

#### **Specific Attribute**

Data type Access Applies to Coercion High-Level Functions

ViBoolean RO None None None

#### Description

Returns whether the buffer used to store sample errors has overflowed. The NI 655x FIFO can contain 4,094 sample errors. If the buffer overflows, the hardware stops storing error information for further errors, but it continues to compare data and count the sample errors encountered.

You can remove sample errors from the FIFO using the <u>niHSDIO_HWC_FetchSampleErrors</u> function. Removing sample errors creates room for additional sample errors to be stored in the FIFO.

# **Alphabetical Attribute List and Default Values**

The following table lists the default values for each property you can configure for your device. An "N/A" in a table cell indicates that the listed property is not supported for that device. A dash indicates that the property does not have a default value or that it is a read-only property. "" is used in the following ways:

- In output terminal properties to indicate to the device not to export the relevant signal
- In trigger source properties to the device that the relevant trigger is not used
- In dynamic and static channels to means "all channels"

C/C++ Attribute	
ADVANCE TRIGGER TYPE	
DATA_ACTIVE_EVENT _LEVEL_ACTIVE_LEVEL	
DATA_ACTIVE_EVENT _OUTPUT_TERMINAL	
DATA_ACTIVE_EVENT_POSITION	
DATA_ACTIVE_EVENT TERMINAL_CONFIGURATION	
DATA_ACTIVE_INTERNAL_ROUTE_DELAY	
DATA_INTERPRETATION	
DATA_POSITION	
DATA_POSITION_DELAY	
NIHSDIO_ATTR_DATA_RATE_MULTIPLIER	
NIHSDIO_ATTR_DATA_TRANSFER_BLOCK_SIZE	
DATA_VOLTAGE_HIGH_LEVEL	3.3 Refe
DATA_VOLTAGE_LOW_LEVEL	3.3

	Refe
NIHSDIO_ATTR_DATA_WIDTH	
DIGITAL_EDGE_ADVANCE _TRIGGER_EDGE	
DIGITAL_EDGE_ADVANCE _TRIGGER_IMPEDANCE	
<u>DIGITAL_EDGE_ADVANCE</u> _ <u>TRIGGER_POSITION</u>	
<u>DIGITAL_EDGE_ADVANCE</u> _ <u>TRIGGER_SOURCE</u>	
DIGITAL_EDGE_ADVANCE_TRIGGER _TERMINAL_CONFIGURATION	
DIGITAL_EDGE_REF _TRIGGER_EDGE	
DIGITAL_EDGE_REF _TRIGGER_IMPEDANCE	
DIGITAL_EDGE_REF _TRIGGER_POSITION	SAM
DIGITAL_EDGE_REF_TRIGGER_SOURCE	
DIGITAL_EDGE_REF_TRIGGER _TERMINAL_CONFIGURATION	
DIGITAL_EDGE_SCRIPT_TRIGGER_EDGE	
DIGITAL_EDGE_SCRIPT _TRIGGER_IMPEDANCE	
DIGITAL_EDGE_SCRIPT _TRIGGER_SOURCE	
DIGITAL_EDGE_SCRIPT_TRIGGER _TERMINAL_CONFIGURATION	
DIGITAL_EDGE_START	
TRIGGER_EDGE	

<u>DIGITAL_EDGE_START</u> TRIGGER POSITION	SA
<u>START_TRIGGER_SOURCE</u>	
DIGITAL_EDGE_START_TRIGGER _TERMINAL_CONFIGURATION	
DIGITAL_LEVEL_PAUSE _TRIGGER_IMPEDANCE	
DIGITAL_LEVEL_PAUSE _TRIGGER_POSITION	SA
DIGITAL_LEVEL_PAUSE _TRIGGER_SOURCE	
DIGITAL_LEVEL_PAUSE_TRIGGER _TERMINAL_CONFIGURATION	
<u>DIGITAL_LEVEL_PAUSE</u> _ <u>TRIGGER_WHEN</u>	
DIGITAL_LEVEL_SCRIPT _TRIGGER_IMPEDANCE	
DIGITAL_LEVEL_SCRIPT _TRIGGER_SOURCE	
DIGITAL_LEVEL_SCRIPT_TRIGGER _TERMINAL_CONFIGURATION	
DIGITAL_LEVEL_SCRIPT _TRIGGER_WHEN	
DIRECT_DMA_ENABLED	
DIRECT_DMA_WINDOW_ADDRESS	
DIRECT_DMA_WINDOW_SIZE	
DYNAMIC_CHANNELS	
<u>END_OF_RECORD_EVENT</u> _OUTPUT_TERMINAL	
<u>END_OF_RECORD_EVENT</u> _ <u>PULSE_POLARITY</u>	
END_OF_RECORD_EVENT TERMINAL_CONFIGURATION	

EVENT_VOLTAGE_HIGH_LEVEL	3.3 Refe
EVENT_VOLTAGE_LOW_LEVEL	3.3 Refe
EXPORTED_ADVANCE_TRIGGER _OUTPUT_TERMINAL	
EXPORTED_ADVANCE_TRIGGER _TERMINAL_CONFIGURATION	
EXPORTED_ONBOARD_REF_CLOCK _OUTPUT_TERMINAL	
EXPORTED_PAUSE_TRIGGER _OUTPUT_TERMINAL	
EXPORTED_PAUSE_TRIGGER _TERMINAL_CONFIGURATION	
EXPORTED_REF_CLOCK _OUTPUT_TERMINAL	
EXPORTED_REF_TRIGGER _OUTPUT_TERMINAL	
EXPORTED_REF_TRIGGER _TERMINAL_CONFIGURATION	
EXPORTED_SAMPLE_CLOCK_DELAY	
EXPORTED_SAMPLE_CLOCK_MODE	
EXPORTED_SAMPLE_CLOCK_OFFSET	
EXPORTED_SAMPLE_CLOCK _OUTPUT_TERMINAL	
EXPORTED_SCRIPT_TRIGGER _OUTPUT_TERMINAL	
EXPORTED_SCRIPT_TRIGGER _TERMINAL_CONFIGURATION	
EXPORTED_START_TRIGGER _OUTPUT_TERMINAL	

EXPORTED_START_TRIGGER	
_TERMINAL_CONFIGURATION	
FETCH_BACKLOG	
FETCH_OFFSET	
FETCH_RELATIVE_TO	MOS
	RE
	Ref
NIHSDIO_ATTR_HWC_FILTER_REPEATED_SAMPLE_ERRORS	
GENERATION_MODE	
NIHSDIO_ATTR_HWC_HARDWARE_COMPARE_MODE	
IDLE_STATE	
INITIAL_STATE	
INPUT_IMPEDANCE	
MARKER_EVENT	
_OUTPUT_TERMINAL	
MARKER_EVENT_POSITION	
MARKER_EVENT_PULSE_POLARITY	
MARKER EVENT	
_TERMINAL_CONFIGURATION	
NUM_RECORDS	
NIHSDIO_ATTR_HWC_NUM_SAMPLE_ERRORS	
OSCILLATOR_PHASE_DAC_VALUE	
PATTERN_MATCH_ADVANCE	
TRIGGER_PATTERN	
PATTERN_MATCH_ADVANCE	
PATTERN_MATCH_PAUSE TRIGGER_PATTERN	
PATTERN MATCH PAUSE	
TRIGGER WHEN	

PATTERN_MATCHES_REF TRICCER_PATTERN	
PATTERN MATCH REF	
_TRIGGER_WHEN	
PATTERN_MATCH_START _TRIGGER_PATTERN	
PATTERN MATCH START _TRIGGER_WHEN	
PAUSE_TRIGGER_TYPE	
READY_FOR_ADVANCE_EVENT _LEVEL_ACTIVE_LEVEL	
READY_FOR_ADVANCE_EVENT _OUTPUT_TERMINAL	
READY_FOR_ADVANCE_EVENT TERMINAL_CONFIGURATION	
READY_FOR_START_EVENT _LEVEL_ACTIVE_LEVEL	
READY_FOR_START_EVENT _OUTPUT_TERMINAL	
READY_FOR_START_EVENT _TERMINAL_CONFIGURATION	
RECORDS_DONE	
REF_CLOCK_IMPEDANCE	
REF_CLOCK_RATE	
<u>REF_CLOCK_SOURCE</u>	
REF_TRIGGER_PRETRIGGER_SAMPLES	
<u>REF_TRIGGER_TYPE</u>	
REPEAT_COUNT	
<u>REPEAT_MODE</u>	
SAMPLE_CLOCK_IMPEDANCE	
SAMPLE_CLOCK_RATE	
SAMPLE_CLOCK_SOURCE	

NIHSDIO_ATTR_HWC_SAMPLE_ERROR_BACKLOG	
NIHSDIO_ATTR_HWC_SAMPLE_ERROR_BUFFER_OVERFLOWED	
NIHSDIO_ATTR_HWC_SAMPLES_COMPARED	
SAMPLES_PER_RECORD	
SCRIPT_TO_GENERATE	
SCRIPT_TRIGGER_TYPE	
NIHSDIO_ATTR_SERIAL_NUMBER	
SPACE_AVAILABLE_IN_STREAMING_WAVEFORM	
START_TRIGGER_TYPE	
<u>STATIC_CHANNELS</u>	
STREAMING_ENABLED	
STREAMING_WAVEFORM_NAME	
TOTAL_ACQUISITION_SIZE	
TOTAL_GENERATION_SIZE	
TRIGGER_VOLTAGE_HIGH_LEVEL	3.3
	Refe
TRIGGER_VOLTAGE_LOW_LEVEL	3.3
	Refe
WAVEFORM_TO_GENERATE	

# **Return Value**

Reports the status of this operation. To obtain a text description of the status code, call the <u>niHSDIO_error_message</u> function. To obtain additional information concerning the error condition, use the <u>niHSDIO_GetError</u> and <u>niHSDIO_ClearError</u> functions.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

# **Operating System Support**

For information about the supported operating system (OS) for your device, refer to the <u>NI-HSDIO Instrument Driver Readme</u>.



**Note** Some devices are not supported under Windows Vista. Refer to your product <u>readme</u> for a complete list of products and their OS support.

# Glossary

Prefixes Numbers/Symbols A B C D E F G H I J L M N O P R S T U V W X Z

Prefix	Meaning	Value
р	pico	10-12
n	nano	10-9
μ	micro	10-6
m	milli	10-3
k	kilo	10 ³
М	mega	106
G	giga	10 ⁹

#### **Numbers and Symbols**

- ° degrees
- negative of, or minus
- < less than
- > greater than
- $\leq$  less than or equal to
- $\geq$  greater than or equal to
- $\Omega$  ohms
- / per
- % percent
- ± plus or minus

# Α

Α

amps

- aberration Signal distortions that cause imperfections in the shape or sharpness of the signal.
- active drive A drive type where the generation voltage high level is configured as the voltage produced at the channel electronics when the Pattern Generation Engine generates a binary one.
- ADE application development environment
- API application programming interface—a standardized set of subroutines or functions, along with the parameters that a program can call.
- asynchronous For hardware, it is a property of an event that occurs at an arbitrary time, without synchronization to a reference clock. In software, it is the property of a function that begins an operation and returns prior to the completion or termination of the operation.

#### В

b bits

B bytes

bidirectional Data channels that can be programmatically configured data as acquisition or generation.

channels

bit Single value for a single position in time, on a single line. A bit can have four possible values: 0, 1, X, and Z.

buffer

- 1. Temporary storage for acquired or generated data (software).
- 2. A collection of samples.

bus

- 1. Group of conductors that interconnect individual circuitry in a computer. Typically, a bus is the expansion vehicle to which I/O or other devices are connected.
  - 2. A logical grouping of multiple channels.

# С

cache	High-speed processor memory that buffers commonly used instructions or data to increase processing throughput.	
channel	Single digital terminal, used for generating and/or acquiring data.	
characteristic impedance	Transmission line parameter that determines how propagating signals are transmitted or reflected in the line.	
clock	<ol> <li>Hardware component that controls timing for reading from or writing to channels.</li> <li>Periodic digital edges that can be used to measure time.</li> </ol>	
CompactPCI	Core specification defined by the PCI Industrial Computer Manufacturer's Group (PICMG).	
compare data	Expected response data from your device under test (DUT).	
control signals	Signals that regulate/control the data transfer.	
counter/timer	A circuit that counts external pulses or clock pulses (timing).	
crosstalk	Ratio, in dB, of the level of the interference on the affected channel to the actual level of the interfering signal.	
current sinking	The ability to dissipate current for analog or digital signals.	
current sourcing	The ability to supply current for analog or digital signals.	

#### D

- DAQ Data Acquisition—Collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing. Also refers to collecting and measuring the same kinds of electrical signals with analog-to-digital and/or digital devices plugged into a PC, and possibly generating control signals with digital-to-analog and/or digital devices in the same PC.
- Data Active The Data Active event indicates when the Pattern event Generation Engine is generating data. If the Pattern Generation Engine is waiting for a trigger or is paused, the Data Active event is deasserted. When the Pattern Generation Engine is generating data, the Data Active event is asserted, synchronous with the data.

data Data interpretation determines whether the input signal interpretation is acquired as a 0 or a 1, based on how it relates to the Acquisition Voltage High and Low Levels and the configured data interpretation mode.

In High or Low mode, when the input signal is sampled below Acquisition Voltage Low Level, a 0 is received. A 1 is not recognized until the acquired signal passes above Acquisition Voltage Low Level and above Acquisition Voltage High Level. Conversely, if the acquired signal was last sampled above Acquisition Voltage High (as a 1), the signal is not be sampled as a 0 until the signal is sampled below Acquisition Voltage High Level and below Acquisition Voltage Low Level.

In Valid or Invalid mode, signals sampled between the Acquisition Voltage High and Acquisition Voltage Low Levels are returned as a 1, while signals sampled either above Acquisition Voltage High Level or below Acquisition Voltage Low Level are returned as a 0.

- data rate an attribute that specifies whether the device to acquires or generates in single data rate (SDR) mode or in double data rate (DDR) mode.
- DC direct current
- default Default parameter value recorded in the driver. In many

### Ε

End of An event that indicates when the device enters its Record Complete state, which indicates that the current record has event been acquired.

event Events are emitted to signify a device state change, the arrival of a certain kind of sample, the production of a certain number of samples, or the passage of time.

eye Diagram constructed by looking at the outputs of a digital diagram transmitter over three periods of the main system clock. For more information, refer to the Digital Waveform Timing document on ni.com/zone.

#### F

- fall time The time that it takes a signal to fall from 80% to 20% of the voltage between the voltage low level and the voltage high level.
- fetch An operation that transfers acquired waveform data from device memory to PC memory.
- FPGA field-programmable gate array—Fundamentally, an FPGA is a semi-conductor device which contains a large quantity of gates (logic devices), which are not interconnected, and whose function is determined by a wiring list, which is downloaded to the FPGA.
- function Set of software instructions executed by a single line of code that can have input and/or output parameters and returns a value when executed.

# G

group Collection of <u>lines.</u>

#### Η

high For generation, the high level is the voltage produced when a level binary one is generated. For acquisition, the high level is the voltage threshold above which the input will be sampled as a binary one.

I/O	input/output—Transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.
idle state	Specifies the values of the channels when the generation operation is paused or has completed.
I _{IH}	current input high—The maximum amount of current required on the input pin when the voltage on the input pin is higher than $\underline{V}_{\underline{IH}}$ .
I _{IL}	current input low—The maximum amount of current required on the input pin when the voltage on the input pin is lower than $\underline{V}_{\underline{IL}}$ .
Initial state	Specifies the values of the channels when the generation operation has not yet started.
instructions	Statements used to define a script.
interrupt	Computer signal indicating that the CPU should suspend its current task to service a designated activity.
inter- symbol interference	In a digital transmission system, distortion of the received signal, in which distortion in the form of temporal spreading and consequent overlap of individual pulses to the degree that the receiver cannot reliably distinguish between state changes.
I _{OH}	current output high—The minimum amount of available current on the output pin when the logic device is driving a logic high.
I _{OL}	current output low—The minimum amount of available current on the output pin when the logic device is driving a logic low.

#### J

jitter The deviation from ideal timing of an event. Jitter is typically measured from the zero-crossing of a reference signal. Jitter typically comes from <u>crosstalk</u>, simultaneous switching outputs, and other regularly occurring interference signals.

# L

line Represents the value of one bit of a sample over all samples. A line is independent of any hardware I/O connector.

line A collection of lines displayed as a single plot on a digital group waveform graph.

line Name of a line within a sample or buffer.

name

- low For generation, the low level is the voltage produced when a
- level binary zero is generated. For acquisition, the low level is the voltage threshold below which the input will be sampled as a binary one.
- LSB least significant bit
- LVDM LVDM is an LVDS-compatible standard that allows for a 100 O parallel termination at the source.
- LVDS low voltage differential signaling. A low-noise, low-power, lowamplitude method for high-speed digital data transfer.

#### Μ

Marker event
 The Marker event is a general-purpose event that is configured within a generation script. The Marker event can be asserted synchronous to any even numbered sample within a waveform within a script.
 MB/s
 Unit for data transfer that means one million or 10⁶ bytes per second.
 Measurement
 Controlled, centralized configuration environment that allows you to configure all of your National Instruments

ExplorerDAQ, GPIB, HSDIO, IMAQ, IVI, Motion, and VISA(MAX)devices.

MSB most significant bit

- Ν
- NIM noise immunity margin—Extra margin between the voltage level being driven by the source and the voltage level required at the destination.

## 

On Board Clock	For NI Digital Waveform Generator/Analyzers, this term refers to the onboard voltage-controlled crystal oscillator (VCXO) clock source.
Onboard Reference clock	On PCI devices, the Onboard Reference Clock is the 10 MHz signal you can export to RTSI 7 on the RTSI trigger bus.
open collector	A drive type where the generation voltage high level is configured as the high-impedance state.
overshoot	Overshoot is a peak distortion following an edge.

#### Ρ

parallel termination	Termination that matches the characteristic impedance of the medium at the end of the transmission line.
Pause trigger	Trigger used to indicate to the device that it should stop generating and/or acquiring. The device resumes when the pause trigger becomes inactive.
PFI	Programmable Function Interface. I/O channels to the digital waveform generator/analyzer. Functionality and specifications will vary by device and operation.
pin number	See <u>terminal</u> .
posttrigger	Acquiring data that occurs after a trigger.
preshoot	Preshoot is a peak distortion preceding an edge.
pretrigger	Acquiring data that occurs before a trigger.
programmable function interface	See <u>PFI</u> .
propagation delay	The amount of time required for a signal to pass through a circuit.
protocol	The exact sequence of bits, characters and control codes used to transfer data between computers and peripherals through a communications channel, such as the GPIB.
PXI	PCI eXtensions for Instrumentation—Rugged, open system for modular instrumentation based on CompactPCI, with special mechanical, electrical, and software features.
PXI trigger bus	<b>PXI</b> equivalent of the <b>RTSI bus</b> , with additional timing and synchronization capabilities.

## R

Ready for Advance event	An event that indicates when the device enters its Wait for Advance Trigger state, which indicates that the acquisition of the previous record is complete.
Ready for Start event	For both acquisition and generation, the Ready For Start event indicates that the NI digital waveform/generator analyzer is configured and ready to receive a Start trigger.
real time	Property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.
Reference clock	Clock to which a device phase locks another, usually faster, clock. A common source for the reference clock is the 10 MHz oscillator present on the PXI backplane.
Reference trigger	This trigger establishes the reference point that separates pretrigger and posttrigger samples.
rise time	The time that it takes a signal to rise from 20% to 80% of the voltage between the voltage low level and the voltage high level.
round trip delay	Time required for the data to move from the digital tester, through the cable and DUT, and back to the tester.
R _{source}	source impedance
RTD	See <u>round trip delay</u> .
R _{term}	termination impedance
RTSI bus	Real-Time System Integration Bus—The National Instruments timing bus that connects DAQ devices directly, by means of connectors on top of the devices, for precise synchronization of functions.

S	
S	seconds
S	sample
sample	The value being generated/acquired on all of the digital data channels during a single sample clock cycle.
Sample clock	Samples are generated or acquired based on Sample clock cycles.
Sample Error event	An event that indicates when the device detects a sample where the actual response and the expected response do not match.
script	Collection of instructions that describe the order and timing of one or more waveforms.
Script trigger	General-purpose trigger that has a role that is determined by the context of the script.
series termination	Termination that places series impedance equal to the characteristic impedance at the source of the transmission line.
settling time	Time required for an amplifier, relay, or other circuits to reach a stable mode of operation.
signal	Means of conveying information. In this help file, signal refers to a digital transmission.
software trigger	Programmed event that triggers an operation such as data acquisition.
Start trigger	The Start trigger transitions a device into a state where the device can respond to Sample clocks. For an acquisition session, the device starts sampling and storing data. For a generation session, the device starts generating samples.
static acquisition	Software-timed (nonclocked) that returns the current digital logic state of the configured data channels with each read.
static generation	Software-timed (nonclocked) that sets the current state of the configured data channels to the requested digital logic state.
streaming	A method of generating waveforms that are too large to fit in device onboard memory by filling an allocated portion of onboard memory with the first part of the

# Т

terminal Named location where a signal is either produced (generated) or consumed (acquired).

t_{fall} <u>fall time</u>

t_{pd} propagation delay

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

trigger A signal sent to the device to control the device in some way. In the context of the NI digital waveform generator/analyzer, triggers are essentially the opposite of <u>events</u>.

t_{rise} <u>rise time</u>

t_s <u>settling time</u>

TTL transistor-transistor logic

#### U

unstrobed Basic digital I/O operations that do not involve the use of I/O control signals in data transfers. Unstrobed data transfers are controlled by software commands. Also known as software-timed I/O.

### V

V volts

VCXO voltage-controlled crystal oscillator

vector See <u>sample</u>.

- VHDCI very high-density connector interface
- VI Virtual Instrument
  - 1. A combination of hardware and/or software elements, typically used with a PC, that has the functionality of a classic stand-alone instrument.
  - 2. A LabVIEW software module (VI), which consists of a front panel user interface and a block diagram program.
- V_{IH} voltage input high—The input voltage level at or above which the logic device senses a binary one.
- V_{IL} Input voltage level at or below which the logic device senses a binary zero.

virtual Channel names that can be defined outside the application channels and used without having to perform scaling operations.

- V_{OD} differential output voltage—The difference in voltage between the positive and complementary conductors of a differential transmission. Can be thought of as the difference of the two conductors.
- V_{OH} voltage output high—The generated voltage level at the output pin when the logic device outputs a binary one.
- V_{OL} voltage output low—The generated voltage level at the output pin when the logic device outputs a binary zero.
- V_{OS} offset voltage—The common mode of the differential signal. Can be thought of as the average of the two conductors.
- V_{RANGE} input voltage range—The absolute voltage, referenced to common, allowed by the receiver.
- V_{TH} threshold voltage—the differential voltage threshold at which the receiver registers a valid logic state.

waveform A collection of digital samples generated or acquired at the DDC connector.

W

# Χ

 $\boldsymbol{x}$  Bit state meaning that the channel is ignored.

## Ζ

- z Bit state meaning that the channel is set to high-impedance.
- $Z_0$  The <u>characteristic AC impedance</u> of the transmission line.
- $Z_{\rm s}$  The impedance at the source of the transmission line.
- $Z_t$  The impedance at the destination of the transmission line.
# **Important Information**

Warranty Copyright Trademarks Patents Warning Regarding Use of NI Products

# Warranty

The NI 654*x*/655*x*/656*x* is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

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 <u>National Instruments trademarks</u>.

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

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

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

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

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

#### Patents

For patents covering National Instruments products, refer to the appropriate location: **Help**»**Patents** in your software, the patents.txt file on your CD, or <u>ni.com/patents</u>.

#### 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>, product <u>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 Developer</u> <u>Exchange</u> at ni.com/exchange. 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.

- Training and Certification—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- <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.
- Declaration of Conformity (DoC)—A DoC is our claim of compliance with the Council of the European Communities using the manufacturers declaration of conformity. This system affords the user protection for electronic compatibility (EMC) and product safety. You can obtain the DoC for your product by visiting ni.com/certification.
- <u>Calibration Certificate</u>—If your product supports calibration, you can obtain the calibration certificate for your product at

ni.com/calibration.

If you searched ni.com and could not find the answers you need, contact your <u>local office</u> or NI corporate headquarters. You also can visit the <u>Worldwide Offices</u> section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

# **Branch Offices**

Office	Telephone Number
Australia	1800 300 800
Austria	43 662 457990-0
Belgium	32 (0) 2 757 0020
Brazil	55 11 3262 3599
Canada	800 433 3488
China	86 21 5050 9800
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	358 (0) 9 725 72511
France	33 (0) 1 57 66 24 24
Germany	49 89 7413130
India	91 80 41190000
Israel	972 0 3 6393737
Italy	39 02 41309277
Japan	0120-527196
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