Interchangeable Virtual Instruments

IVI-4.15: IviDigitizer Class Specification

September 24, 2015 Edition
Revision 2.2

Important Information

The IVI-4.15: IviDigitizer Class Specification is authored by the IVI Foundation member companies. For a vendor membership roster list, please visit the IVI Foundation web site at www.ivifoundation.org.

The IVI Foundation wants to receive your comments on this specification. You can contact the Foundation through the web site at www.ivifoundation.org.

Warranty

Trademarks

Table of Contents

IviDigitizer Class Specification

IviDigitizer Revision History

This section is an overview of the revision history of the IviDigitizer specification.

Table 1. IviDigitizer Class Specification Revisions

Revision Number	Date of Revision	Revision Notes
Revision 1.0	October 20, 2009	First approved version of IVI-4.15 IviDigitizer class specification.
Revision 2.0	June 9, 2010	Incorporated IVI.NET
Revision 2.1	October 13, 2010	Editorial changes: section 12.3.1 Configure Sample Clock had wrong type for the source parameter.
Revision 2.2	October 14,2011	Editorial IVI.NET change. Change references to process-wide locking to AppDomain-wide locking. Add an overload to the Create factory method that takes locking related parameters. Editorial changes: - Section 4.1.4.1: Improve wording for better clarity. Minor changes: - Added section 4.3.3 Configure Active Trigger Source high level function (IVI-C only). - Section 11.2.3: add new attribute values for the Reference Oscillator Source for PXI/PXIe.
Revision 2.2	June 21, 2013	Editorial IVI.NET change. Correct spelling of PxiClk10 and PxiExpressClk100 .NET enumeration members to match the assembly. Change WindowConditionEnum to WindowCondition in Section 23.3.1, .NET Method Prototype.
Revision 2.2	October 16, 2013	Editorial IVI.NET change. Change all of the methods that use Iwaveform<Byte> to Iwaveform<Sbyte> instead.
Revision 2.2	September 24, 2015	Editorial Change – Clarified the use of one-based index for COM, and zero-based index for .NET for repeated capabilities in sections 4.2.5, 4.2.27, and 16.2.11.

API Versions

Architecture	Drivers that comply with version 2.2 comply with all of the versions below.
C	1.0, 2.0

COM	1.0, 2.0
.NET	2.0

Drivers that comply with this version of the specification also comply with earlier, compatible versions of the specification as shown in the table above. The driver may benefit by advertising that it supports all the API versions listed in the table above.

1. Overview of the IviDigitizer Specification

1.1 Introduction

This specification defines the IVI class for frequency digitizers. The IviDigitizer class is designed to support the typical digitizer as well as common extended functionality found in more complex instruments. This section summarizes the *IviDigitizer Class Specification* and contains general information that the reader might need in order to understand, interpret, and implement aspects of this specification. These aspects include the following:

- IviDigitizer class overview
- The definitions of terms and acronyms
- References

1.2 IviDigitizer Class Overview

This specification defines the IVI class for digitizers called IviDigitizer. The IviDigitizer class is designed to support the typical digitizer as well as common extended functionality found in more complex instruments. The IviDigitizer class conceptualizes a digitizer as an instrument that can acquire time varying voltage waveforms.

The IviDigitizer class is divided into the base capability group and extensions. The base capability group functions and attributes are used to configure a digitizer for typical waveform acquisition (this includes setting the channel, the acquisition, and the triggering sub-systems), initiating the waveform acquisition, and returning a waveform. The base capability group support only edge triggering. The IviDigitizerBase Capabilities are described in Section 4.

In addition to the base capabilities, the IviDigitizer class defines extended capabilities for digitizers that can:

- Combine channels for higher acquisition rates and longer waveform records
- Have advanced triggering options such as TV, runt, glitch, width, and Window
- Retrieve time-stamped data
- Use an external frequency reference
- Report device and channel temperatures
- Trigger or arm on multiple sources

The IviDigitizer extended capabilities are arranged into a set of extension capability groups.

1.3 References

Several other documents and specifications are related to this specification. These other related documents are as follows:

IVI-3.1: Driver Architecture Specification

IVI-3.2: Inherent Capabilities Specification

IVI-3.3: Standard Cross Class Capabilities Specification

IVI-3.18: IVI.NET Utility Classes and Interfaces Specification

IVI- 5.0: Glossary

1.4 Definitions of Terms and Acronyms

Refer to *IVI-5.0: Glossary* for a description of the terms and acronyms used in this specification. This specification does not define any additional terms.

2. IviDigitizer Class Capabilities

2.1 Introduction

The IviDigitizer specification divides digitizer capabilities into a base capability group and multiple extension capability groups. Each capability group is discussed in a separate section. This section defines names for each capability group and gives an overview of the information presented for each capability group.

2.2 IviDigitizer Group Names

The capability group names for the IviDigitizer class are defined in the following table. The Group Name is used to represent a particular capability group and is returned as one of the possible group names from the Class Group Capabilities attribute.

Table 2-1. IviDigitizer Group Names

Group Name	Description
`IviDigitizerBase`	Base Capabilities of the IviDigitizer specification. This group includes the capability to acquire waveforms using edge triggering.
`IviDigitizerMultiRecordAcquisition`	Extension: IviDigitizer with the ability to do multi-record acquisitions.
`IviDigitizerBoardTemperature`	Extension: IviDigitizer with the ability to report the temperature of the digitizer.
`IviDigitizerChannelFilter`	Extension: IviDigitizer with the ability to control the channel input filter frequency.
`IviDigitizerChannelTemperature`	Extension: IviDigitizer with the ability to report the temperature of individual digitizer channels.
`IviDigitizerTimeInterleavedChannels`	Extension: IviDigitizer with the ability to combine two or more input channels to achieve higher acquisitions rates and/or record lengths.
`IviDigitizerDataInterleavedChannels`	Extension: IVIdigitizer with the ability to interleave the data from two or more input channels, usually to create complex (I/Q) data.
`IviDigitizerReferenceOscillator`	Extension: IviDigitizer with the ability to use an external reference oscillator.
`IviDigitizerSampleClock`	Extension: IviDigitizer with the ability to use an external sample clock.
`IviDigitizerSampleMode`	Extension: IviDigitizer with the ability to control whether the digitizer is using real-time or equivalent-time sampling.
`IviDigitizerSelfCalibration`	Extension: IviDigitizer with the ability to perform self calibration.
`IviDigitizerDownconversion`	Extension: IviDigitizer with the ability to do frequency translation or downconversion in hardware.
`IviDigitizerArm`	Extension: IviDigitizer with the ability to arm on positive or negative edges.

Table 2-1. IviDigitizer Group Names

Group Name	Description
`IviDigitizerMultiArm`	Extension: IviDigitizer with the ability to arm on one or more sources.
`IviDigitizerGlitchArm`	Extension: IviDigitizer with the ability to arm on glitches.
`IviDigitizerRuntArm`	Extension: IviDigitizer with the ability to arm on runts.
`IviDigitizerSoftwareArm`	Extension: IviDigitizer with the ability to arm acquisitions.
`IviDigitizerTVArm`	Extension: IviDigitizer with the ability to arm on standard TV signals.
`IviDigitizerWidthArm`	Extension: IviDigitizer with the ability to arm on a variety of conditions regarding pulse widths.
`IviDigitizerWindowArm`	Extension: IviDigitizer with the ability to arm on signals entering or leaving a defined voltage range.
`IviDigitizerTriggerModifier`	Extension: IviDigitizer with the ability to perform an alternative triggering function in the event that the specified trigger event doesn't occur.
`IviDigitizerMultiTrigger`	Extension: IviDigitizer with the ability to trigger on one or more sources.
`IviDigitizerPretriggerSamples`	Extension: IviDigitizer with the ability to specify a number of samples to fill up the data buffer with pre-trigger data.
`IviDigitizerTriggerHoldoff`	Extension: IviDigitizer with the ability to specify a length of time after the digitizer detects a trigger during which the digitizer ignores additional triggers.
`IviDigitizerGlitchTrigger`	Extension: IviDigitizer with the ability to trigger on glitches.
`IviDigitizerRuntTrigger`	Extension: IviDigitizer with the ability to trigger on runts.
`IviDigitizerSoftwareTrigger`	Extension: IviDigitizer with the ability to trigger acquisitions.
`IviDigitizerTVTrigger`	Extension: IviDigitizer with the ability to trigger on standard television signals.
`IviDigitizerWidthTrigger`	Extension: IviDigitizer with the ability to trigger on a variety of conditions regarding pulse widths.
`IviDigitizerWindowTrigger`	Extension: IviDigitizer with the ability to trigger on signals entering or leaving a defined voltage range.

2.3 Repeated Capability Names

The IviDigitizer Class Specification defines three repeated capabilities. Refer to the sections of *IVI-3.1, Driver Architecture Specification* that deal with repeated capabilities. The relevant sections are Section 2.7, *Repeated Capabilities*, Section 4.1.9, *Repeated Capabilities*, Section 4.2.5, *Repeated Capabilities*, Section 4.3.9, *Repeated Capabilities*, and Section 5.9, *Repeated Capability Identifiers and Selectors*.

- Channel
- ArmSource
- TriggerSource

2.3.1 Channel

In the configuration store, the name for the Channel repeated capability shall be exactly one of “Channel” or “IviDigitizerChannel”. Drivers that implement multiple repeated capabilities with the name “Channel” shall use the latter form to disambiguate the name.

2.3.2 ArmSource

In the configuration store, the name for the ArmSource repeated capability shall be exactly one of “ArmSource” or “IviDigitizerArmSource”. Drivers that implement multiple repeated capabilities with the name “ArmSource” shall use the latter form to disambiguate the name.

2.3.3 TriggerSource

In the configuration store, the name for the TriggerSource repeated capability shall be exactly one of “TriggerSource” or “IviDigitizerTriggerSource”. Drivers that implement multiple repeated capabilities with the name “TriggerSource” shall use the latter form to disambiguate the name.

2.4 Boolean Attribute and Parameter Values

This specification uses True and False as the values for Boolean attributes and parameters. The following table defines the identifiers that are used for True and False in the IVI.NET, IVI-COM, and IVI-C architectures.

Boolean Value	IVI.NET Identifier	IVI-COM Identifier	IVI-C Identifier
True	`true`	`VARIANT_TRUE`	`VI_TRUE`
False	`false`	`VARIANT_FALSE`	`VI_FALSE`

2.5 .NET Namespace

The .NET namespace for the IviDigitizer class is `Ivi.Digitizer`.

2.6 .NET IviDigitizer Session Factory

The IviDigitizer .NET assembly contains a factory method called Create for creating instances of IviDigitizer class-compliant IVI.NET drivers from driver sessions and logical names. Create is a static method accessible from the static IviDigitizer class.

Refer to *IVI-3.5: Configuration Server Specification* for a description of how logical names and session names are defined in the configuration store.

Refer to Section 8, *IVI.NET Specific Driver Constructor*, of *IVI-3.2: Inherent Capabilities Specification*, for more details on how the `idQuery`, `reset`, and `options` parameters affect the instantiation of the driver.

Refer to Section 4.3.11, *Multithread Safety*, of *IVI-3.1: Driver Architecture Specification* for a complete description of IVI.NET driver locking. Refer to Section 8, Table 8.2 *Required Lock Type Behavior for Drivers With the Same Access Key*, of *IVI-3.2, Inherent Capability Specification*, for an explanation of how the values for `lockType` and `accessKey` are used to determine the kind of multithreaded lock to use for the driver instance.

.NET Method Prototype

```
IIviDigitizer Ivi.Digitizer.Create(String name);
IIviDigitizer Ivi.Digitizer.Create(String name,
                                   Boolean idQuery,
                                   Boolean reset);
IIviDigitizer Ivi.Digitizer.Create(String name,
                                   Boolean idQuery,
                                   Boolean reset,
                                   String options);

IIviDigitizer Ivi.Digitizer.Create(String resourceName,
                                   Boolean idQuery,
                                   Boolean reset,
                                   LockType lockType,
                                   String accessKey,
                                   String options);
```

Parameters

Inputs	**Description**	**Base Type**
`name`	A session name or a logical name that points to a session that uses an IVI.NET IviDigitizer class-compliant driver.	`String`
`idQuery`	Specifies whether to verify the ID of the instrument. The default is False.	`Boolean`
`reset`	Specifies whether to reset the instrument. The default is False.	`Boolean`
`lockType`	Specifies whether to use AppDomain-wide locking or machine-wide locking.	`Ivi.Driver.LockType`
`accessKey`	Specifies a user-selectable access key to identify the lock. Driver instances that are created with the same accessKey will be protected from simultaneous access by multiple threads within an AppDomain or across AppDomains, depending upon the value of the lockType parameter.	`String`
`options`	A string that allows the user to specify the initial values of certain inherent attributes. The default is an empty string.	`String`

Outputs	**Description**	**Base Type**
`Return Value`	Interface reference to the IIviDigitizer interface of the driver referenced by `session`.	`IIviDigitizer`

Defined Values

Name	*Description*		
		Language	*Identifier*
AppDomain	The lock is AppDomain-wide.		
		.NET	`Ivi.Driver.LockType.AppDomain`
Machine	The lock is machine-wide.		
		.NET	`Ivi.Driver.LockType.Machine`

.NET Exceptions

The *IVI-3.2: Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Usage

To create a driver that implements the IviDigitizer instrument class API from the logical name "My LogicalName", use the following:

```
IIviDigitizer counter = IviDigitizer.Create("MyLogicalName");
```

In this case, the ID of the instrument will not be verified, the instrument will not be reset, and options will be supplied from the configuration store and/or driver defaults.

3. General Requirements

This section describes the general requirements a specific driver must meet in order to be compliant with this specification. In addition, it provides general requirements that specific drivers must meet in order to comply with a capability group, attribute, or function.

3.1 Minimum Class Compliance

To be compliant with the IviDigitizer Class Specification, an IVI specific driver shall conform to all of the requirements for an IVI class-compliant specific driver as specified in *IVI-3.1: Driver Architecture Specification*, implement the inherent capabilities that *IVI-3.2: Inherent IVI Capabilities Specification* defines, and implements the IviDigitizerBase capability group.

3.1.1 Disable

Refer to *IVI-3.2: Inherent Capabilities Specification* for the prototype of this function. The IviDigitizer specification does not define additional requirements on the Disable function.

3.2 Capability Group Compliance

IVI-3.1: Driver Architecture Specification defines the general rules for a specific driver to be compliant with a capability group.

4. IviDigitizerBase Capability Group

4.1 Overview

The IviDigitizer base capabilities support digitizers that can acquire waveforms from multiple channels with an edge trigger. The IviDigitizer base capabilities define attributes and their values to configure the digitizer's channel, acquisition, and trigger sub-systems. The IviDigitizer base capabilities also include functions for configuring the digitizer as well as initiating waveform acquisition and retrieving waveforms.

The IviDigitizer base capabilities organize the configurable settings into three main categories: the channel sub-system, the acquisition sub-system, and the trigger sub-system.

4.1.1 Channel Sub-System

The channel sub-system configures the range of voltages the digitizer acquires and how the digitizer couples the input signal to the acquisition sub-system. The main channel sub-system attributes include:

- Channel Enabled
- Vertical Coupling
- Vertical Range
- Vertical Offset

All of the channel sub-system attributes represent capabilities that are repeated on each of an instrument's channels. They can be set as a group with the Configure Channel function.

For each channel, the *Channel Enabled* attribute specifies whether the digitizer acquires a waveform for that channel. This attribute may be used to simply disable an unused channel, but will be more useful when using the capabilities in extension groups for combining channels (for increased sample rate) or allocating memory (so an unused channel's memory will become available to other active channels). For instance, if two channels have been combined into one, one of those channels must be disabled. This attribute can later be read to determine which channel is active and which is not.

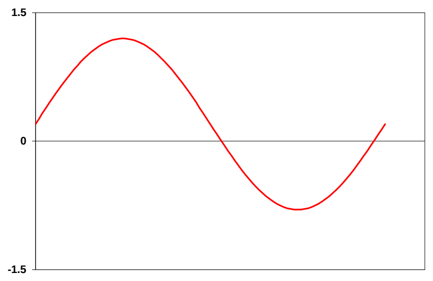
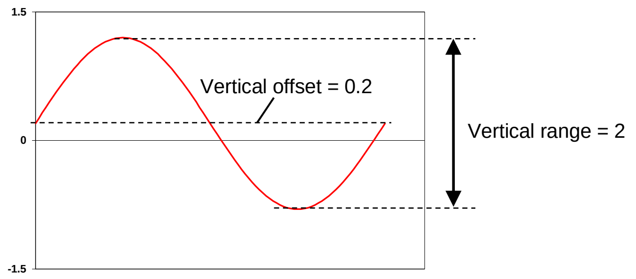
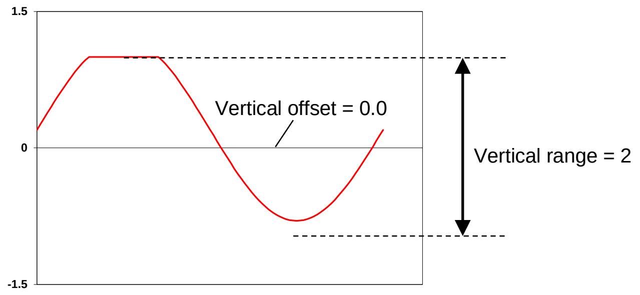
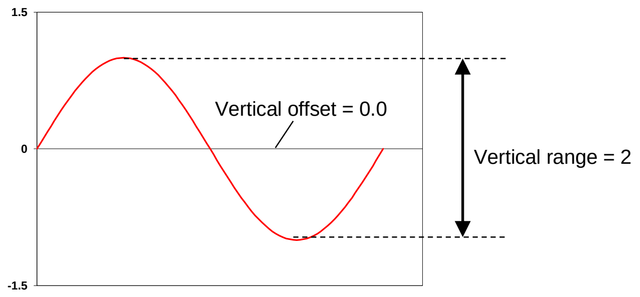
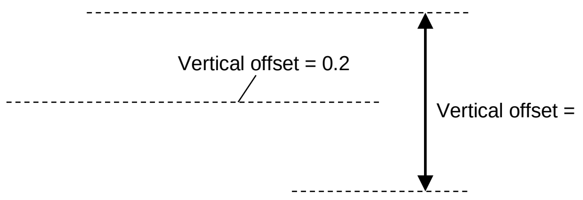
The *Vertical Coupling* attribute specifies how to couple the input signal to the channel sub-system. Most digitizers will support AC coupling (which blocks any DC component of the input signal) and DC coupling (which allows the entire signal, including any DC offset, to be sampled).

The *Vertical Range* attribute specifies the absolute value of the range of voltages that the digitizer acquires. This range is specified as a peak-to-peak voltage number. The value of the *Vertical Offset* parameter determines the center of this range.

The *Vertical Offset* attribute specifies the center of the range specified by the *Vertical Range* attribute with respect to ground.

It is important for these parameters to be set properly before a signal is measured. Digitizers perform best if the input signal is scaled to a range that matches the physical measurement range of the analog-to-digital conversion hardware. Otherwise, clipping of the signal may occur or quantization noise may increase. Table 4-1displays some relevant examples.

Table 4-1. Examples of sampled data versus channel property settings

Actual applied signal (sinewave with 1-volt amplitude and 0.2-volt DC offset)	
Measured data with: • Vertical offset = 0.2 • Vertical range = 2 • Coupling = DC	
Measured data with: • Vertical offset = 0 • Vertical range = 2 • Coupling = DC	
Measured data with: • Vertical offset = 0 • Vertical range = 2 • Coupling = AC	
Measured data with: • Vertical offset = 0.2 • Vertical range = 2 • Coupling = AC	

4.1.2 Acquisition Sub-System

The acquisition sub-system configures the number of waveform records, the size of each record, and the sample rate. The configurable Acquisition sub-system attributes include:

- Sample Rate
- Num Records
- Record Size

While very simple digitizers may support only a single sample rate and can record only a single record, more advanced digitizers can vary the sample rate and can collect multiple independent sets of data with a single acquisition, with the ability to flexibly allocate memory resources as needed.

The *Sample Rate* attribute controls the data sample rate, in number of samples per second. Some digitizers may have the ability to accept an arbitrary value for this attribute (up to a given limit), while others may support only one possible value or a list of discrete values. If a value is specified that the digitizer cannot support, the value is coerced to the next highest sample rate that can be supported. If a value is specified that is higher than the maximum sample rate available in the digitizer, an error will be returned.

Note that some digitizers have the ability to combine multiple channels together, resulting in a single channel with twice the maximum sample rate. In the IviDigitizer interface, this feature is controlled with the *IviDigitizerTimeInterleavedChannels* extension group. The *Sample Rate* attribute is sensitive to this feature. If channels are combined, the maximum value for *Sample Rate* will increase accordingly.

The *Num Records* and *Record Size* attributes control the number and size of distinct data records that a digitizer will capture for a single acquisition. An acquisition begins with a call to the *Initiate Acquisition* function. Generally speaking, when an acquisition is initiated the digitizer will wait for a trigger event to occur (see section 4.1.4 for a description of the trigger sub-system) and will then record sampled data in a memory buffer. Each such data set is referred to as a record. Simple digitizers will be able to collect only a single record for each acquisition. More sophisticated digitizers will have the ability to collect multiple data records in a single acquisition. This feature requires multiple trigger events. Before the acquisition begins, the *Num Records* attribute determines the number of records to capture, and the *Record Size* attribute determines the size of each record. When acquisition begins the digitizer fills up a record and waits for another trigger event before proceeding to fill up the next record. The process continutes until *Num Records* have been captured, at which time the digitizer will return to an idle state. To retrieve data from a multi-record acquisition, the Fetch functions from the IviDigitizerMultiRecordAcquisition extension group (section 5) must be used.

When capturing data, it is up to the user to manage the digitizer's memory. Some digitizers have fixed memory configurations while others may have global memory blocks that can be allocated as needed. Each active channel will capture *Num Records* records. The total amount of memory required for this depends on the number of records, the size of each record, and the number of active channels.[1] If the user attempts to allocate more memory than the digitizer has available, the driver will return an error.

The Fetch functions from the IviDigitizerMultiRecordAcquisition extension group (section 5) can also be used for fetching partial records. This is useful for example when the digitizer memory is so large that the data must be fetched in chunks.

[1] Note that channels may become inactive for more than one reason – they can be intentionally turned off because they are unused, or they may be combined with other channels to obtain a higher sample rate. In either case there may be an effect on the amount of memory available, depending on the capabilities of the specific digitizer.

4.1.3 IviDigitizer Arm and Trigger States

The IviDigitizer interface supports both arming and triggering states in digitizers. For digitizers that support it, the full state diagram is shown in Figure 4-1.

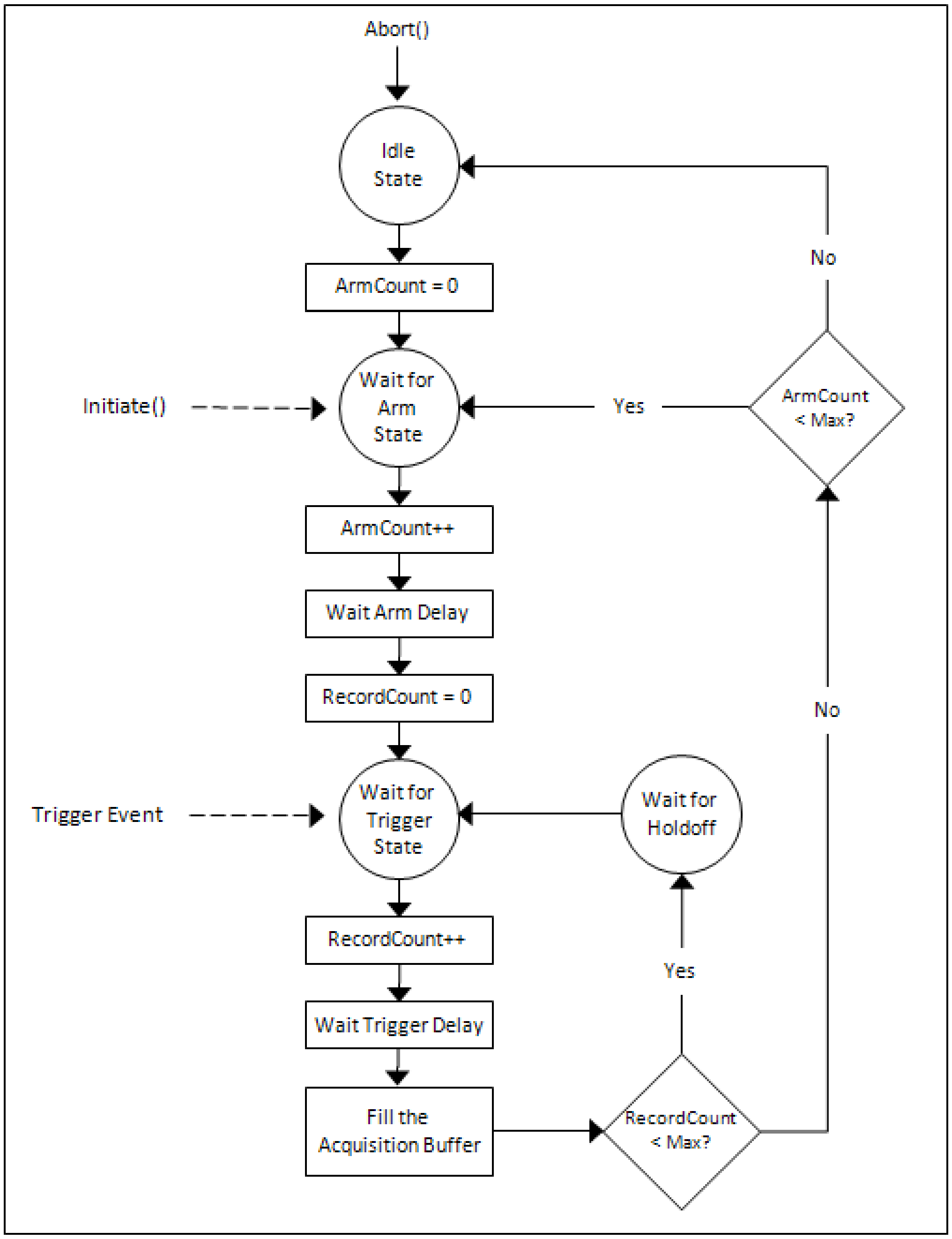


Figure 4-1 IviDigitizer Arm/Trigger Model

Digitizers are not required to support all of the possible states. In particular, many digitizers do not support arming. However, for digitizers that do support it, the IviDigitizer interface includes both a Trigger sub-system and an Arm sub-system, both of which work in the same fashion.

4.1.4 Trigger Sub-System

The trigger sub-system configures the type of event that triggers the digitizer. The global trigger subsystem attributes are:

- Active Trigger Source
- Trigger Delay
- Trigger Holdoff
- Trigger Modifier
- Pretrigger Samples

The *Active Trigger Source* attribute (a string) specifies which of the available trigger sources is to be used.

The *Trigger Delay* attribute attribute specifies the position of the first point in the captured waveform record relative to the Trigger Event. If the *Trigger Delay* value is positive, the first point in the waveform record occurs after the trigger event. If the value is negative, the first point in the waveform record occurs before the trigger event. Note that this specification uses the term *Trigger Delay* instead of the *Acquisition Start Time* terminology employed by the Iviscope specification. In typical systems, these represent the same quantity.

The *Trigger Holdoff* attribute specifies the length of time after the digitizer detects a trigger during which the digitizer ignores additional triggers. The *Trigger Holdoff* attribute affects the instrument operation only when the digitizer is configured to acquire multiple records (the *Num Records* attribute is greater than 1). If a trigger event occurs while the digitizer is still filling up a record, it will be ignored even if the *Trigger Holdoff* attribute is set to zero. This attribute is identical to the *Trigger Holdoff* attribute in the IviScope interface.

The *Trigger Modifier* attribute specifies the digitizer's behavior in the absence of the configured trigger. This attribute is identical to the *Trigger Modifier* attribute in the IviScope interface.

The *Pretrigger Samples* attribute is defined as the number of samples needed to fill up the data buffer with pre-trigger data. This attribute is used to capture as much pre-trigger data as possible without losing important events.

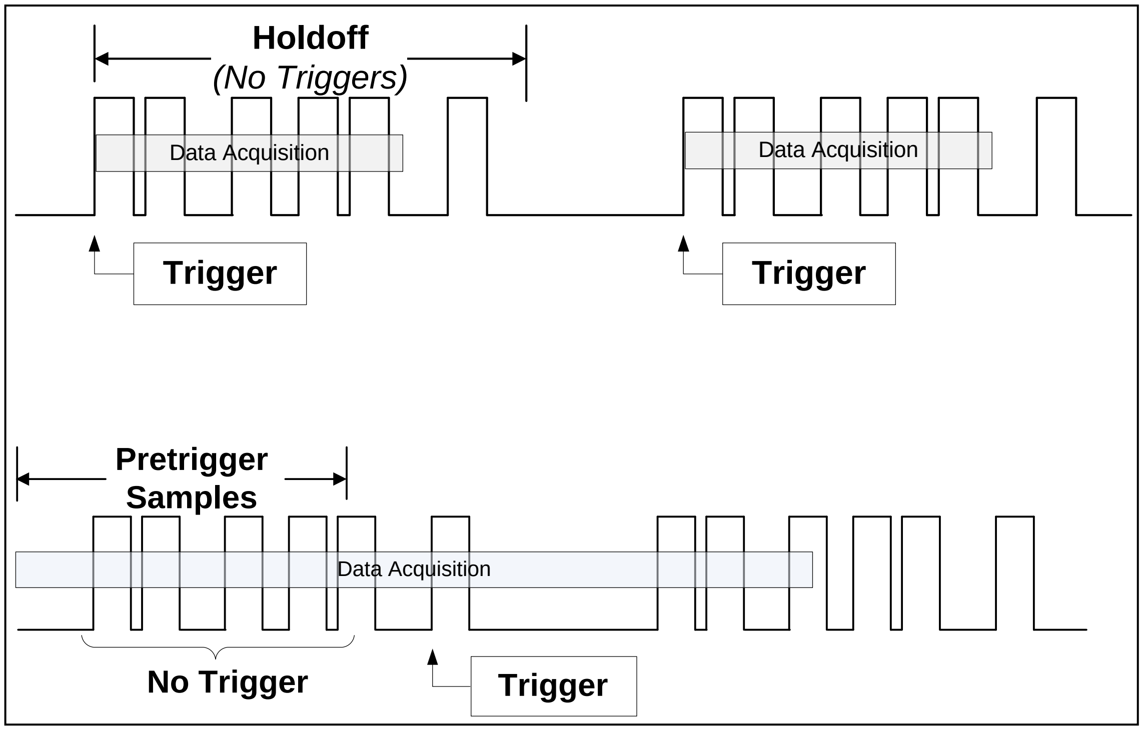
While the Pretrigger Samples attribute and the Trigger Holdoff attribute can be used to obtain similar results, these attributes actually perform quite different functions and are used for different applications.

- Trigger Holdoff is measured from the time of a trigger event, and is specified in units of time (seconds). It is used in situations where the user is attempting to trigger on a part of a long repeating signal. When the data capture buffer is filled, the Trigger Holdoff attribute can be used to prevent the digitizer from triggering again until the sequence repeats itself. Trigger Holdoff is measured starting from the time of the trigger.

- Pretrigger Samples is used to gather as much pre-trigger data as possible, without missing an important event. Pretrigger Samples is measured in number of samples and is the minimum number of samples that must be recorded before the digitizer will respond to a trigger event. This guarantees that some data will be recorded before the trigger event occurs. The counting of pretrigger samples begins when the user calls the Initiate Acquisition function. When recording multiple records, the counting of pretrigger samples begins after each prior record is filled and when the digitizer is able to collect a new data record, i.e., at the end of the preceding record plus any re-arm time that the digitizer may require.

Trigger Holdoff and Pretrigger Samples are *not* equivalent, although they do perform similar functions. The difference lies in subtle timing relationships between these parameters and the trigger event. Trigger Holdoff is measured starting from the time of a trigger event, while Pretrigger Samples is measured from the time that the digitizer is ready to capture a new record (the time recording the previous record is finished plus the re-arm time, if any). This means that Pretrigger Samples timing is always synchronous with the sample clock. Trigger Holdoff is *not* synchronous with the sample clock, since the actual trigger event may occur at any time.

Error! Reference source not found. below illustrates the relationship between *Trigger Holdoff* and *Pretrigger Samples*. The upper half of the figure illustrates *Trigger Holdoff* for an acquisition with *Num Records* greater than one. Here, the second (and subsequent) triggers will not be accepted until the data acquisition is finished *and* the holdoff period has expired. The lower half illustrates *Pretrigger Samples* and shows how trigger events will not be accepted until a sufficient number of samples has been collected in the digitizer's data buffer.



Vertical range = 2

4.1.4.1 Setting Up Triggers

The process for setting triggers in the IviDigitizer interface is simple, but the IviDigitizer trigger subsystem works differently from triggers in the IviScope interface. In IviDigitizer, triggers are a repeated capability. Trigger parameters like the *Trigger Coupling* and *Trigger Level* attributes are associated with each trigger source – they are not global parameters as in the IviScope interface. The interface includes an *Active Trigger Source* attribute (a string) that is used to specify which of the available trigger sources is to be used.

To set up a trigger, simply follow these steps:

1. Set the *Active Trigger Source* attribute. This is a string that specifies a particular trigger source, such as "Channel 1". Trigger sources are a repeated capability, and the valid trigger source names can be discovered by traversing the TriggerSource repeated capability list.

2. Set the *Trigger Type* attribute. This attribute is specific to each Trigger Source. Like all other Trigger Source attributes, setting this attribute applies **only** to the specified trigger source repeated capability. If the Active Trigger Source is changed to another trigger source, the (active) trigger type may change as well.

3. Refer to this document or to the vendor's document to determine what other parameters need to be set for the given *Trigger Type*. For instance, if using an edge trigger then the *Trigger Level* and *Trigger Slope* attributes should be set. Different trigger types have different requirements.

Each trigger source has the following attributes:

- Trigger Coupling
- Trigger Hysteresis
- Trigger Level
- Trigger Type

The *Trigger Coupling* attribute specifies how the digitizer couples the trigger source to the trigger sub-system. Commonly, this attribute will be set to DC (which allows the trigger signal to be used without modification) or AC (which strips away any DC component of the trigger signal). The IviDigitizer interface also includes a number of more complex coupling types that mirror the capabilities in the IviScope interface.

The *Trigger Hysteresis* attribute specifies the trigger hysteresis in Volts.

The *Trigger Level* attribute specifies the voltage threshold for the trigger sub-system.

The *Trigger Type* attribute specifies the type of event that triggers the digitizer. The most common type of trigger is the edge trigger, which causes the instrument to trigger whenever a signal passes through a predefined voltage level. The IviDigitizer interface mirrors the IviScope interface, and includes a number of complex trigger types that are supported identically in both IviDigitizer and IviScope. For each supported trigger type, the IviDigitizer interface includes a number of special-purpose attributes that are used to control the triggering operation.

For digitizers that support it, the IviDigitizer interface includes the ability to specify arming conditions for triggering. Arming is set up in the same fashion as triggering, using the IviDigitizerArm extension group.

The IviDigitizer interface also supports the use of multiple trigger sources for a single acquisition. For digitizers that support it, this feature is controlled with the IviDigitizerMultiTrigger extension group. This allows multiple trigger sources to be specified. The trigger sources can be AND'ed together (so that all triggers must occur before the instrument to make a measurement) or OR'ed together (so that the first-occurring trigger event causes the instrument to make a measurement).

4.1.4.2 Configuring Edge Triggers

The following attributes configure the edge trigger. These attributes can be set as a group with the Configure Edge Trigger Source function.

- Trigger Level
- Trigger Source
- Trigger Slope

The *Trigger Level* attribute specifies the voltage threshold for the trigger sub-system. The *Trigger Source* attribute specifies the source the digitizer monitors for the trigger event. Most of the trigger types use the values held in the *Trigger Level* and *Trigger Source* attributes.

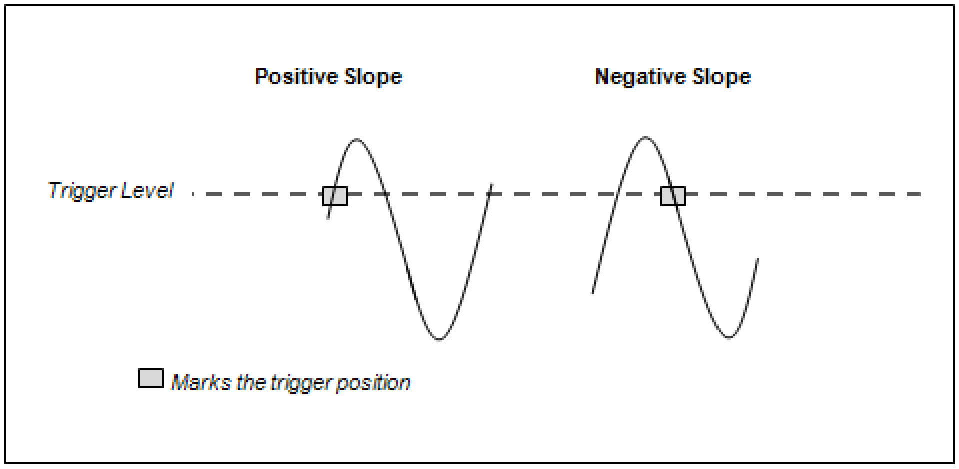


Figure 4-2 Edge Triggers

The *Trigger Slope* attribute specifies whether a positive or negative edge triggers the digitizer.

When the trigger type is edge, the values held in the *Trigger Level*, *Trigger Source*, and *Trigger Slope* attributes define the trigger event. The digitizer triggers when the signal from the trigger source crosses the threshold level with the polarity that the *Trigger Level* and *Trigger Coupling* attributes specify.

4.1.4.3 Software Triggers

One type of trigger source is "software". Setting the *Active Trigger Source* attribute to Software means that the instrument will only trigger when given a software command. To execute a software trigger, use the SendSoftwareTrigger function.

If the SendSoftwareTrigger function is called when the *Active Trigger Source* attribute is not set to Software, an error will be returned.

For digitizers that support arming, the *Active Arm Source* can also be set to Software. In this circumstance the SendSoftwareArm command is used to arm the instrument.

4.1.4.4 Immediate Triggers

Another type of trigger is "immediate". Setting the *Active Trigger Source* attribute to Immediate means that the instrument will trigger immediately, without waiting for any event.

The Trigger Type attribute has no effect on Immediate triggers.

If using multiple triggers, Immediate cannot be one of the specified trigger sources. Doing so will return an error.

4.1.4.5 Using Disabled Channels as Trigger Sources

Some digitizer implementations include the ability to trigger on signals in disabled channels. Although data is not available from disabled channels, they may still be available for use as trigger sources. The IviDigitizer interface supports this functionality but does not require it. Users should refer to the manufacturer's documentation to learn more about this capability in any given digitizer.

4.1.5 Reading Data

The IviDigitizerBase Capabilities define functions that retrieve waveforms from the digitizer. These functions return the following information:

- The waveform record as an array of voltages.
- The time that corresponds to the first point in the waveform array relative to the trigger event (the *InitialXOffset)*
- The first valid data point and the number of valid data points in the waveform record.

The true time reference for a captured waveform is the **Trigger Event**, *not* the sampling times, because the trigger event occurs asynchronously with respect to the sampling clock. Thus, the time between the trigger event and the next sampling clock pulse varies randomly in the range [0 … sampling interval]. When reading data, one value that is returned is the *InitialXOffset*, which specifies the position of the first point in the waveform record relative to the trigger event. If the value is positive, the first point in the waveform record occurs *after* the trigger event. If the value is negative, the first point in the waveform record occurs *before* the trigger event. Figure 4-3 hows an example of an acquisition with a positive acquisition start time (positive *Trigger Delay* attribute) while Figure 4-4 shows the effect of the negative acquisition start time value (negative *Trigger Delay)*. Note that both figures show *Trigger Holdoff*, which only applies when multiple records are being captured.

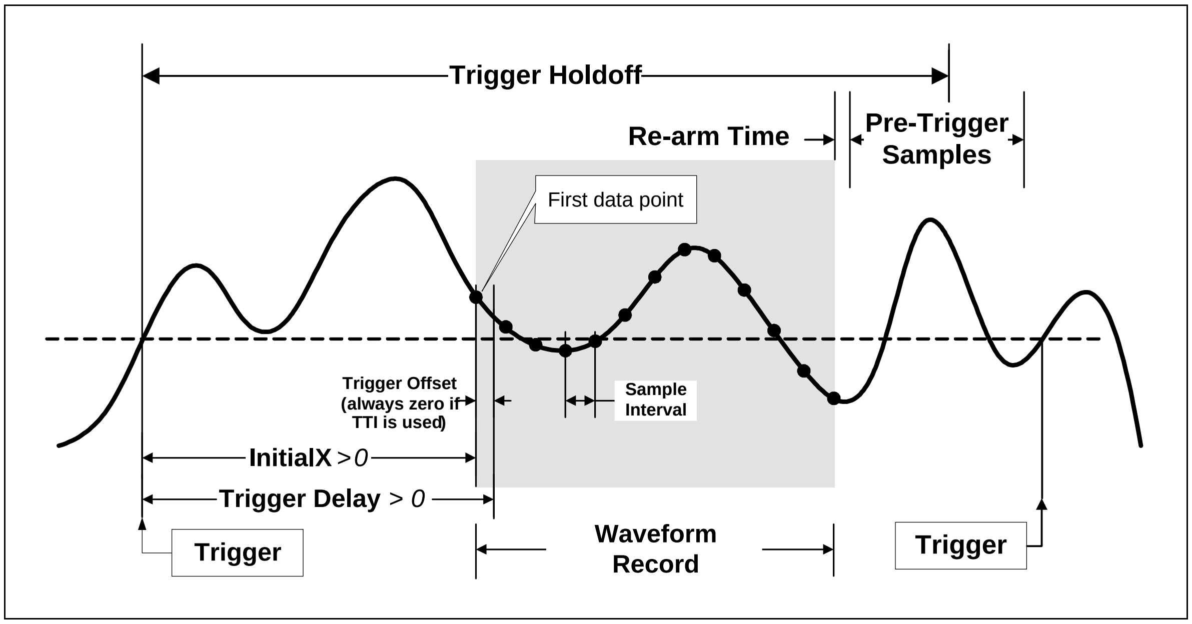


Figure 4-3 Positive Trigger Delay

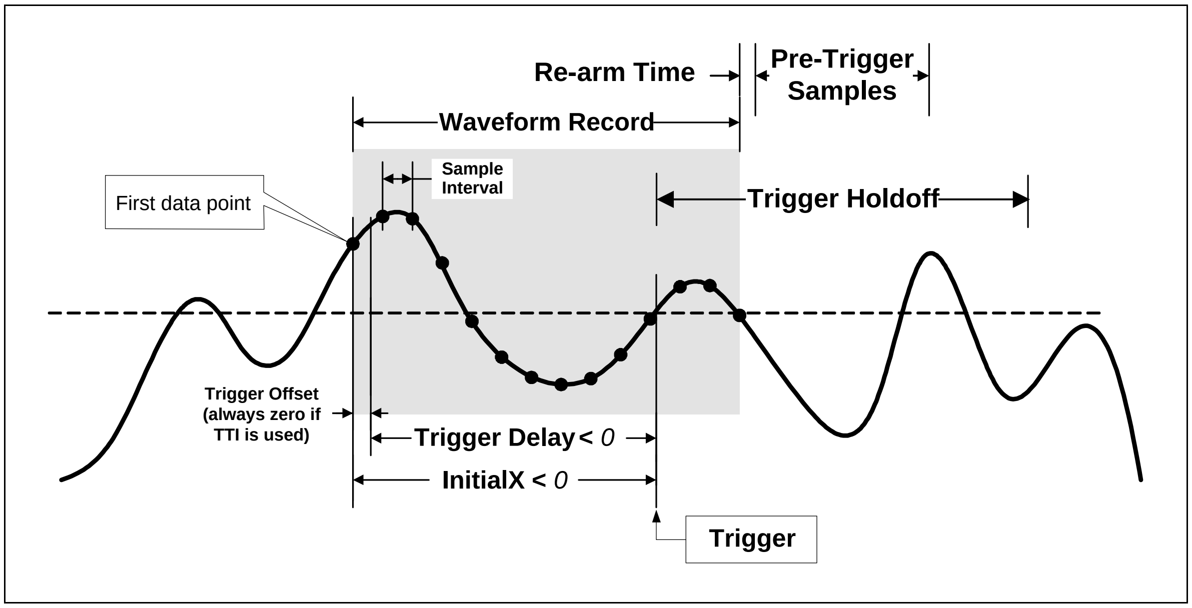


Figure 4-4 Negative Trigger Delay

The **Trigger Offset** value shown in these figures measures the time between the trigger event and the first captured data point. Some digitizers feature a **Trigger Time Interpolator** (TTI) which shifts all the samples in time using interpolation in order to align the first sample with the start of the acquisition. In such a case, the value of InitialX is always exactly equal to the acquisition start time (and it will match the **Trigger Delay)**.

The **Re-arm Time** value shown in these figures is an important characteristic of some digitizers. These digitizers are not able to start a new acquisition immediately following the preceding acquisition – they require some time for memory management and other internal functions. The re-arm time is not controllable by the user and is not addressed in the IviDigitizer interface, but it is important to consider when writing software to control digitizers.

4.1.5.1 Allocating Waveform Buffer Memory

In Read and Fetch calls, it is important to allocate sufficient amounts of memory for the waveform buffer. If the waveform buffer is not large enough to hold all of the requested data, the driver will fill the buffer as fully as possible and return the number of points that were actually retrieved in the function call's *ActualPoints* parameter. To retrieve the remainder of the data, one or more additional Fetch calls must be made. Although the IviDigitizer driver supports this operation, it results in a performance degradation that is unnecessary if enough memory is available.

The amount of memory needed may be greater than one might expect. To maximize data transfer rates, some digitizers are designed to perform DMA transfers at defined memory boundaries. Depending on the alignment of the data buffer, this can result in a few "garbage" data points at the beginning of the buffer. (These invalid points are identified by means of the *FirstValidPoint* parameter in Read and Fetch function calls.) Other digitizers may make temporary use of extra buffer memory to increase performance when retrieving multiple records. In the case of multi-record acquisitions, the digitizer's memory is divided into segments, one for each record. Some digitizers manage their internal memory by pages of fixed size, with the constraint that a segment must start at the beginning of a page. Readout may also be constrained by entire pages, and may be further complicated when time- or data-interleaving of channels is used. All these constraints can introduce significant overhead in the amount of memory that must be reserved for each record, especially when the number of points per record is small. Finally, for digitizers allowing the capture of pre-trigger data, the memory is used as a circular buffer continuously recording samples, until the acceptance of a valid trigger defines the actual beginning of the record. In this case, the superfluous samples may be located anywhere in the segment, so that a reordering of the samples by the driver must be performed. An additional segment of memory allows performing this operation more efficiently.

An example of the memory after a 5-record acquisition with a digitizer using circular buffers, and requiring segment "padding" and reordering is shown in Figure 4-5.

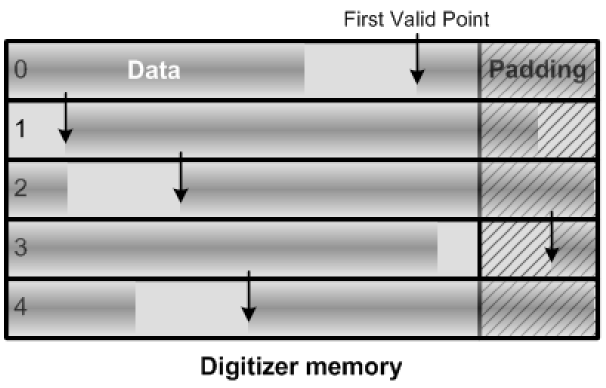


Figure 4-5 Digitizer memory immediately after a multi-record acquisition

The best readout performance (especially for small record sizes) is usually obtained by transferring the whole acquisition at once, and to perform the reordering on the host computer. To be able to easily perform this reordering of each record, a digitizer manufacturer may choose to require the waveform buffer to be larger than the digitizer memory by one segment, as shown in Figure 4-6.

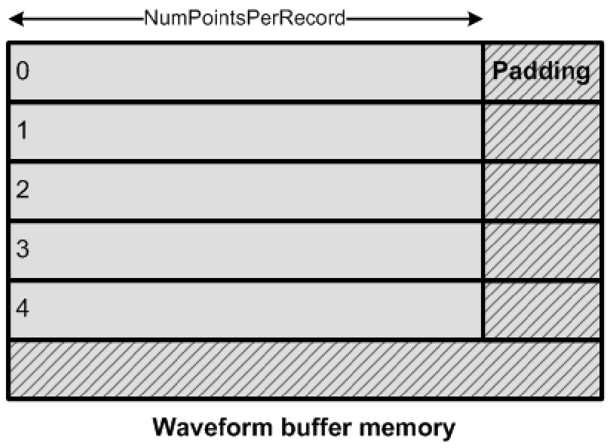


Figure 4-6 Waveform buffer memory for multi-record readout

The whole acquisition is transferred as depicted in Figure 4-7 (note that this is an intermediate state), and the reordering can then be performed as shown in Figure 4-8.

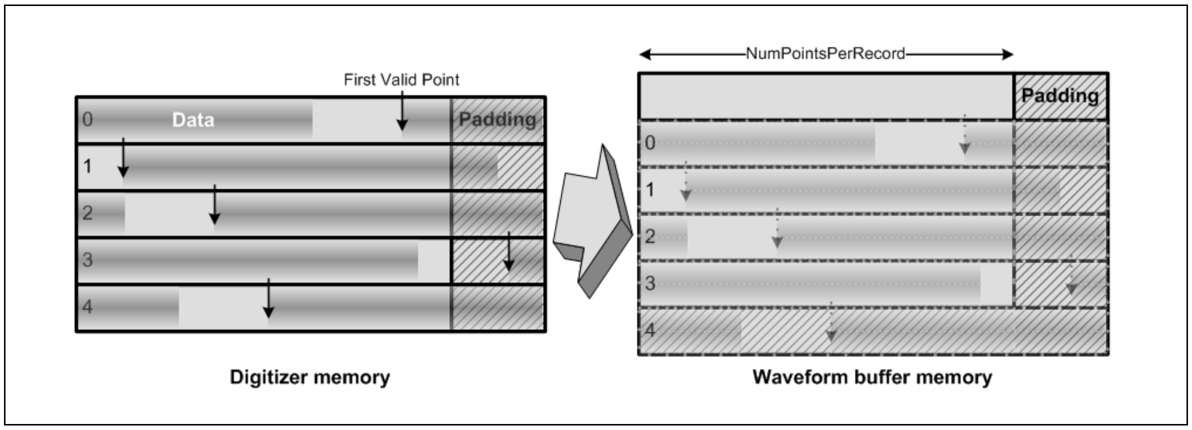


Figure 4-7 Multi-record transfer

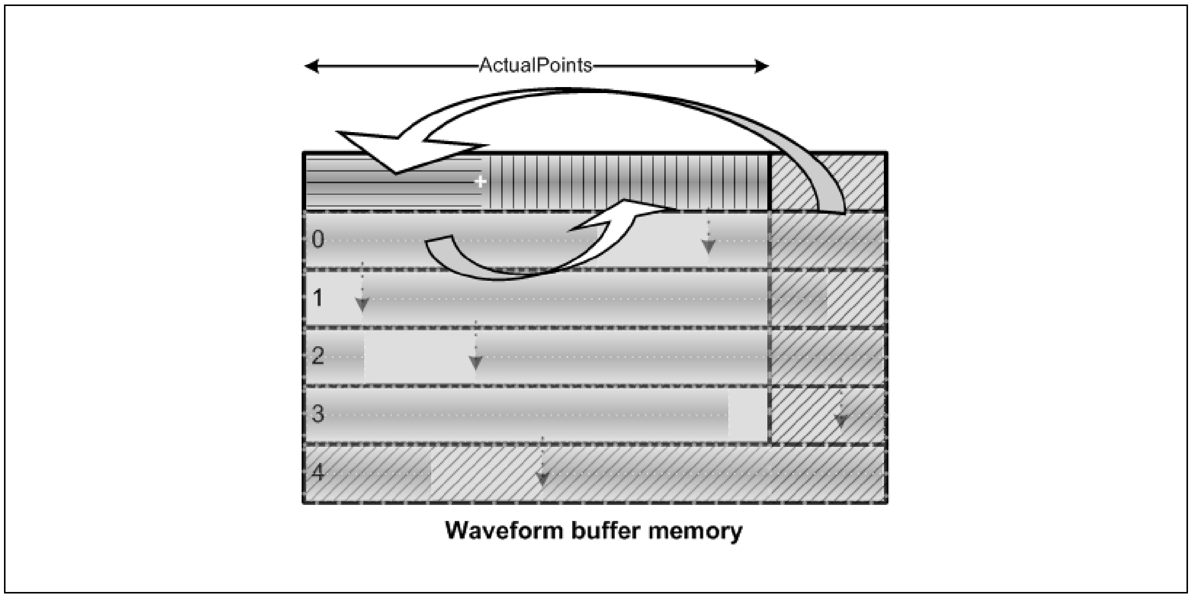


Figure 4-8 Reordering of the records

After reordering, each record has contiguous samples. All records may also be placed contiguously, and the (consolidated) padding pushed to the end of the buffer, e.g. as shown in Figure 4-9.

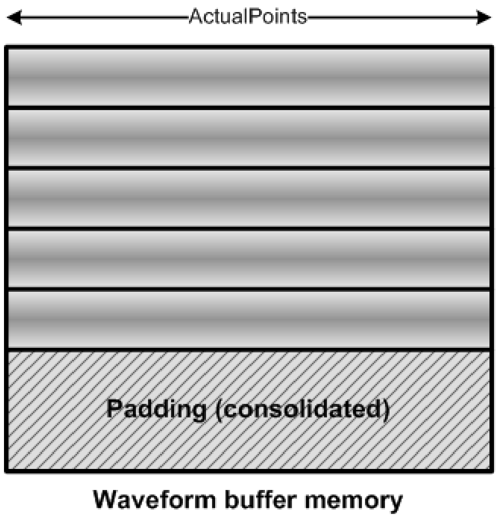


Figure 4-9 Waveform buffer memory after circular buffer reordering of the records

Another example for a 5-records acquisition is shown in Figure 4-10. In this case, all records have been transferred from the digitizer into the waveform buffer as contiguous data blocks and do not require reordering of the record samples. However, alignment constraints may still exist. As a result, the return parameter *FirstValidPoint* must be used to obtain the index of the first point of each record when reading the waveform.

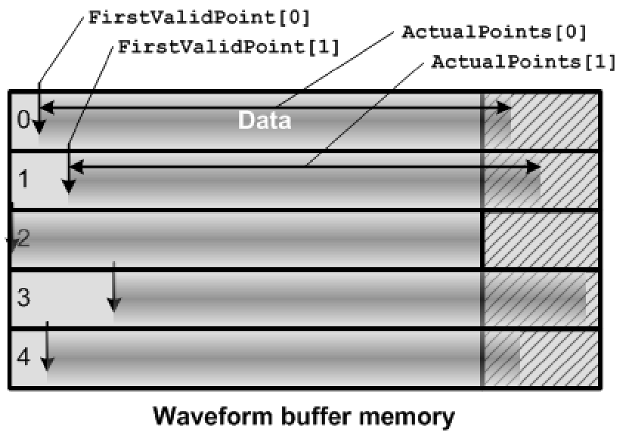


Figure 4-10 Waveform buffer memory not requiring circular buffer reordering of the records

It is important to understand that the transition between Figure 4-5 (i.e. data in the digitizer memory) and Figure 4-9 or Figure 4-10 (i.e. data in the waveform buffer memory, ready to be accessed by the user) is done by the digitizer and/or the driver, not the user.

Vendors generally publish these types of memory requirements in their documentation, but the IviDigitizer interface also supports a function (QueryMinWaveformMemory) that returns the number of samples that should be allocated for best data transfer performance. For single record Read and Fetch calls, the maximum possible value that the FirstValidPoint output parameter may assume can be queried with the MaxFirstValidPointValue read-only attribute.

In IVI-COM, the Read and Fetch calls will allocate the proper amount of memory automatically. Instead of passing an array into the function calls, simply pass a pointer to a NULL SAFEARRAY. The driver will then allocate sufficient memory automatically. It is also allowed to pass a pointer to a non-NULL (user allocated, or driver allocated from a previous call) SAFEARRAY, in which case the driver shall not reallocate memory. For best performance, i.e. to avoid allocating the waveform buffer memory for every Read or Fetch call, the user should let the driver allocate the SAFEARRAY buffer memory by passing a pointer to a NULL SAFEARRAY on the first call, ensuring sufficient memory is allocated. Then the same SAFEARRAY should be reused for subsequent calls, until acquisition parameters changes require a larger buffer size.

In IVI-C, the waveform buffer memory must be allocated in advance by the user.

4.1.5.2 Valid Data Points

The Read and Fetch functions in the IviDigitizer interface, including those of the IviDigitizerMultiRecordAcquisition extension group, utilize several parameters that may require some explanation:

- OffsetWithinRecord
- ActualPoints
- FirstValidPoint

OffsetWithinRecord is the offset within a record to begin fetching data from. A common reason to use this parameter would be when the digitizer's data record is too large to be retrieved with a single Read/Fetch call. OffsetWithinRecord specifies an offset into the digitizer's data record. The first retrieved data point will come from that offset – data that comes before the OffsetWithinRecord index will not be retrieved. Using multiple Fetch calls with appropriate values for OffsetWithinRecord and NumPointsPerRecord allows data from a single record to be retrieved in chunks. An example of the retrieval of a partial record is shown in Figure 4-11.

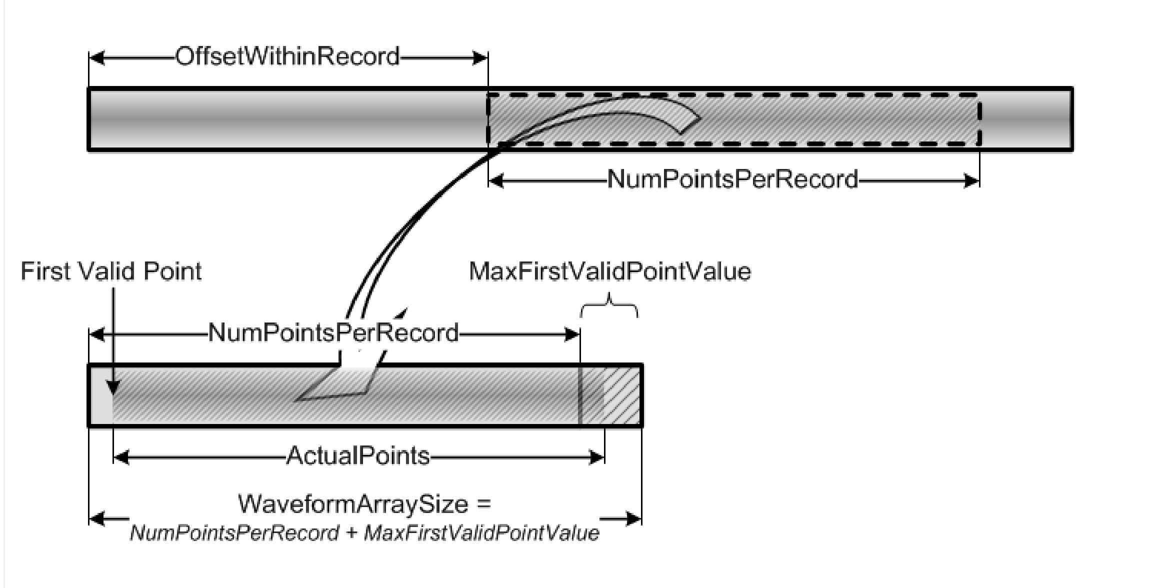


Figure 4-11 Retrieving a partial record

The *ActualPoints* parameter returns the number of valid data points that were retrieved from the digitizer. This number may be less than the full size of the *WaveformArray* memory buffer. This may happen for any of several reasons – the memory buffer may be larger than needed, or the data acquisition might have been interrupted before it was finished, for instance. Also, if *FirstValidPoint* (see below for more details on this parameter) is non-zero, it will necessarily be smaller than the length of the *WaveformArray* buffer. The *ActualPoints* parameter should be examined after Read or Fetch calls to ensure that valid data was returned.

FirstValidPoint is used to maximize data transfer performance. Some digitizers transfer data to the computer using DMA channels that must be aligned at specific byte boundaries. Transferring data into non-aligned memory would cause a severe performance penalty. The data *WaveformArray* parameter, which is intended to receive the data from the digitizer, may not be aligned at the correct memory boundary. In that case the digitizer may transfer its data into *WaveformArray* memory space starting at the first available memory location that is properly aligned. In these cases, the first few bytes of the *WaveformArray* memory buffer will not contain valid data (see an example of this in section 4.1.5.1, Figure 4-10). The *FirstValidPoint* parameter should be examined to determine if this situation has occurred. It gives the offset into the *WaveformArray* memory buffer where the first valid data point can be found. The digitizer's manufacturer should supply documentation as to whether this situation should be expected to occur.

4.1.6 Combining Channels

The IviDigitizer interface allows digitizer channels to be combined in two different ways, and for two different reasons:

- Some digitizers have the ability to interleave multiple analog-to-digital converter channels into a single, higher-rate channel. This is accomplished by forcing each channel to sample the same data with a small time offset. (Note that this means that one of the digitizer's physical input connections will be unused – the signal applied to one input connector will be internally routed to two different ADCs.) If two channels are combined, then the ADC in the first channel samples the data, followed by the ADC from the second channel. The two ADCs proceed to sample the rest of the data in a ping-pong fashion. This results in a sample rate that is twice the rate that either ADC could accomplish by itself.

  To combine channels in this fashion, use the IviDigitizerTimeInterleavedChannels extension group. Channels are combined using a simple comma-separated list.

  In addition, some digitizers have the ability to automatically combine channels based on a desired sample rate. If the Sample Rate attribute is set to a value that is higher than a single channel can support, multiple channels will automatically be combined to achieve the desired sample rate. The IviDigitizerTimeInterleavedChannelListAuto attribute is used to control this feature. This feature should be used, if possible, in cases where instrument interchangeability is desired since the use of the IviDigitizerTimeInterleavedChannelListAuto attribute does not depend on the name of a channel.

- Some digitizers include a feature that allows data from two channels to be interleaved. Usually, the data from two channels is interleaved so that a read or fetch call returns complex (I/Q) data. Use the IviDigitizerDataInterleavedChannels extension group to control this feature. Like the IviDigitizerTimeInterleavedChannels extension group, the IviDigitizerDataInterleavedChannels extension group combines channels using a comma-separated list.

  The IviDigitizerDataInterleavedChannels extension group specifically allows digitizers to return complex data values. If the real and imaginary parts of a complex signal (I and Q) are connected to two digitizer channels, this extension group allows that data to be interleaved in read and fetch calls, resulting in an array of complex data.

These features are distinguished in the following ways:

- When using time-interleaved channels, the data returned by a fetch or read call consists of points that are sequential in time, at a higher data rate than a single ADC could support.

- When using data-interleaved channels, the data returned by a fetch or read call consists of several points (one per combined channel) that were taken at the *same* time – one for each channel. The following data values return data for the next time point (again, one point for each channel, sampled at the same time).

Channels can be combined in both ways at the same time. If so, then time-interleaving applies before data-interleaving. For instance, a four-channel digitizer may combine channels 1 and 2 to achieve a higher data rate, and may then combine channels 3 and 4 to achieve the same data rate. Then the two remaining active channels (channel 1 and channel 3) may be data-interleaved.

When interleaving channels, the Record Size is the number of points across all channels. Note that data-interleaved channels cannot return an arbitrary number of data points. If two channels are data-combined, then the number of returned points must always be a multiple of two, because each of the two channels will return one data point per time sample. In this case, attempting to retrieve an odd number of data points would result in a meaningless request. In Read and Fetch calls, the number of points requested must then be a multiple of the number of combined channels. Otherwise, an error will occur. This is not the case with time-interleaved channels, where any number of points can be returned.

4.1.6.1 Notes on the Use of Interleaved Channels

Users who often switch interleaving features on and off should be aware of some subtleties that may be encountered.

Like all IVI drivers, the IviDigitizer interface is in some cases designed to support functionality that is not implemented in all digitizers. The IviDigitizer interface allows for (but does not require) very general interleaving capabilities. Digitizers that do not support the full generality of this interface are free to implement a subset, but this subset’s behavior must be documented.

For instance, the interface makes it possible to combine any two channels in a four-channel digitizer to achieve a higher sample rate. The digitizer hardware may then require the other two channels be combined as well (so there is only a single time base for the entire digitizer). In such a case, there must be two active channels – one chosen by the user when the two original channels are combined, and one chosen by the digitizer when it combines the remaining pair of channels. In this case, the remaining “active” channel should be the lowest-numbered channel on the digitizer, as documented by the hardware manufacturer. But users should be free to change the second active channel, if desired.

A similar situation applies to data-interleaved channels. It must be possible for users to choose which two channels are to be data-interleaved, and to change this selection as needed.

Another common use case is for users to route signals to two different inputs of a digitizer and then combine the channels to achieve higher sample rates. In this case, one of the combined channels becomes the active channel and the other is set to the inactive state. Users can then change the active channel setting to take data from the alternate input connectors.

In all of these cases, the programmer must first disable the active channel before changing the active channel. This is necessary in order to avoid “invisible” coupling between the Enabled state of each channel and the ‘combined channel’ list. In the IVI interface, the state of the TimeInterleavedChannelList property (which is a property of each channel) is ignored if value of the Enabled flag is ‘false’. The TimeInterleavedChannelList property is only used for active channels. The same is true of the DataInterleavedChannelList.

For example, suppose a two-channel digitizer is initialized so that Channel 1 is enabled, Channel 2 is disabled, and the TimeInterleavedChannelList property of Channel 1 contains the name of Channel 2. Data can then be taken from Channel 1 at double-data-rate speeds.

To change the active channel in this example to Channel 2, Channel 1 must be disabled, Channel 2 enabled, and the TimeInterleavedChannelList property for Channel 2 must contain the name of Channel 1. Both channels cannot be enabled at the same time if they are combined, and attempts to do so will result in an error. However, both channels can be disabled at the same time as long as no Fetch/Read calls are made while the channels are in the disabled state.

4.2 IviDigitizerBase Attributes

The IviDigitizerBase capability group defines the following attributes:

- Active Trigger Source
- Channel Count
- Channel Enabled
- Channel Item (IVI-COM and IVI.NET only)
- Channel Name (IVI-COM and IVI.NET only)
- Input Connector Selection
- Input Impedance
- Is Idle
- Is Measuring
- Is Waiting For Arm
- Is Waiting For Trigger
- Max First Valid Point Value
- Max Samples Per Channel
- Min Record Size
- Num Acquired Records
- Num Records To Acquire
- Record Size
- Sample Rate
- Trigger Coupling
- Trigger Delay
- Trigger Hysteresis
- Trigger Level
- Trigger Output Enabled
- Trigger Slope
- Trigger Source Count
- Trigger Source Item (IVI-COM and IVI.NET only)
- Trigger Source Name (IVI-COM and IVI.NET only)
- Trigger Type
- Vertical Coupling
- Vertical Offset
- Vertical Range

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

4.2.1 Active Trigger Source

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	R/W	N/A	None	ConfigureActiveTriggerSource

.NET Property Name

`Trigger.ActiveSource`

COM Property Name

`Trigger.ActiveSource`

C Constant Name

`IVIDIGITIZER_ATTR_ACTIVE_TRIGGER_SOURCE`

Description

Specifies the source the digitizer monitors for the trigger event. The value specified here must be one of the valid repeated capability names for the TriggerSource repeated capability.

If an IVI driver supports a trigger source and the trigger source is listed in IVI-3.3 *Cross Class Capabilities Specification,* Section 3, then the IVI driver shall accept the standard string for that trigger source. This attribute is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new trigger source strings for trigger sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

This attribute only affects instrument behavior when either the IviDigitizerMultiTrigger extension group is not supported or the Trigger Source Operator is set to None.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.2 Channel Count

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	N/A

.NET Property Name

`Channels.Count`

This property is inherited from `IIviRepeatedCapabilityCollection`.

COM Property Name

`Channels.Count`

C Constant Name

`IVIDIGITIZER_ATTR_CHANNEL_COUNT`

Description

Returns the number of channels available on the device.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.3 Channel Enabled

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	Channel	None	Configure Channel

.NET Property Name

`Channels[].Enabled`

COM Property Name

`Channels.Item().Enabled`

C Constant Name

`IVIDIGITIZER_ATTR_CHANNEL_ENABLED`

Description

Specifies whether the digitizer acquires a waveform for the channel.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.4 Channel Item (IVI-COM & IVI.NET Only)

Data Type	Access	Applies To	Coercion	High Level Functions
`IIviDigitizerChannel*`	RO	Channel	None	N/A

.NET Property Name

```
IIviDigitizerChannel Channels[String name];
```

This indexer is inherited from `IIviRepeatedCapabilityCollection`. The name parameter uniquely identifies a particular channel in the channels collection.

COM Property Name

```
Channels.Item([in] BSTR Name)
```

C Constant Name

```
N/A
```

Description

Channel Item uniquely identifies a channel in the channels collection. It returns an interface pointer which can be used to control the attributes and other functionality of that channel.

The Item property takes a channel name. If the user passes an invalid value for the `Name` parameter, the property returns an error.

Valid names include physical repeated capability identifiers and virtual repeated capability identifiers.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.5 Channel Name (IVI-COM & IVI.NET Only)

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	RO	Channel	None	N/A

.NET Property Name

`Channels[].Name`

This property is inherited from `IIviRepeatedCapabilityIdentification`.

COM Property Name

`Channels.Name([in] LONG index)`

C Constant Name

`N/A`
(Use the `GetChannelName` function.)

Description

This property returns the physical channel identifier that corresponds to the index that the user specifies. If the driver defines a qualified channel name, this property returns the qualified name.

In COM, the index is one-based. In .NET, the index is zero-based.

For COM, if the value that the user passes for the Index parameter is less than one or greater than the value of the Channel Count attribute, the property returns an empty string in the Name parameter and returns the Invalid Value error.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.6 Input Connector Selection

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	Channel	None	N/A

.NET Property Name

`Channels[].InputConnectorSelection`

COM Property Name

`Channels.Item().InputConnectorSelection`

C Constant Name

`IVIDIGITIZER_ATTR_INPUT_CONNECTOR_SELECTION`

Description

Some digitizers include multiple connectors for each digitizer input channel. These connectors are often simply a matter of convenience for system cabling – multiple signals can be routed to the various connectors, the desired signal can be sent into the digitizer by changing an internal switch. With other digitizers, the connectors may be of different types or even different impedances. This attribute is used to determine which connector is to be used.

Values for this attribute are 1-based. Digitizers that have only a single connector for each channel should only support a value of 1 for this attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.7 Input Impedance

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	Channel	Up	N/A

.NET Property Name

`Channels[].InputImpedance`

COM Property Name

`Channels.Item().InputImpedance`

C Constant Name

`IVIDIGITIZER_ATTR_INPUT_IMPEDANCE`

Description

The input impedance of this channel. The units are Ohms.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.8 Is Idle

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	Is Idle (IVI-C only)

.NET Property Name

`Acquisition.Status.IsIdle`

.NET Enumeration Name

`AcquisitionStatusResult`

COM Property Name

`Acquisition.Status.IsIdle`

COM Enumeration Name

`IviDigitizerAcquisitionStatusResultEnum`

C Constant Name

`IVIDIGITIZER_ATTR_IS_IDLE`

Description

Returns whether the device is currently in the Idle state. If the driver cannot query the digitizer to return its state, the driver returns the value Unknown.

Defined Values

Name	***Description***		
		Language	***Identifier***
True	The digitizer is currently in the Idle state.		
		.NET	`AcquisitionStatusResult.True`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
		COM	`IviDigitizerAcquisitionStatusResultTrue`
False	The digitizer is not currently in the Idle state..		
		.NET	`AcquisitionStatusResult.False`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
		COM	`IviDigitizerAcquisitionStatusResultFalse`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Idle state.		
		.NET	`AcquisitionStatusResult.Unknown`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`
		COM	`IviDigitizerAcquisitionStatusResultUnknown`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.9 Is Measuring

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	Is Measuring (IVI-C only)

.NET Property Name

`Acquisition.Status.IsMeasuring`

.NET Enumeration Name

`AcquisitionStatusResult`

COM Property Name

`Acquisition.Status.IsMeasuring`

COM Enumeration Name

`IviDigitizerAcquisitionStatusResultEnum`

C Constant Name

`IVIDIGITIZER_ATTR_IS_MEASURING`

Description

Returns whether the device is currently in the Measuring state. If the driver cannot query the digitizer to return its state, the driver returns the value Unknown.

Defined Values

Name	***Description***		
		Language	***Identifier***
True	The digitizer is currently in the Measuring state.		
		.NET	`AcquisitionStatusResult.True`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
		COM	`IviDigitizerAcquisitionStatusResultTrue`
False	The digitizer is not currently in the Measuring state.		
		.NET	`AcquisitionStatusResult.False`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
		COM	`IviDigitizerAcquisitionStatusResultFalse`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Measuring state.		
		.NET	`AcquisitionStatusResult.Unknown`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`
		COM	`IviDigitizerAcquisitionStatusResultUnknown`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.10 Is Waiting For Arm

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	Is Waiting For Arm (IVI-C only)

.NET Property Name

`Acquisition.Status.IsWaitingForArm`

.NET Enumeration Name

`AcquisitionStatusResult`

COM Property Name

`Acquisition.Status.IsWaitingForArm`

COM Enumeration Name

`IviDigitizerAcquisitionStatusResultEnum`

C Constant Name

`IVIDIGITIZER_ATTR_IS_WAITING_FOR_ARM`

Description

Returns whether the device is currently in the Waiting For Arm state. If the driver cannot query the digitizer to return its state, the driver returns the value Unknown.

Defined Values

Name	*Description*		
		Language	*Identifier*
True	The digitizer is currently in the Waiting For Arm state.		
		.NET	`AcquisitionStatusResult.True`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
		COM	`IviDigitizerAcquisitionStatusResultTrue`
False	The digitizer is not currently in the Waiting For Arm state.		
		.NET	`AcquisitionStatusResult.False`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
		COM	`IviDigitizerAcquisitionStatusResultFalse`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Waiting For Arm state.		
		.NET	`AcquisitionStatusResult.Unknown`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`
		COM	`IviDigitizerAcquisitionStatusResultUnknown`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.11 Is Waiting For Trigger

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	Is Waiting For Trigger (IVI-C only)

.NET Property Name

`Acquisition.Status.IsWaitingForTrigger`

.NET Enumeration Name

`AcquisitionStatusResult`

COM Property Name

`Acquisition.Status.IsWaitingForTrigger`

COM Enumeration Name

`IviDigitizerAcquisitionStatusResultEnum`

C Constant Name

`IVIDIGITIZER_ATTR_IS_WAITING_FOR_TRIGGER`

Description

Returns whether the device is currently in the Waiting For Trigger state. If the driver cannot query the digitizer to return its state, the driver returns the value Unknown.

Defined Values

Name	*Description*		
		Language	*Identifier*
True	The digitizer is currently in the Waiting For Trigger state.		
		.NET	`AcquisitionStatusResult.True`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
		COM	`IviDigitizerAcquisitionStatusResultTrue`
False	The digitizer is not currently in the Waiting For Trigger state.		
		.NET	`AcquisitionStatusResult.False`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
		COM	`IviDigitizerAcquisitionStatusResultFalse`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Waiting For Trigger state.		
		.NET	`AcquisitionStatusResult.Unknown`
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`
		COM	`IviDigitizerAcquisitionStatusResultUnknown`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.12 Max First Valid Point Value

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	RO	N/A	None	N/A

.NET Property Name

`Acquisition.MaxFirstValidPointValue`

COM Property Name

`Acquisition.MaxFirstValidPointValue`

C Constant Name

`IVIDIGITIZER_ATTR_MAX_FIRST_VALID_POINT_VAL`

Description

Returns the maximum value that the First Valid Point parameter of the readout functions may assume. This value is necessary to calculate the minimum size of the required data buffer to retrieve the entire acquisition.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.13 Max Samples Per Channel

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	RO	N/A	None	N/A

.NET Property Name

`Acquisition.MaxSamplesPerChannel`

COM Property Name

`Acquisition.MaxSamplesPerChannel`

C Constant Name

`IVIDIGITIZER_ATTR_MAX_SAMPLES_PER_CHANNEL`

Description

Returns the maximum number of samples per channel that can be captured.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.14 Min Record Size

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	RO	N/A	None	N/A

.NET Property Name

`Acquisition.MinRecordSize`

COM Property Name

`Acquisition.MinRecordSize`

C Constant Name

`IVIDIGITIZER_ATTR_MIN_RECORD_SIZE`

Description

Indicates the minimum waveform record size. If the digitizer can support any arbitrary size record, then this attribute returns 1.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.15 Num Acquired Records

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	RO	N/A	None	N/A

.NET Property Name

`Acquisition.NumberOfAcquiredRecords`

COM Property Name

`Acquisition.NumAcquiredRecords`

C Constant Name

`IVIDIGITIZER_ATTR_NUM_ACQUIRED_RECORDS`

Description

Gets the total number of records acquired since the acquistion was last initiated. You may call this method while an acquisition is in progress.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.16 Num Records To Acquire

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	R/W	N/A	None	Configure Acquisition Record

.NET Property Name

`Acquisition.NumberOfRecordsToAcquire`

COM Property Name

`Acquisition.NumRecordsToAcquire`

C Constant Name

`IVIDIGITIZER_ATTR_NUM_RECORDS_TO_ACQUIRE`

Description

Specifies the number of waveform records to acquire. One waveform record is acquired for each recognized trigger per active channel.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.17 Record Size

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	R/W	N/A	None	Configure Acquisition Record

.NET Property Name

`Acquisition.RecordSize`

COM Property Name

`Acquisition.RecordSize`

C Constant Name

`IVIDIGITIZER_ATTR_RECORD_SIZE`

Description

Specifies the number of samples to acquire in each waveform record.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.18 Sample Rate

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	N/A	Up	Configure Acquisition Record

.NET Property Name

`Acquisition.SampleRate`

COM Property Name

`Acquisition.SampleRate`

C Constant Name

`IVIDIGITIZER_ATTR_SAMPLE_RATE`

Description

Specifies the rate of the sample clock in samples per second.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.19 Trigger Coupling

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	N/A

.NET Property Name

`Trigger.Sources[].Coupling`

.NET Enumeration Name

`TriggerCoupling`

COM Property Name

`Trigger.Sources.Item().Coupling`

COM Enumeration Name

`IviDigitizerTriggerCouplingEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_COUPLING`

Description

Specifies how the digitizer couples the trigger source.

Defined Values

Name	*Description*		
		Language	*Identifier*
AC	The digitizer AC couples the trigger signal.		
		.NET	`TriggerCoupling.AC`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_AC`
		COM	`IviDigitizerTriggerCouplingAC`
DC	The digitizer DC couples the trigger signal.		
		.NET	`TriggerCoupling.DC`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_DC`
		COM	`IviDigitizerTriggerCouplingDC`
HF Reject	The digitizer filters out the high frequencies from the trigger signal.		
		.NET	`TriggerCoupling.HFReject`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_HF_REJECT`
		COM	`IviDigitizerTriggerCouplingHFReject`
LF Reject	The digitizer filters out the low frequencies from the trigger signal.		

		.NET	`TriggerCoupling.LFReject`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_LF_REJECT`
		COM	`IviDigitizerTriggerCouplingLFReject`
Noise Reject	The digitizer filters out the noise from the trigger signal.		
		.NET	`TriggerCoupling.NoiseReject`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_NOISE_REJECT`
		COM	`IviDigitizerTriggerCouplingNoiseReject`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.20 Trigger Delay

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	N/A	Down	N/A

.NET Property Name

`Trigger.Delay`

COM Property Name

`Trigger.Delay`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_DELAY`

Description

Specifies the length of time from the trigger event to the first point in the waveform record. If the value is positive, the first point in the waveform record occurs after the trigger event. If the value is negative, the first point in the waveform record occurs before the trigger event. For C and COM the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.21 Trigger Hysteresis

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	TriggerSource	None	N/A

.NET Property Name

`Trigger.Sources[].Hysteresis`

COM Property Name

`Trigger.Sources.Item().Hysteresis`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_HYSTERESIS`

Description

Specifies the trigger hysteresis in Volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.22 Trigger Level

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	TriggerSource	None	Configure Edge Trigger Source Configure Glitch Trigger Source Configure Width Trigger Source

.NET Property Name

`Trigger.Sources[].Level`

COM Property Name

`Trigger.Sources.Item().Level`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_LEVEL`

Description

Specifies the voltage threshold for the trigger sub-system. The units are Volts. This attribute affects instrument behavior only when the Trigger Type is set to one of the following values: Edge Trigger, Glitch Trigger, or Width Trigger.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.23 Trigger Output Enabled

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	N/A	None	N/A

.NET Property Name

`Trigger.OutputEnabled`

COM Property Name

`Trigger.OutputEnabled`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_OUTPUT_ENABLED`

Description

Specifies whether or not an accepted trigger appears at an output of the digitizer.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.24 Trigger Slope

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure Edge Trigger Source

.NET Property Name

`Trigger.Sources[].Edge.Slope`

.NET Enumeration Name

`Slope`

COM Property Name

`Trigger.Sources.Item().Edge.Slope`

COM Enumeration Name

`IviDigitizerTriggerSlopeEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_SLOPE`

Description

Specifies whether a rising or a falling edge triggers the digitizer. This attribute affects instrument operation only when the Trigger Type attribute is set to Edge Trigger.

Defined Values

Name	***Description***		
		Language	***Identifier***
Negative	A negative (falling) edge passing through the trigger level triggers the digitizer.		
		.NET	`Slope.Negative`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`
		COM	`IviDigitizerTriggerSlopeNegative`
Positive	A positive (rising) edge passing through the trigger level triggers the digitizer.		
		.NET	`Slope.Positive`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`
		COM	`IviDigitizerTriggerSlopePositive`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.25 Trigger Source Count

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	N/A

.NET Property Name

`Trigger.Sources.Count`

This property is inherited from `IIviRepeatedCapabilityCollection`.

COM Property Name

`Trigger.Sources.Count`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_SOURCE_COUNT`

Description

Returns the number of trigger sources available on the device.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.26 Trigger Source Item (IVI-COM & IVI.NET only)

Data Type	Access	Applies To	Coercion	High Level Functions
`IIviDigitizerTriggerSource*`	RO	TriggerSource	None	N/A

.NET Property Name

```
IIviDigitizerTriggerSource Sources[String name];
```

This indexer is inherited from `IIviRepeatedCapabilityCollection`. The name parameter uniquely identifies a particular trigger source in the trigger sources collection.

COM Property Name

```
Trigger.Sources.Item([in] BSTR Name)
```

C Constant Name

```
N/A
```

Description

Trigger Source Item uniquely identifies a trigger source in the trigger sources collection. It returns an interface pointer which can be used to control the attributes and other functionality of that trigger source.

The Item property takes a trigger source name. If the user passes an invalid value for the `Name` parameter, the property returns an error.

Valid names include physical repeated capability identifiers and virtual repeated capability identifiers.

If an IVI driver supports a trigger source and the trigger source is listed in IVI-3.3 *Cross Class Capabilities Specification,* Section 3, then the IVI driver shall accept the standard string for that trigger source. This attribute is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new trigger source strings for trigger sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

4.2.27 Trigger Source Name (IVI-COM & IVI.NET only)

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	RO	TriggerSource	None	N/A

.NET Property Name

`Trigger.Sources[].Name`

This property is inherited from `IIviRepeatedCapabilityIdentification`.

COM Property Name

`Trigger.Sources.Name([in] LONG Index)`

C Constant Name

`N/A`
(Use the `GetTriggerSourceName` function.)

Description

This property returns the physical trigger source identifier that corresponds to the index that the user specifies. If the driver defines a qualified trigger source name, this property returns the qualified name.

In COM, the index is one-based. In .NET, the index is zero-based.

For COM, if the value that the user passes for the Index parameter is less than one or greater than the value of the Trigger Source Count attribute, the property returns an empty string in the Name parameter and returns the Invalid Value error.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.28 Trigger Type

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	N/A

.NET Property Name

`Trigger.Sources[].Type`

.NET Enumeration Name

`TriggerType`

COM Property Name

`Trigger.Sources.Item().Type`

COM Enumeration Name

`IviDigitizerTriggerTypeEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_TYPE`

Description

The kind of event that triggers the digitizer.

Defined Values

Name	***Description***		
		Language	***Identifier***
Trigger Edge	Configures the digitizer for edge triggering. An edge trigger occurs when the trigger signal specified with the Trigger Source attribute passes the voltage threshold specified with the Trigger Level attribute and has the slope specified with the Trigger Slope attribute.		
		.NET	`TriggerType.Edge`
		C	`IVIDIGITIZER_VAL_EDGE_TRIGGER`
		COM	`IviDigitizerTriggerEdge`
Trigger Width	Configures the digitizer for width triggering. Use the IviDigitizerWidthTrigger extension properties and methods to configure the trigger.		
		.NET	`TriggerType.Width`
		C	`IVIDIGITIZER_VAL_WIDTH_TRIGGER`
		COM	`IviDigitizerTriggerWidth`
Trigger Runt	Configures the digitizer for runt triggering. Use the IviDigitizerRuntTrigger extension properties and methods to configure the trigger.		
		.NET	`TriggerType.Runt`

		C	`IVIDIGITIZER_VAL_RUNT_TRIGGER`
		COM	`IviDigitizerTriggerRunt`
Trigger Glitch	Configures the digitizer for glitch triggering. Use the IviDigitizerGlitchTrigger extension properties and methods to configure the trigger.		
		.NET	`TriggerType.Glitch`
		C	`IVIDIGITIZER_VAL_GLITCH_TRIGGER`
		COM	`IviDigitizerTriggerGlitch`
Trigger TV	Configures the digitizer for triggering on TV signals. Use the IviDigitizerTVTrigger extension properties and methods to configure the trigger.		
		.NET	`TriggerType.TV`
		C	`IVIDIGITIZER_VAL_TV_TRIGGER`
		COM	`IviDigitizerTriggerTV`
Trigger Window	Configures the digitizer for window triggering. Use the IviDigitizerWindowTrigger extension properties and methods to configure the trigger.		
		.NET	`TriggerType.Window`
		C	`IVIDIGITIZER_VAL_WINDOW_TRIGGER`
		COM	`IviDigitizerTriggerWindow`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.29 Vertical Coupling

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	Channel	None	Configure Channel

.NET Property Name

`Channels[].Coupling`

.NET Enumeration Name

`VerticalCoupling`

COM Property Name

`Channels.Item().Coupling`

COM Enumeration Name

`IviDigitizerVerticalCouplingEnum`

C Constant Name

`IVIDIGITIZER_ATTR_VERTICAL_COUPLING`

Description

Specifies how the digitizer couples the input signal for the channel.

Defined Values

Name	*Description*		
		Language	*Identifier*
AC	The digitizer AC couples the input signal.		
		.NET	`VerticalCoupling.AC`
		C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_AC`
		COM	`IviDigitizerVerticalCouplingAC`
DC	The digitizer DC couples the input signal.		
		.NET	`VerticalCoupling.DC`
		C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_DC`
		COM	`IviDigitizerVerticalCouplingDC`
Gnd	The digitizer couples the channel to the ground.		
		.NET	`VerticalCoupling.Ground`
		C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_GND`
		COM	`IviDigitizerVerticalCouplingGnd`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.30 Vertical Offset

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	Channel	None	Configure Channel

.NET Property Name

`Channels[].Offset`

COM Property Name

`Channels.Item().Offset`

C Constant Name

`IVIDIGITIZER_ATTR_VERTICAL_OFFSET`

Description

The location of the center of the range that you specify with the Range attribute. The units are Volts, with respect to ground. For example, to acquire a sine wave spanning 0.0 to 10.0 volts, set Offset to 5.0 volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.2.31 Vertical Range

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	Channel	Up	Configure Channel

.NET Property Name

`Channels[].Range`

COM Property Name

`Channels.Item().Range`

C Constant Name

`IVIDIGITIZER_ATTR_VERTICAL_RANGE`

Description

The absolute value of the input range the digitizer can acquire for the channel. The units are Volts. For example, to acquire a sine wave spanning -5.0 to 5.0 volts, set Range to 10.0 volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

4.3 IviDigitizerBase Functions

The IviDigitizerBase capability group defines the following functions:

- Abort
- Configure Acquisition Record
- Configure Active Trigger Source (IVI-C only)
- Configure Channel
- Configure Edge Trigger Source
- Create Waveform (IVI.NET Only)
- Fetch Waveform Int16
- Fetch Waveform Int32
- Fetch Waveform Int8
- Fetch Waveform Real64
- Get Channel Name (IVI-C only)
- Get Trigger Source Name (IVI-C only)
- Initiate Acquisition
- Is Idle (IVI-C only)
- Is Measuring (IVI-C only)
- Is Waiting For Arm (IVI-C only)
- Is Waiting For Trigger (IVI-C only)
- QueryMinWaveformMemory
- Read Waveform Int16
- Read Waveform Int32
- Read Waveform Int8
- Read Waveform Real64
- Wait For Acquisition Complete

This section describes the behavior and requirements of each function.

4.3.1 Abort

Description

Aborts an acquisition and returns the digitizer to the Idle state.

.NET Method Prototype

```
void Acquisition.Abort ();
```

COM Method Prototype

```
HRESULT Acquisition.Abort ();
```

C Prototype

```
ViStatus IviDigitizer_Abort (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.3.2 Configure Acquisition Record

Description

This function configures the most commonly configured attributes of the digitizer acquisition sub-system. These attributes are the samples per record, the number of records to acquire, and the sample rate.

.NET Method Prototype

```
void Acquisition.ConfigureAcquisition (Int64 numberOfRecordsToAcquire,
                                       Int64 recordSize,
                                       Double sampleRate);
```

COM Method Prototype

```
HRESULT Acquisition.ConfigureAcquisition ([in] long NumRecordsToAcquire,
                                          [in] long RecordSize,
                                          [in] double SampleRate);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureAcquisition (ViSession Vi,
                                            ViInt64 NumRecordsToAcquire,
                                            ViInt64 RecordSize,
                                            ViReal64 SampleRate);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

`NumRecordsToAcquire (C/COM)`
`numberOfRecordsToAcquire (.NET)`
	Specifies the number of records in the acquisition. This value sets the Num Records To Acquire attribute.	`ViInt64`

`RecordSize`	Specifies the number of samples in each record. This value sets the Record Size attribute.	`ViInt64`
`SampleRate`	Specifies the sample rate in samples per second. This value sets the Sample Rate attribute.	`ViReal64`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.3.3 Configure Active Trigger Source (IVI-C Only)

Description

This function sets the Active Trigger Source attribute, which specifies the source the digitizer monitors for the trigger event. The value specified as parameter must be one of the valid repeated capability names for the TriggerSource repeated capability.

If an IVI driver supports a trigger source and the trigger source is listed in IVI-3.3 *Cross Class Capabilities Specification,* Section 3, then the IVI driver shall accept the standard string for that trigger source. The source parameter is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new trigger source strings for trigger sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

The Active Trigger Source attribute only affects instrument behavior when either the IviDigitizerMultiTrigger extension group is not supported or the Trigger Source Operator is set to None.

.NET Method Prototype

```
N/A
```

(Use the `Trigger.ActiveSource` property)

COM Method Prototype

```
N/A
```

(Use the `Trigger.ActiveSource` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureActiveTriggerSource (ViSession Vi,
                                                    ViString Source);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`Source`	Specifies the trigger source that is to be set as the Active Trigger Source.	`ViString`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.4 Configure Channel

Description

This function configures the most commonly configured attributes of the digitizer channel sub-system. These attributes are the range, offset, coupling, and whether the channel is enabled.

.NET Method Prototype

```
void Channels[].Configure (Double range,
                           Double offset,
                           VerticalCoupling coupling,
                           Boolean enabled);
```

COM Method Prototype

```
HRESULT Channels.Item().Configure ([in] double Range,
                                   [in] double Offset,
                                   [in] IviDigitizerVerticalCouplingEnum Coupling,
                                   [in] VARIANT_BOOL Enabled);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureChannel (ViSession Vi,
                                        ViConstString ChannelName,
                                        ViReal64 Range,
                                        ViReal64 Offset,
                                        ViInt32 Coupling,
                                        ViBoolean Enabled);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`ChannelName`	Name of the digitizer channel to configure.	`ViConstString`
`Range`	Specifies the vertical range. This value sets the Vertical Range attribute.	`ViReal64`
`Offset`	Specifies the vertical offset. This value sets the Vertical Offset attribute.	`ViReal64`
`Coupling`	Specifies how to couple the input signal. This value sets the Vertical Coupling attribute.	`ViInt32`
`Enabled`	Specifies if the channel is enabled for acquisition. This value sets the Channel Enabled attribute.	`ViBoolean`

Defined Values for the Coupling Parameter

Name	*Description*		
		Language	*Identifier*
AC	The digitizer AC couples the input signal.		
		.NET	`VerticalCoupling.AC`
		C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_AC`
		COM	`IviDigitizerVerticalCouplingAC`

DC	The digitizer DC couples the input signal.		
		.NET	`VerticalCoupling.DC`
		C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_DC`
		COM	`IviDigitizerVerticalCouplingDC`
Gnd	The digitizer couples the channel to the ground.		
		.NET	`VerticalCoupling.Ground`
		C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_GND`
		COM	`IviDigitizerVerticalCouplingGnd`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.3.5 Configure Edge Trigger Source

Description

This function sets the edge triggering attributes. An edge trigger occurs when the trigger signal that the end-user specifies with the Source parameter passes through the voltage threshold that the end-user specifies with the level parameter and has the slope that the end-user specifies with the Slope parameter. This function affects instrument behavior only if the Trigger Type is Edge Trigger. Set the Trigger Type and Trigger Coupling before calling this function. If the trigger source is one of the analog input channels, an application program should configure the vertical range, vertical coupling, and the maximum input frequency before calling this function.

.NET Method Prototype

```
void Trigger.Sources[].Edge.Configure (Double level,
                                       Slope slope);
```

COM Method Prototype

```
HRESULT Trigger.Sources.Item().Edge.Configure ([in] double Level,
                                      [in] IviDigitizerTriggerSlopeEnum Slope);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureEdgeTriggerSource (ViSession Vi,
                                                  ViConstString Source,
                                                  ViReal64 Level,
                                                  ViInt32 Slope);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the trigger source that is to be configured.	`ViString`
`Level`	Specifies the trigger level. This value sets the Trigger Level attribute.	`ViReal64`
`Slope`	Specifies the trigger slope. This value sets the Trigger Slope attribute.	`ViInt32`

Defined Values for the Slope Parameter

Name	***Description***		
		Language	***Identifier***
Negative	A negative (falling) edge passing through the trigger level triggers the digitizer.		
		.NET	`Slope.Negative`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`
		COM	`IviDigitizerTriggerSlopeNegative`
Positive	A positive (rising) edge passing through the trigger level triggers the digitizer.		
		.NET	`Slope.Positive`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`
		COM	`IviDigitizerTriggerSlopePositive`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.3.6 Create Waveform (IVI.NET Only)

Description

This function creates a waveform object and shall allocate the necessary memory to transfer a waveform from the instrument to the host.

If size is zero, the driver shall allocate the waveform memory with a size based on the current driver configuration.

.NET Method Prototype

```
IWaveform<Double> Acquisition.CreateWaveformDouble (Int64 size);
IWaveform<Int32> Acquisition.CreateWaveformInt32 (Int64 size);
IWaveform<Int16> Acquisition.CreateWaveformInt16 (Int64 size);
IWaveform<SByte> Acquisition.CreateWaveformSByte (Int64 size);
```

COM Method Prototype

```
N/A
```

C Prototype

```
N/A
```

Parameters

Inputs	Description	Base Type
`size`	The number of points in the waveform array.	`Int64`
Outputs	**Description**	**Base Type**
`Return value (.NET)`	The newly allocated waveform.	`Ivi.Driver.IWaveform<T>`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.3.7 Fetch Waveform Int16

Description

This function returns the waveform the digitizer acquired for the specified channel. The waveform is from a previously initiated acquisition.

You use the Initiate Acquisition function to start an acquisition on the channels that the end-user configures with the Configure Channel function. The digitizer acquires waveforms on the concurrently enabled channels. If the channel is not enabled for the acquisition, this function returns the Channel Not Enabled error.

You can use the Acquisition Status function to determine when the acquisition is complete. You must call the FetchWaveformInt16 function separately for each enabled channel to obtain the waveforms. Alternatively, you can use the Wait For Acquisition Complete function to block the calling program until the acquisition is finished.

You can call the Read Waveform Int16 function instead of the Initiate Acquisition function. The Read Waveform Int16 function starts an acquisition on all enabled channels, waits for the acquisition to complete, and returns the waveform (as well as various waveform parameters) for the specified channel. You call this function to obtain the waveforms for each of the remaining channels.

The behavior is different for IVI-C/IVI-COM and IVI.NET as follows:

IVI-C/IVI-COM: After this function executes, each element in the `WaveformArray` parameter is an unscaled value directly from the digitizer's analog-to-digital converter (ADC).

IVI.NET: For .NET the return value of IWaveform<Int16> is a waveform object. Refer to Section 4, *Common Properties and Methods of Waveform and Spectrum Interfaces*, and Section 5, *IWaveform<T> Interface*, of *IVI-3.18: IVI.NET Utility Classes and Interfaces Specification,* for the definition of the IWaveform object and information regarding its use. In particular, refer to Section 4.2, *How to use Waveform and Spectrum Types*, in *IVI-3.18: IVI.NET Utility Classes and Interfaces Specification*, for more information about how to implement these methods.

For IVI.NET, the waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform. To allocate memory during the call to this method, one of the following approaches may be used:

- Set the waveform parameter to `(IWaveform<Int16>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error. Casting `null` to `IWaveform<Int16>` provides the strong typing necessary to select the correct IVI.NET overload of the Fetch Waveform method. The method will allocate a new waveform with an appropriate extent for the current configuration of the driver. The new waveform is returned to the client. The driver may allocate more memory than necessary for the data array if the larger size has the potential to provide some present or future efficiency benefit, that is, the Capacity may exceed the ValidPointCount.
- Set the waveform parameter to an instance of the waveform object with zero sized data. This permits the client to choose the concrete class that implements the Waveform but defer to the driver for the size and creation of the data array. This may result in sub-optimal performance if the waveform/spectrum was not of the class that the driver prefers. If the data array is not of a supported size or type, the driver shall throw the Invalid Waveform Data Type or Invalid Spectrum Data Type exception. The driver is permitted to allocate new memory or use memory from an existing source for the data array.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

.NET Method Prototype

```
IWaveform<Int16> Channels[].Measurement.FetchWaveform (
                                            IWaveform<Int16> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.FetchWaveformInt16 (
                                     [in, out] SAFEARRAY(short)* WaveformArray,
                                     [in, out] __int64 * ActualPoints,
                                     [in, out] __int64 * FirstValidPoint,
                                     [in, out] double* InitialXOffset,
                                     [in, out] double* InitialXTimeSeconds,
                                     [in, out] double* InitialXTimeFraction,
                                     [in, out] double* XIncrement,
                                     [in, out] double* ScaleFactor,
                                     [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_FetchWaveformInt16 (ViSession Vi,
                                          ViConstString ChannelName,
                                          ViInt64 WaveformArraySize,
                                          ViInt16 WaveformArray[],
                                          ViInt64* ActualPoints,
                                          ViInt64* FirstValidPoint,
                                          ViReal64* InitialXOffset,
                                          ViReal64* InitialXTimeSeconds,
                                          ViReal64* InitialXTimeFraction,
                                          ViReal64* XIncrement,
                                          ViReal64* ScaleFactor,
                                          ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`

`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<Int16>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver. IWaveform<Int16>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViInt16[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need tc shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`

`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<Int16>`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

This function can also return these additional class-defined status codes:

- Channel Not Enabled
- Incompatible Fetch

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled
- Incompatible Fetch

4.3.8 Fetch Waveform Int32

Description

This function operates identically to the Fetch Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details.

.NET Method Prototype

```
IWaveform<Int32> Channels[].Measurement.FetchWaveform (
                                             IWaveform<Int32> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.FetchWaveformInt32 (
                                     [in, out] SAFEARRAY(long)* WaveformArray,
                                     [in, out] __int64 * ActualPoints,
                                     [in, out] __int64 * FirstValidPoint,
                                     [in, out] double* InitialXOffset,
                                     [in, out] double* InitialXTimeSeconds,
                                     [in, out] double* InitialXTimeFraction,
                                     [in, out] double* XIncrement,
                                     [in, out] double* ScaleFactor,
                                     [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_FetchWaveformInt32 (ViSession Vi,
                                          ViConstString ChannelName,
                                          ViInt64 WaveformArraySize,
                                          ViInt32 WaveformArray[],
                                          ViInt64* ActualPoints,
                                          ViInt64* FirstValidPoint,
                                          ViReal64* InitialXOffset,
                                          ViReal64* InitialXTimeSeconds,
                                          ViReal64* InitialXTimeFraction,
                                          ViReal64* XIncrement,
                                          ViReal64* ScaleFactor,
                                          ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`

`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<Int32>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver.IWaveform<Int32>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViInt32[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`

`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<Int32>`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return these additional class-defined status codes:

- Channel Not Enabled
- Incompatible Fetch

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled
- Incompatible Fetch

4.3.9 Fetch Waveform Int8

Description

This function operates identically to the Fetch Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details.

.NET Method Prototype

```
IWaveform<SByte> Channels[].Measurement.FetchWaveform (
                                            IWaveform<SByte> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.FetchWaveformInt8 (
                                     [in, out] SAFEARRAY(BYTE)* WaveformArray,
                                     [in, out] __int64 * ActualPoints,
                                     [in, out] __int64 * FirstValidPoint,
                                     [in, out] double* InitialXOffset,
                                     [in, out] double* InitialXTimeSeconds,
                                     [in, out] double* InitialXTimeFraction,
                                     [in, out] double* XIncrement,
                                     [in, out] double* ScaleFactor,
                                     [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_FetchWaveformInt8 (ViSession Vi,
                                         ViConstString ChannelName,
                                         ViInt64 WaveformArraySize,
                                         ViInt8 WaveformArray[],
                                         ViInt64* ActualPoints,
                                         ViInt64* FirstValidPoint,
                                         ViReal64* InitialXOffset,
                                         ViReal64* InitialXTimeSeconds,
                                         ViReal64* InitialXTimeFraction,
                                         ViReal64* XIncrement,
                                         ViReal64* ScaleFactor,
                                         ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`

`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<SByte>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver.IWaveform<SByte>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViInt8[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`

`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<SByte>`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

This function can also return these additional class-defined status codes:
- Channel Not Enabled
- Incompatible Fetch

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:
- Channel Not Enabled
- Incompatible Fetch

4.3.10 Fetch Waveform Real64

Description

This function operates identically to the Fetch Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details. Note that for this function, after completion each element in the `WaveformArray` parameter is the actual sampled voltage in Volts.

.NET Method Prototype

```
IWaveform<Double> Channels[].Measurement.FetchWaveform (
                                             IWaveform<Double> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.FetchWaveformReal64 (
                                    [in, out] SAFEARRAY(double)* WaveformArray,
                                    [in, out] __int64 * ActualPoints,
                                    [in, out] __int64 * FirstValidPoint,
                                    [in, out] double* InitialXOffset,
                                    [in, out] double* InitialXTimeSeconds,
                                    [in, out] double* InitialXTimeFraction,
                                    [in, out] double* XIncrement);
```

C Prototype

```
ViStatus IviDigitizer_FetchWaveformReal64 (ViSession Vi,
                                           ViConstString ChannelName,
                                           ViInt64 WaveformArraySize,
                                           ViReal64 WaveformArray[],
                                           ViInt64* ActualPoints,
                                           ViInt64* FirstValidPoint,
                                           ViReal64* InitialXOffset,
                                           ViReal64* InitialXTimeSeconds,
                                           ViReal64* InitialXTimeFraction,
                                           ViReal64* XIncrement);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`

`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<Double>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver. IWaveform<Double>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViReal64[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`

`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<Double>`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

This function can also return these additional class-defined status codes:
- Channel Not Enabled
- Incompatible Fetch

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:
- Channel Not Enabled
- Incompatible Fetch

4.3.11 Get Channel Name (IVI-C Only)

Description

This function returns the specific driver defined channel name that corresponds to the one-based index that the user specifies. If the driver defines a qualified channel name, this function returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the Channel Count attribute, the function returns an empty string in the `Name` parameter and returns the Invalid Value error.

.NET Method Prototype

`N/A`

(Use the `Channels[].Name` property)

COM Method Prototype

`N/A`

(Use the `Channels.Item().Name` property)

C Prototype

```
ViStatus IviDigitizer_GetChannelName (ViSession Vi,
                                      ViInt32 ChannelIndex,
                                      ViInt32 ChannelNameBufferSize,
                                      ViChar ChannelName[]);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`ChannelIndex`	A one-based index that defines which name to return	`ViInt32`
`ChannelNameBufferSize`	The number of bytes in the ViChar array that the use specifies for the Source`Name` parameter.	`ViInt32`

Outputs	Description	Base Type
`ChannelName`	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the driver stores the channel name. The caller may pass `VI_NULL` for this parameter if the `ChannelNameBufferSize` parameter is 0.	`ViChar[]`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.12 Get Trigger Source Name (IVI-C Only)

Description

This function returns the specific driver defined trigger source name that corresponds to the one-based index that the user specifies. If the driver defines a qualified trigger source name, this function returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the Trigger Source Count attribute, the function returns an empty string in the `Name` parameter and returns the Invalid Value error.

.NET Method Prototype

```
N/A
```

(Use the `Trigger.Sources[].Name` property)

COM Method Prototype

```
N/A
```

(Use the `Trigger.Sources.Item().Name` property)

C Prototype

```
ViStatus IviDigitizer_GetTriggerSourceName (ViSession Vi,
                                             ViInt32 SourceIndex,
                                             ViInt32 SourceNameBufferSize,
                                             ViChar SourceName[]);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`SourceIndex`	A one-based index that defines which name to return	`ViInt32`
`SourceNameBufferSize`	The number of bytes in the ViChar array that the user specifies for the Source`Name` parameter.	`ViInt32`

Outputs	Description	Base Type
`SourceName`	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the driver stores the trigger sourcename. The caller may pass `VI_NULL` for this parameter if the Source`NameBufferSize` parameter is 0.	`ViChar[]`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.13 Initiate Acquisition

Description

This function initiates a waveform acquisition. After calling this function, the digitizer leaves the idle state and waits for a trigger. The digitizer acquires a waveform for each channel the end-user has enabled with the Configure Channel function.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

.NET Method Prototype

```
void Acquisition.Initiate ();
```

COM Method Prototype

```
HRESULT Acquisition.Initiate ();
```

C Prototype

```
ViStatus IviDigitizer_InitiateAcquisition (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

4.3.14 Is Idle (IVI-C Only)

Description

This function is used to determine if the digitizer is currently in the Idle state.

.NET Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsIdle` property)

COM Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsIdle` property)

C Prototype

```
ViStatus IviDigitizer_IsIdle (ViSession Vi,
                              ViInt32* Status);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Outputs	Description	Base Type
`Status`	Returns whether the digitizer is currently in the Idle state. If the driver cannot query the instrument to determine its state, the driver returns the value Unknown..	`ViInt32`

Defined Values for the Status parameter

Name	*Description*		
		Language	***Identifier***
True	The digitizer is currently in the Idle state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
False	The digitizer is not currently in the Idle state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Idle state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.15 Is Measuring (IVI-C Only)

Description

This function is used to determine if the digitizer is currently in the Measuring state.

.NET Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsMeasuring` property)

COM Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsMeasuring` property)

C Prototype

```
ViStatus IviDigitizer_IsMeasuring (ViSession Vi,
                                   ViInt32* Status);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Outputs	Description	Base Type
`Status`	Returns whether the digitizer is currently in the Measuring state. If the driver cannot query the instrument to determine its state, the driver returns the value Unknown.	`ViInt32`

Defined Values for the Status parameter

Name	*Description*		
		Language	*Identifier*
True	The digitizer is currently in the Measuring state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
False	The digitizer is not currently in the Measuring state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Measuring state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.16 Is Waiting For Arm (IVI-C Only)

Description

This function is used to determine if the digitizer is currently in the Waiting For Arm state.

.NET Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsWaitingForArm` property)

COM Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsWaitingForArm` property)

C Prototype

```
ViStatus IviDigitizer_IsWaitingForArm (ViSession Vi,
                                       ViInt32* Status);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Outputs	Description	Base Type
`Status`	Returns whether the digitizer is currently in the Waiting For Arm state. If the driver cannot query the instrument to determine its state, the driver returns the value Unknown.	`ViInt32`

Defined Values for the Status parameter

Name	*Description*		
		Language	*Identifier*
True	The digitizer is currently in the Waiting For Arm state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
False	The digitizer is not currently in the Waiting For Arm state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Waiting For Arm state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.17 Is Waiting For Trigger (IVI-C Only)

Description

This function is used to determine if the digitizer is currently in the Waiting For Trigger state.

.NET Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsWaitingForTrigger` property)

COM Method Prototype

```
N/A
```

(Use the `Acquistion.Status.IsWaitingForTrigger` property)

C Prototype

```
ViStatus IviDigitizer_IsWaitingForTrigger (ViSession Vi,
                                           ViInt32* Status);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Outputs	Description	Base Type
`Status`	Returns whether the digitizer is currently in the WaitingForTrigger state. If the driver cannot query the instrument to determine its state, the driver returns the value Unknown.	`ViInt32`

Defined Values for the Status parameter

Name	*Description*		
		Language	*Identifier*
True	The digitizer is currently in the Waiting For Trigger state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`
False	The digitizer is not currently in the Waiting For Trigger state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`
Unknown	The driver cannot query the instrument to determine if the digitizer is in the Waiting For Trigger state.		
		C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.18 Query Min Waveform Memory (IVI-C and IVI-COM Only)

Description

In IVI-C, the waveform buffer memory must be allocated in advance by the user. This function is used to determine the minimum amount of memory that is needed to fetch or read data from the digitizer with maximum performance. The returned value includes the memory needed to handle DMA alignment issues and any internal memory that is used by the digitizer hardware or the driver. The parameters to this function are similar to the parameters used in Read and Fetch functions. Users should call this function before allocating data buffer memory, and then call a Read or Fetch function with the same parameter values.

Note that this function will return a value that can be used to allocate the optimally-sized memory buffer for Read and Fetch calls with the same passed parameters. If the Read and Fetch calls specify fewer data points, the data buffer will still be large enough and no performance penalty will be realized (aside from wasted memory space). If the Read and Fetch calls specify more data points they will simply fill the allocated memory buffer as fully as possible. Subsequent Fetch calls can then be made to retrieve the remaining data.

In IVI-COM, the Read and Fetch calls will allocate the proper amount of memory automatically. Instead of passing an array into the function calls, simply pass a pointer to a NULL SAFEARRAY. The driver will then allocate sufficient memory automatically. It is also allowed to pass a pointer to a non-NULL (user allocated, or driver allocated from a previous call) SAFEARRAY, in which case the driver shall not reallocate memory. In this case, the user should call the QueryMinWaveform method to determine the necessary SAFEARRAY size. For best performance, i.e. to avoid allocating the waveform buffer memory for every Read or Fetch call, the user should let the driver allocate the SAFEARRAY buffer memory by passing a pointer to a NULL SAFEARRAY on the first call, ensuring sufficient memory is allocated. Then the same SAFEARRAY should be reused for subsequent calls, until acquisition parameters changes require a larger buffer size.

.NET Method Prototype

```
N/A
```

(Use the appropriate Acquisition.CreateWaveformXX or Acquisition.CreateWaveformCollectionXX method.)COM Method Prototype

```
HRESULT Acquisition.QueryMinWaveformMemory ([in] long DataWidth,
                                             [in] __int64 NumRecords,
                                             [in] __int64 OffsetWithinRecord,
                                             [in] __int64 NumPointsPerRecord,
                                             [out, retval] __int64* NumSamples);
```

C Prototype

```
ViStatus IviDigitizer_QueryMinWaveformMemory (ViSession Vi,
                                              ViInt32 DataWidth,
                                              ViInt64 NumRecords,
                                              ViInt64 OffsetWithinRecord,
                                              ViInt64 NumPointsPerRecord,
                                              ViInt64* NumSamples);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle.	ViSession

`DataWidth`	Specifies the size of the sampled data that will be retrieved. The value of this parameter must be 8, 16, 32 or 64, corresponding to the intended Read/Fetch function.	`ViInt32`
`NumRecords`	Specifies the number of records that will be read.	`ViInt64`
`OffsetWithinRecord`	Specifies the start index within the record from which the data should be retrieved.	`ViInt64`
`NumPointsPerRecord`	Specifies the number of data points to return.	`ViInt64`

Outputs	Description	Base Type
`NumSamples`	Returns the minimum buffer size in samples needed for a subsequent Read or Fetch call with the same readout parameters.	`ViInt64`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

4.3.19 Read Waveform Int16

Description

This function initiates an acquisition on the channels that the end-user configures with the Configure Channel function. If the channel is not enabled for the acquisition, this function returns Channel Not Enabled error. It then waits for the acquisition to complete, and returns the waveform for the channel the end-user specifies. If the digitizer did not complete the acquisition within the time period the user specified with the `MaxTimeMilliseconds` parameter, the function returns the Max Time Exceeded error.

You call the Fetch Waveform function to obtain the waveforms for each of the remaining enabled channels without initiating another acquisition. After this function executes, each element in the `WaveformArray` parameter is an unscaled value directly from the digitizer's analog-to-digital converter (ADC).

The behavior is different for IVI-C/IVI-COM and IVI.NET as follows:

IVI-C/IVI-COM: After this function executes, each element in the `WaveformArray` parameter is an unscaled value directly from the digitizer's analog-to-digital converter (ADC).

IVI.NET: For .NET the return value of IWaveform<Int16> is a waveform object. Refer to Section 4, *Common Properties and Methods of Waveform and Spectrum Interfaces*, and Section 5, *IWaveform<T> Interface*, of *IVI-3.18: IVI.NET Utility Classes and Interfaces Specification,* for the definition of the IWaveform object and information regarding its use. In particular, refer to Section 4.2, *How to use Waveform and Spectrum Types*, in *IVI-3.18: IVI.NET Utility Classes and Interfaces Specification*, for more information about how to implement these methods.

For IVI.NET, the waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform. To allocate memory during the call to this method, one of the following approaches may be used:

- Set the waveform parameter to `(IWaveform<Int16>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error. Casting `null` to `IWaveform<Int16>` provides the strong typing necessary to select the correct IVI.NET overload of the Fetch Waveform method. The method will allocate a new waveform with an appropriate extent for the current configuration of the driver. The new waveform is returned to the client. The driver may allocate more memory than necessary for the data array if the larger size has the potential to provide some present or future efficiency benefit, that is, the Capacity may exceed the ValidPointCount.
- Set the waveform parameter to an instance of the waveform object with zero sized data. This permits the client to choose the concrete class that implements the Waveform but defer to the driver for the size and creation of the data array. This may result in sub-optimal performance if the waveform/spectrum was not of the class that the driver prefers. If the data array is not of a supported size or type, the driver shall throw the Invalid Waveform Data Type or Invalid Spectrum Data Type exception. The driver is permitted to allocate new memory or use memory from an existing source for the data array.

If the Num Records attribute set for the acquisition is not equal to 1, then the Read function shall return an error. You must use the Initiate Acquisition function and the Fetch Multi-Record Waveform functions in this case.

.NET Method Prototype

```
IWaveform<Int16> Channels[].Measurement.ReadWaveform (
                                             PrecisionTimeSpan maxTime,
                                             IWaveform<Int16> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.ReadWaveformInt16 (
                                   [in] long MaxTimeMilliseconds,
                                   [in, out] SAFEARRAY(short)* WaveformArray,
                                   [in, out] __int64* ActualPoints,
                                   [in, out] __int64* FirstValidPoint,
                                   [in, out] double* InitialXOffset,
                                   [in, out] double* InitialXTimeSeconds,
                                   [in, out] double* InitialXTimeFraction,
                                   [in, out] double* XIncrement,
                                   [in, out] double* ScaleFactor,
                                   [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_ReadWaveformInt16 (ViSession Vi,
                                         ViConstString ChannelName,
                                         ViInt32 MaxTimeMilliseconds,
                                         ViInt64 WaveformArraySize,
                                         ViInt16 WaveformArray[],
                                         ViInt64* ActualPoints,
                                         ViInt64* FirstValidPoint,
                                         ViReal64* InitialXOffset,
                                         ViReal64* InitialXTimeSeconds,
                                         ViReal64* InitialXTimeFraction,
                                         ViReal64* XIncrement,
                                         ViReal64* ScaleFactor,
                                         ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
Vi	Instrument handle.	ViSession
ChannelName	Name of the channel from which to retrieve the data.	ViConstString
MaxTimeMilliseconds (C/COM)	Specifies the maximum time the end-user allows for this method to complete in milliseconds. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	ViInt32
maxTime (.NET)	Specifies the maximum time the end-user allows for this method to complete. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	Ivi.Driver. PrecisionTimeSpan
WaveformArraySize (C)	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM, the driver will allocate the memory buffer, unless a non-empty SAFEARRAY is passed as input. In the latter case, the driver shall not reallocate a memory buffer.	ViInt64

`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<Int16>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver.IWaveform<Int16>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C. Units for the individual array elements are unscaled ADC values.	`ViInt16[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`

`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<Int16>`

Defined Values for the MaxTimeMilliseconds Parameter (C/COM)

Name	***Description***		
		Language	***Identifier***
Immediate	The function returns immediately. If no valid data is available, the function returns an error.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`
		COM	`IviDigitizerTimeOutImmediate`
Infinite	The function waits indefinitely for the acquisition to complete before returning the data.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`
		COM	`IviDigitizerTimeOutInfinite`

Defined Values for the maxTime Parameter (.NET)

Name	***Description***		
		Language	***Identifier***
Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		.NET	`PrecisionTimeSpan.Zero`
Infinite	The function waits indefinitely for the measurement to complete.		
		.NET	`PrecisionTimeSpan.MaxValue`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return these additional class-defined status codes:

- Channel Not Enabled
- Incompatible Fetch
- Max Time Exceeded

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled
- Incompatible Fetch

4.3.20 Read Waveform Int32

Description

This function operates identically to the Read Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details.

.NET Method Prototype

```
IWaveform<Int32> Channels[].Measurement.ReadWaveform (
                                             PrecisionTimeSpan maxTime,
                                             IWaveform<Int32> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.ReadWaveformInt32 (
                                   [in] long MaxTimeMilliseconds,
                                   [in, out] SAFEARRAY(long)* WaveformArray,
                                   [in, out] __int64* ActualPoints,
                                   [in, out] __int64* FirstValidPoint,
                                   [in, out] double* InitialXOffset,
                                   [in, out] double* InitialXTimeSeconds,
                                   [in, out] double* InitialXTimeFraction,
                                   [in, out] double* XIncrement,
                                   [in, out] double* ScaleFactor,
                                   [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_ReadWaveformInt32 (ViSession Vi,
                                         ViConstString ChannelName,
                                         ViInt32 MaxTimeMilliseconds,
                                         ViInt64 WaveformArraySize,
                                         ViInt32 WaveformArray[],
                                         ViInt64* ActualPoints,
                                         ViInt64* FirstValidPoint,
                                         ViReal64* InitialXOffset,
                                         ViReal64* InitialXTimeSeconds,
                                         ViReal64* InitialXTimeFraction,
                                         ViReal64* XIncrement,
                                         ViReal64* ScaleFactor,
                                         ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`MaxTimeMilliseconds (C/COM)`	Specifies the maximum time the end-user allows for this method to complete in milliseconds. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`ViInt32`
`maxTime (.NET)`	Specifies the maximum time the end-user allows for this method to complete. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`Ivi.Driver.PrecisionTimeSpan`

`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM, the driver will allocate the memory buffer, unless a non-empty SAFEARRAY is passed as input. In the latter case, the driver shall not reallocate a memory buffer.	`ViInt64`
`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<Int32>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver. IWaveform<Int32>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C. Units for the individual array elements are unscaled ADC values.	`ViInt32[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`

`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<Int32>`

Defined Values for the MaxTimeMilliseconds Parameter (C/COM)

Name	*Description*		
		Language	*Identifier*
Immediate	The function returns immediately. If no valid data is available, the function returns an error.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`
		COM	`IviDigitizerTimeOutImmediate`
Infinite	The function waits indefinitely for the acquisition to complete before returning the data.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`
		COM	`IviDigitizerTimeOutInfinite`

Defined Values for the maxTime Parameter (.NET)

Name	*Description*		
		Language	*Identifier*
Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		`.NET`	`PrecisionTimeSpan.Zero`
Infinite	The function waits indefinitely for the measurement to complete.		
		`.NET`	`PrecisionTimeSpan.MaxValue`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled
- Incompatible Fetch
- Max Time Exceeded

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled
- Incompatible Fetch

4.3.21 Read Waveform Int8

Description

This function operates identically to the Read Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details.

.NET Method Prototype

```
IWaveform<SByte> Channels[].Measurement.ReadWaveform (
                                             PrecisionTimeSpan maxTime,
                                             IWaveform<SByte> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.ReadWaveformInt8 (
                                   [in] long MaxTimeMilliseconds,
                                   [in, out] SAFEARRAY(BYTE)* WaveformArray,
                                   [in, out] __int64* ActualPoints,
                                   [in, out] __int64* FirstValidPoint,
                                   [in, out] double* InitialXOffset,
                                   [in, out] double* InitialXTimeSeconds,
                                   [in, out] double* InitialXTimeFraction,
                                   [in, out] double* XIncrement,
                                   [in, out] double* ScaleFactor,
                                   [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_ReadWaveformInt8 (ViSession Vi,
                                        ViConstString ChannelName,
                                        ViInt32 MaxTimeMilliseconds,
                                        ViInt64 WaveformArraySize,
                                        ViInt8 WaveformArray[],
                                        ViInt64* ActualPoints,
                                        ViInt64* FirstValidPoint,
                                        ViReal64* InitialXOffset,
                                        ViReal64* InitialXTimeSeconds,
                                        ViReal64* InitialXTimeFraction,
                                        ViReal64* XIncrement,
                                        ViReal64* ScaleFactor,
                                        ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`MaxTimeMilliseconds (C/COM)`	Specifies the maximum time the end-user allows for this method to complete in milliseconds. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`ViInt32`
`maxTime (.NET)`	Specifies the maximum time the end-user allows for this method to complete. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`Ivi.Driver.PrecisionTimeSpan`

`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM, the driver will allocate the memory buffer, unless a non-empty SAFEARRAY is passed as input. In the latter case, the driver shall not reallocate a memory buffer.	`ViInt64`
`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<SByte>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver.IWaveform<SByte>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C. Units for the individual array elements are unscaled ADC values.	`ViInt8[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`

`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<SByte>`

Defined Values for the MaxTimeMilliseconds Parameter (C/COM)

Name	***Description***		
		Language	***Identifier***
Immediate	The function returns immediately. If no valid data is available, the function returns an error.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`
		COM	`IviDigitizerTimeOutImmediate`
Infinite	The function waits indefinitely for the acquisition to complete before returning the data.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`
		COM	`IviDigitizerTimeOutInfinite`

Defined Values for the maxTime Parameter (.NET)

Name	***Description***		
		Language	***Identifier***
Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		`.NET`	`PrecisionTimeSpan.Zero`

Infinite	The function waits indefinitely for the measurement to complete.		
		`.NET`	`PrecisionTimeSpan.MaxValue`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled
- Incompatible Fetch
- Max Time Exceeded

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled
- Incompatible Fetch

4.3.22 Read Waveform Real64

Description

This function operates identically to the Read Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details. Note that for this function, after completion each element in the `WaveformArray` parameter is the actual sampled voltage in Volts.

.NET Method Prototype

```
IWaveform<Double> Channels[].Measurement.ReadWaveform (
                                             PrecisionTimeSpan maxTime,
                                             IWaveform<Double> waveform);
```

COM Method Prototype

```
HRESULT Channels.Item().Measurement.ReadWaveformReal64 (
                                   [in] long MaxTimeMilliseconds,
                                   [in, out] SAFEARRAY(double)* WaveformArray,
                                   [in, out] __int64* ActualPoints,
                                   [in, out] __int64* FirstValidPoint,
                                   [in, out] double* InitialXOffset,
                                   [in, out] double* InitialXTimeSeconds,
                                   [in, out] double* InitialXTimeFraction,
                                   [in, out] double* XIncrement);
```

C Prototype

```
ViStatus IviDigitizer_ReadWaveformReal64 (ViSession Vi,
                                          ViConstString ChannelName,
                                          ViInt32 MaxTimeMilliseconds,
                                          ViInt64 WaveformArraySize,
                                          ViReal64 WaveformArray[],
                                          ViInt64* ActualPoints,
                                          ViInt64* FirstValidPoint,
                                          ViReal64* InitialXOffset,
                                          ViReal64* InitialXTimeSeconds,
                                          ViReal64* InitialXTimeFraction,
                                          ViReal64* XIncrement);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`MaxTimeMilliseconds (C/COM)`	Specifies the maximum time the end-user allows for this method to complete in milliseconds. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`ViInt32`
`maxTime (.NET)`	Specifies the maximum time the end-user allows for this method to complete. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`PrecisionTimeSpan`

`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM, the driver will allocate the memory buffer, unless a non-empty SAFEARRAY is passed as input. In the latter case, the driver shall not reallocate a memory buffer.	`ViInt64`
`waveform (.NET)`	The waveform object into which the measurement data is stored. The waveform memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform object using the Create Waveform method and set the waveform parameter to that waveform object. To allocate memory during the call to this method, set the waveform parameter to `(IWaveform<Double>)null`. Note that this is critically different than setting waveform to `null`, which generates a build error. The method will also allocate memory during the call if the waveform parameter is set to a waveform object with zero sized data.	`IviDriver.IWaveform<Double>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C. Units for the individual array elements are unscaled ADC values.	`ViReal64[]`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument.	`ViInt64`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array. This value will often be zero. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates.	`ViInt64`

`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event.	`ViReal64`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time a which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time.	`ViReal64`
`XIncrement (C/COM)`	The time between points in the acquired waveform ir seconds.	`ViReal64`
`Return Value (.NET)`	A waveform object containing the measurement data.	`Ivi.Driver. IWaveform<Double>`

Defined Values for the MaxTimeMilliseconds Parameter (C/COM)

Name	***Description***		
		Language	***Identifier***
Immediate	The function returns immediately. If no valid data is available, the function returns an error.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`
		COM	`IviDigitizerTimeOutImmediate`
Infinite	The function waits indefinitely for the acquisition to complete before returning the data.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`
		COM	`IviDigitizerTimeOutInfinite`

Defined Values for the maxTime Parameter (.NET)

Name	***Description***		
		Language	***Identifier***
Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		.NET	`PrecisionTimeSpan.Zero`
Infinite	The function waits indefinitely for the measurement to complete.		
		.NET	`PrecisionTimeSpan.MaxValue`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Chanel Not Enabled
- Incompatible Fetch
- Max Time Exceeded

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled
- Incompatible Fetch

4.3.23 Wait For Acquisition Complete

Description

Waits until the configured acquisition is complete. If no acquisition is currently running, this function returns immediately. If the acquisition does not complete within the time period the user specified with the MaxTimeMilliseconds parameter, the function returns the Max Time Exceeded error.

.NET Method Prototype

```
void Acquisition.WaitForAcquisitionComplete (PrecisionTimeSpan maxTime);
```

COM Method Prototype

```
HRESULT Acquisition.WaitForAcquisitionComplete ([in] long MaxTimeMilliseconds);
```

C Prototype

```
ViStatus IviDigitizer_WaitForAcquisitionComplete (ViSession Vi,
                                                  ViInt32 MaxTimeMilliseconds);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`MaxTimeMilliseconds (C/COM)`	Specifies the maximum time the end-user allows for this function to complete. The units are milliseconds. The values of Immediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`ViInt32`
`maxTime (.NET)`	Specifies the maximum time the end-user allows for this method to complete. The values of Imediate and Infinite, as defined in IviDigitizer Function Parameter Value Definitions, are also allowed.	`Ivi.Driver. PrecisionTimeSpan`

Defined Values for the MaxTimeMilliseconds Parameter (C/COM)

Name	*Description*		
		Language	*Identifier*
Immediate	The function returns immediately. If no valid data is available, the function returns an error.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`
		COM	`IviDigitizerTimeOutImmediate`
Infinite	The function waits indefinitely for the acquisition to complete.		
		C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`
		COM	`IviDigitizerTimeOutInfinite`

Defined Values for the maxTime Parameter (.NET)

Name	*Description*		
		Language	*Identifier*
Immediate	The function returns immediately. If no valid measurement value exists, the function returns an error.		
		.NET	`PrecisionTimeSpan.Zero`
Infinite	The function waits indefinitely for the measurement to complete.		
		.NET	`PrecisionTimeSpan.MaxValue`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Max Time Exceeded

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

Note that the .NET MaxTimeExceededException is defined in *IVI-3.2: Inherent Capabilities Specification*.

4.4 IviDigitizerBase Behavior Model

The following behavior diagram illustrates the IviDigitizerBase group capability behavior.

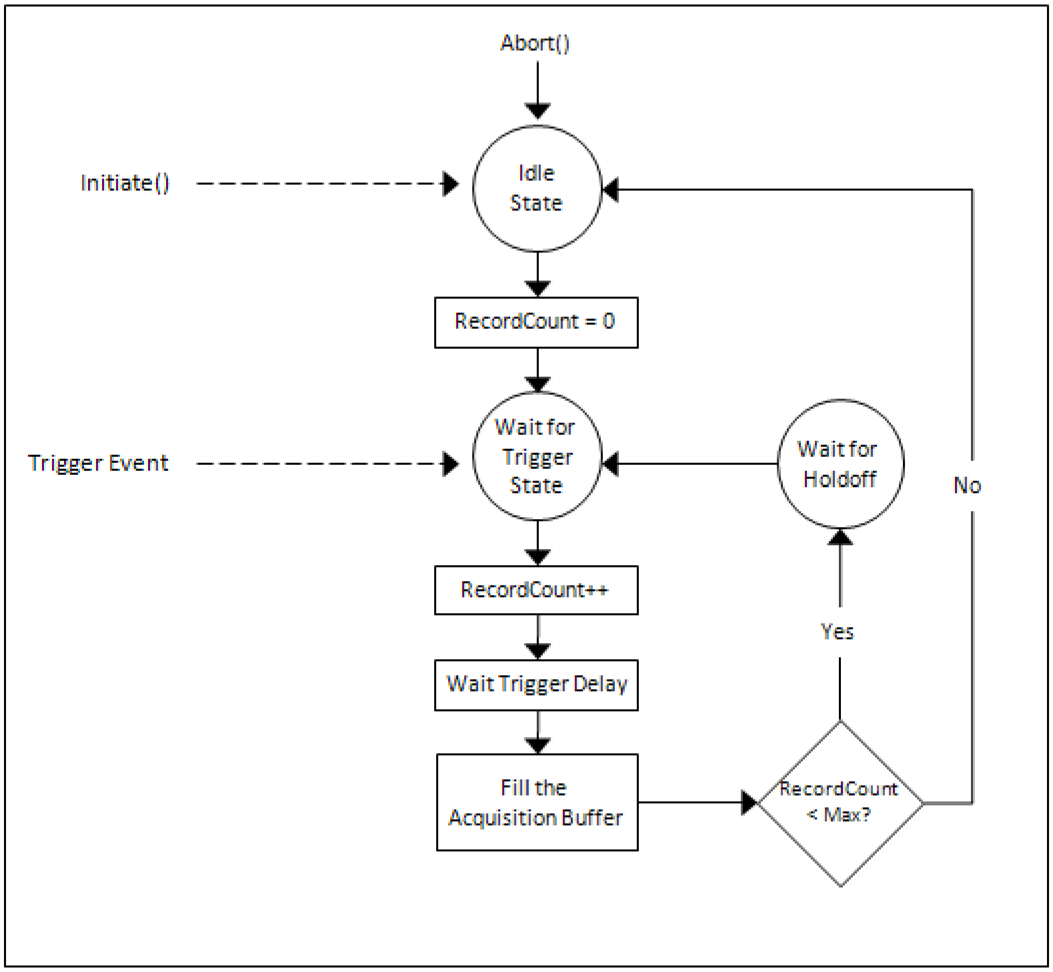


Figure 4-12 IviDigitizerBase Behavior Model

Typically the user configures the digitizer while it is in the *Idle* state. You can configure the digitizer by calling the high-level configure channel, configure acquisition type, configure acquisition record, configure trigger, and configure edge trigger source functions as well as by directly calling the appropriate channel, acquisition and trigger sub-system attributes.

To acquire waveforms, the IviDigitizer class presents high-level read waveform functions, as well as the low-level functions initiate acquisition, acquisition status, fetch waveform, and abort.

The Read Waveform functions initiate a waveform acquisition and returns the acquired waveform after the digitizer has returned to the *Idle* state.

The Initiate Acquisition, Fetch Waveform, and Abort functions give the user lower-level control over the measurement process. Initiate Acquisition initiates a waveform acquisition and moves the instrument into the *Wait-For-Trigger* state. The type of trigger is configured with the Trigger sub-system attributes or with the configure edge trigger source function.

Multiple versions of the Read and Fetch functions are provided to accommodate digitizers of various native data sizes for each waveform sample. Waveforms can be retrieved as arrays of 8-bit, 16-bit, and 32-bit integers as well as 32-bit and 64-bit floating-point numbers. For the data sizes not natively supported by the digitizer, the driver shall automatically convert the data to the appropriate format. Even though this conversion may reduce performance, it shall be supported for interchangeability. Manufacturers shall mention in their documentation the conversion rules they use (e.g. whether the data bits are LSB- or MSB-justified). The Scale Factor and Scale Offset output parameters shall also allow converting the integer values into Volts. If the data type is smaller (i.e. uses less bits) than the digitizer's native type, the most significant bits shall be kept.

If the trigger delay is negative, the first point in the waveform record occurs prior to the trigger event. When the trigger event occurs, the waveform record contains the amount of pre-trigger data that corresponds to the trigger delay. The digitizer leaves the *Wait-for-Trigger* state and acquires the remaining points in the waveform record.

If the trigger delay equals zero, the first point in the waveform record occurs at the time of the trigger event. When the trigger event occurs, the digitizer leaves the *Wait-for-Trigger* state and acquires all the points in the waveform record.

If the acquisition start time is greater than zero, the first point in the waveform record occurs after the trigger event. When the trigger event occurs, the digitizer leaves the *Wait-for-Trigger* state, waits a length of time that is equal to the trigger delay, and acquires all the points in the waveform record.

If the digitizer has acquired the requested number of records, it returns to the *Idle* state. However, if the digitizer must acquire additional records to satisfy the Num Records requirement specified by the user, it then moves to the *Wait-For-Samples* state. The digitizer then waits until the hold-off time expires before moving to the *Wait-For-Trigger*.

Note that the hold-off time is measured from the moment the digitizer exits the *Wait-for-Trigger* state, not from the moment when the digitizer enters the *Wait-for-Samples* state.

After the instrument meets its acquisition complete criterion, the digitizer returns to the *Idle* state. (This criterion is typically 95-98% of the acquisition record; there may be instrument specific attributes that allow you to configure the completion criterion.) You can use the Acquisition Status function to determine if the acquisition is complete or is still in progress.

The Fetch Waveform functions are used to return a waveform from a previously initiated measurement. The Read Waveform and Fetch Waveform functions return the following parameters:

- a waveform array
- the time of the first point in the waveform array in relationship to the trigger event
- the index of the index of the first valid point in the waveform record
- the number of valid points in the waveform record

5. IviDigitizerMultiRecordAcquisition Extension Group

5.1 IviDigitizerMultiRecordAcquisition Overview

The IviDigitizerMultiRecordAcquisition extension group supports digitizers with the ability to perform multi-record acquisitions, and/or fetch of partial records. If the number of waveform records set for the acquisition is greater than 1, the acquisition shall be initiated with the Initiate Acquisition function, and the data shall be retrieved with the Fetch Multi-Record Waveform functions once the acquisition is complete.

5.2 IviDigitizerMultiRecordAcquisition Waveform Collection (IVI.NET Only)

For IVI.NET, the IviDigitizerMultiRecordAcquisition extension group defines the IWaveformCollection<T> interface for returning multi-record waveforms from the Digitizer.

Note that the IVI Foundation does not provide an implementation of IWaveformCollection<T>. Each specific IVI.NET IviDigitizer instrument driver implements CreateWaveform Collection and the Fetch Multi-Record Waveform methods based on an implementation of IWaveformCollection<T> that is suitable to the driver. In addition to implementing IWaveformCollection<T>, such implementations will also need to implement methods and properties that allow the driver to allocate memory in a manner that the driver can use, and to manage the collection (Add, Remove, Clear, etc.).

IWaveformCollection<T> derives from IEnumerable<T> and IEnumerable.

The IWaveformCollection<T> interface defines the following properties:

- Item (IVI.NET Only)
- ValidWaveformCount

This section describes the behavior and requirements of each property.

5.2.1 Item

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	RO	N/A	None	N/A

.NET Property Name

```
IWaveform<T> this [Int64 index]
```

COM Property Name

```
N/A
```

C Constant Name

```
N/A
```

Description

Returns the waveform at the specified index from the collection. The index must refer to a valid, measured waveform, and the range of indexes accepted is 0 to ValidWaveformCount – 1.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

5.2.2 Valid Waveform Count

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	RO	N/A	None	N/A

.NET Property Name

`IWaveformCollection<T>.ValidWaveformCount`

COM Property Name

`N/A`

C Constant Name

`N/A`

Description

The number of waveform objects in the collection which contain multi-record measurement data from the instrument.

.NET Exceptions

Barring errors that may be generated by the .NET runtime, this property shall always succeed.

5.3 IviDigitizerMultiRecordAcquisition Functions

The IviDigitizerMultiRecordAcquisition extension group defines the following functions:

- Create Waveform Collection (IVI.NET Only)
- Fetch Multi-Record Waveform Int16
- Fetch Multi-Record Waveform Int32
- Fetch Multi-Record Waveform Int8
- Fetch Multi-Record Waveform Real64

This section describes the behavior and requirements of each function.

5.3.1 Create Waveform Collection (IVI.NET Only)

Description

This function creates an array of waveform objects and shall allocate the necessary memory to transfer each waveform in the array from the instrument to the host.

If numberOfWaveforms is zero, the driver shall allocate the number of Waveforms based on the current driver configuration.

If sizePerWaveform is zero, the driver shall allocate the memory for each waveform memory with a size based on the current driver configuration.

No waveforms measurements are stored in the waveform collection immediately after returning from this method, and the value of IWaveformCollection<T>.ValidWaveformCount shall be 0.

.NET Method Prototype

```
IWaveformCollection<Double> Acquisition.CreateWaveformCollectionDouble (
                               Int64 numberOfWaveforms,
                               Int64 sizePerWaveform);
IWaveformCollection<Int32> Acquisition.CreateWaveformCollectionInt32 (
                               Int64 numberOfWaveforms,
                               Int64 sizePerWaveform);
IWaveformCollection<Int16> Acquisition.CreateWaveformCollectionInt16 (
                               Int64 numberOfWaveforms,
                               Int64 sizePerWaveform);
IWaveformCollection<SByte> Acquisition.CreateWaveformCollectionSByte (
                               Int64 numberOfWaveforms,
                               Int64 sizePerWaveform);
```

COM Method Prototype

```
N/A
```

C Prototype

```
N/A
```

Parameters

Inputs	Description	Base Type
`numberOfWaveforms`	The number of waveforms in the array of waveforms.	`Int64`
`sizePerWaveform`	The number of elements in each waveform.	`Int64`

Outputs	Description	Base Type
`Return value (.NET)`	The newly allocated multi-record waveform.	`Ivi.Driver.IWaveform<T>[]`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

5.3.2 Fetch Multi-Record Waveform Int16

Description

This function returns the (multi-record) waveform the digitizer acquired for the specified channel. The waveform is from a previously initiated acquisition.

The acquired waveform records can be retrieved all together, or in chunks. You specify the first record and the number of consecutive records to fetch. Note that the record number is zero-based, and reset when you initiate a new acquisition, i.e. record 0 is the *first* record of the *last* acquisition. The return parameter `ActualRecords` indicates how many of the requested records have actually completed successfully. The return parameters `ActualPoints`, `FirstValidPoint`, `InitialXOffset`, `InitialXTimeSeconds` and `InitialXTimeFraction` have a value for each record, and therefore are arrays of size `NumRecords`. However, if the value of `ActualRecords` is smaller than the requested number of records, these arrays shall have only their first `ActualRecords` elements valid. If a NULL pointer is passed in for any of these parameters, the driver shall ignore it. Note that the value of `FirstValidPoint` is relative to the start of the waveform array, such that:

```
Sample i of record R = WaveformArray[FirstValidPoint[R]+i]
```

You must use the Initiate Acquisition function to start a multi-record (Num Record > 1) acquisition on the channels that the end-user configures with the Configure Channel function. The digitizer acquires waveforms on the concurrently enabled channels. If the channel is not enabled for the acquisition, this function returns the Channel Not Enabled error.

You can use the Acquisition Status function to determine when the acquisition is complete. You must call the FetchMultiRecordWaveformInt16 function separately for each enabled channel to obtain the waveforms. Alternatively, you can use the Wait For Acquisition Complete function to block the calling program until the acquisition is finished.

The behavior is different for IVI-C/IVI-COM and IVI.NET as follows:

IVI-C/IVI-COM: After this function executes, each element in the `WaveformArray` parameter is an unscaled value directly from the digitizer's analog-to-digital converter (ADC).

IVI.NET: For .NET the return value of IWaveformCollection<Int16> is a waveform collection object. Refer to Section 4, *Common Properties and Methods of Waveform and Spectrum Interfaces*, and Section 5, *IWaveform<T> Interface*, of *IVI-3.18: IVI.NET Utility Classes and Interfaces Specification,* for the definition of the IWaveform object and information regarding its use.

The waveform collection memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform collection object using the Create Waveform Collection method and set the waveforms parameter to that waveform collection. To allocate memory during the call to this method, set the waveforms parameter to `(IWaveformCollection<Int16>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error. Casting `null` to `IWaveformCollection<Int16>` provides the strong typing necessary to select the correct IVI.NET overload of the Fetch Multi-Record Waveform method.

For .NET, if the waveforms parameter is null, the driver will create a waveform collection object and will determine the number of waveforms in the collection, and the size of each waveform, based on the current settings of the instrument and the method parameters.

The IVI.NET return value maps to C and COM out parameters as follows:

- The C/COM WaveformArray parameter is an array of values returned from the instrument that includes values for all of the waveforms being returned. In .NET, these values are stored in the individual waveforms in the waveform collection.
- The C/COM ActualRecords parameter corresponds to the waveform collection ValidWaveformCount property.
- The C/COM ActualPoints parameter is an array of sizes, one per waveform being returned. In .NET, these values are represented by the ValidPointCount property of each individual waveform.
- The C/COM FirstValidPoint parameter is an array of first valid point values, one per waveform being returned. In .NET, these values are represented by the FirstValidPoint property of each individual waveform.
- The C/COM InitialXOffset parameter is an array of offsets between the trigger time and the time of the first point in a waveform, one per waveform being returned. In .NET, these values are calculated for each individual waveform by taking the difference between the waveform Start Time property and TriggerTime property.
- The C/COM InitialXTimeSeconds and InitialXTimeFraction parameters are both arrays that, when corresponding values are combined, yield the start time of a wavefrom, one time (seconds + fractional seconds) per waveform being returned. In .NET, these values are represented by the StartTime property of each individual waveform.
- The C/COM XIncrement parameter is the time span separating each measured point, and is the same for all of waveforms being returned. In .NET, this value is represented by the IntervalPerPoint property of each individual waveform. In .NET, all of the waveforms in a waveform collection shall have the same value for IntervalPerPoint.
- The C/COM ScaleFactor parameter is the scale factor that must be applied (along with the scale offset) to each measured point to yield the actual, real value for the point. In .NET, this value is represented by the Scale property of each individual waveform. In .NET, all of the waveforms in a waveform collection shall have the same value for Scale.
- The C/COM ScaleOffset parameter is the scale offset that must be applied (along with the scale factor) to each measured point to yield the actual, real value for the point. In .NET, this value is represented by the Offset property of each individual waveform. In .NET, all of the waveforms in a waveform collection shall have the same value for Offset.

This function does not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. Call the Error Query function at the conclusion of the sequence to check the instrument status.

.NET Method Prototype

```
IWaveformCollection<Int16>
            Channels[].MultiRecordMeasurement.FetchMultiRecordWaveform (
                             Int64 firstRecord,
                             Int64 numberOfRecords,
                             Int64 offsetWithinRecord,
                             Int64 numberOfPointsPerRecord,
                             IWaveformCollection<Int16> waveforms);
```

COM Method Prototype

```
HRESULT Channels.Item().MultiRecordMeasurement.FetchMultiRecordWaveformInt16 (
                             [in] __int64 FirstRecord,
                             [in] __int64 NumRecords,
                             [in] __int64 OffsetWithinRecord,
                             [in] __int64 NumPointsPerRecord,
                             [in, out] SAFEARRAY(short)* WaveformArray,
                             [in, out] __int64 * ActualRecords,
                             [in, out] SAFEARRAY(__int64)* ActualPoints,
                             [in, out] SAFEARRAY(__int64)* FirstValidPoint,
                             [in, out] SAFEARRAY(double)* InitialXOffset,
                             [in, out] SAFEARRAY(double)* InitialXTimeSeconds,
                             [in, out] SAFEARRAY(double)* InitialXTimeFraction,
                             [in, out] double* XIncrement,
                             [in, out] double* ScaleFactor,
                             [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_FetchMultiRecordWaveformInt16 (ViSession Vi,
                                               ViConstString ChannelName,
                                               ViInt64 FirstRecord,
                                               ViInt64 NumRecords,
                                               ViInt64 OffsetWithinRecord,
                                               ViInt64 NumPointsPerRecord,
                                               ViInt64 WaveformArraySize,
                                               ViInt16 WaveformArray[],
                                               ViInt64* ActualRecords,
                                               ViInt64 ActualPoints[],
                                               ViInt64 FirstValidPoint[],
                                               ViReal64 InitialXOffset[],
                                               ViReal64 InitialXTimeSeconds[],
                                               ViReal64 InitialXTimeFraction[],
                                               ViReal64* XIncrement,
                                               ViReal64* ScaleFactor,
                                               ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`FirstRecord`	Specifies the number of the first record to read.	`ViInt64`

`NumRecords (C/COM)`
`NumberOfRecords (.NET)`
	Specifies the number of consecutive records to read. If NumRecords is greater than 1, this parameter allows full or partial (if OffsetWithinRecord and NumPointsPerRecord are specified accordingly) data to be retrieved from multiple digitizer records in a single Fetch call. If NumRecords is less than or equal to zero, an error will be returned.	`ViInt64`

`OffsetWithinRecord`	Specifies the start index within the record from which the data should be retrieved. While normally zero, this parameter allows users to retrieve partial records. Data that comes before the OffsetWithinRecord index will not be retrieved. This is perhaps most useful when retrieving very large data records because it allows records to be retrieved in several smaller chunks.	`ViInt64`

`NumPointsPerRecord (C/COM)` `NumberOfPointsPerRecord (.NET)`	Specifies the number of data points per record to return This number may be larger than the amount of data available. Use the ActualPoints parameter to determine how many data points were returned.	`ViInt64`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`
`waveforms (.NET)`	A Waveform collection object with a particular number of waveforms, each with data of a particular size needed only for reusing waveform object across reads. The waveform collection memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform collection object using the Create Waveform Collection method and set the waveforms parameter to that waveform collection. To allocate memory during the call to this method, set the waveforms parameter to `(IWaveformCollection<Int16>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error.	`IWaveformColle tion<Int16>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViInt16[]`
`ActualRecords (C/COM)`	Indicates how many records in the acquisition completed successfully. The arrays `ActualPoints`, `FirstValidPoint`, `InitialXOffset`, `InitialXTimeSeconds` and `InitialXTimeFraction` have the corresponding first contiguous values valid.	`ViInt64`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument for each completed record. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`

`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array for each completed record. This valu will often be simply the record index (zero to ActualRecords-1) times NumPointsPerRecord. However, some digitizer hardware designs transfer dat most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`XIncrement (C/COM)`	The time between points in the acquired waveform in seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`

`Return Value (.NET)`	A waveform collection object with data from the channel. (In .NET, this is the return value of the method.) (The *IVI-3.2: Inherent Capabilities Specification* defines the IwaveformCollection interface.)	`IWaveformCollec tion<Int16>`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled

5.3.3 Fetch Multi-Record Waveform Int32

Description

This function operates identically to the Fetch Multi-Record Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details.

.NET Method Prototype

```
IWaveformCollection<Int32>
            Channels[].MultiRecordMeasurement.FetchMultiRecordWaveform (
                             Int64 firstRecord,
                             Int64 numberOfRecords,
                             Int64 offsetWithinRecord,
                             Int64 numberOfPointsPerRecord,
                             IWaveformCollection<Int32> waveforms);
```

COM Method Prototype

```
HRESULT Channels.Item().MultiRecordMeasurement.FetchMultiRecordWaveformInt32 (
                             [in] __int64 FirstRecord,
                             [in] __int64 NumRecords,
                             [in] __int64 OffsetWithinRecord,
                             [in] __int64 NumPointsPerRecord,
                             [in, out] SAFEARRAY(long)* WaveformArray,
                             [in, out] __int64 * ActualRecords,
                             [in, out] SAFEARRAY(__int64)* ActualPoints,
                             [in, out] SAFEARRAY(__int64)* FirstValidPoint,
                             [in, out] SAFEARRAY(double)* InitialXOffset,
                             [in, out] SAFEARRAY(double)* InitialXTimeSeconds,
                             [in, out] SAFEARRAY(double)* InitialXTimeFraction,
                             [in, out] double* XIncrement,
                             [in, out] double* ScaleFactor,
                             [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_FetchMultiRecordWaveformInt32 (ViSession Vi,
                                                     ViConstString ChannelName,
                                                     ViInt64 FirstRecord,
                                                     ViInt64 NumRecords,
                                                     ViInt64 OffsetWithinRecord,
                                                     ViInt64 NumPointsPerRecord,
                                                     ViInt64 WaveformArraySize,
                                                     ViInt32 WaveformArray[],
                                                     ViInt64* ActualRecords,
                                                     ViInt64 ActualPoints[],
                                                     ViInt64 FirstValidPoint[],
                                                     ViReal64 InitialXOffset[],
                                                     ViReal64 InitialXTimeSeconds[],
                                                     ViReal64 InitialXTimeFraction[],
                                                     ViReal64* XIncrement,
                                                     ViReal64* ScaleFactor,
                                                     ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`

`FirstRecord`	Specifies the number of the first record to read.	`ViInt64`
`NumRecords (C/COM)` `NumberOfRecords (.NET)`	Specifies the number of consecutive records to read. If NumRecords is greater than 1, this parameter allows full or partial (if OffsetWithinRecord and NumPointsPerRecord are specified accordingly) data t be retrieved from multiple digitizer records in a single Fetch call. If NumRecords is less than or equal to zero, an error will be returned.	`ViInt64`
`OffsetWithinRecord`	Specifies the start index within the record from which the data should be retrieved. While normally zero, this parameter allows users to retrieve partial records. Data that comes before the OffsetWithinRecord index will not be retrieved. This is perhaps most useful when retrieving very large data records because it allows records to be retrieved in several smaller chunks.	`ViInt64`
`NumPointsPerRecord (C/COM)` `NumberOfPointsPerRecord (.NET)`	Specifies the number of data points per record to return This number may be larger than the amount of data available. Use the ActualPoints parameter to determine how many data points were returned.	`ViInt64`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`
`waveforms (.NET)`	A Waveform collection object with a particular number of waveforms, each with data of a particular size needed only for reusing waveform object across reads. The waveform collection memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform collection object using the Create Waveform Collection method and set the waveforms parameter to that waveform collection. To allocate memory during the call to this method, set the waveforms parameter to `(IWaveformCollection<Int32>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error.	`IWaveformCollection<Int32>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViInt32[]`
`ActualRecords (C/COM)`	Indicates how many records in the acquisition completed successfully. The arrays `ActualPoints`, `FirstValidPoint`, `InitialXOffset`, `InitialXTimeSeconds` and `InitialXTimeFraction` have the corresponding first contiguous values valid.	`ViInt64`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument for each completed record. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array for each completed record. This value will often be simply the record index (zero to ActualRecords-1) times NumPointsPerRecord. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`

`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`XIncrement (C/COM)`	The time between points in the acquired waveform in seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform collection object with data from the channel. (In .NET, this is the return value of the method.) (The IVI-3.2: Inherent Capabilities Specification defines the IwaveformCollection interface.)	`IWaveformColle tion<Int32>`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled

5.3.4 Fetch Multi-Record Waveform Int8

Description

This function operates identically to the Fetch Multi-Record Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details.

.NET Method Prototype

```
IWaveformCollection<SByte>
            Channels[].MultiRecordMeasurement.FetchMultiRecordWaveform (
                              Int64 firstRecord,
                              Int64 numberOfRecords,
                              Int64 offsetWithinRecord,
                              Int64 numberOfPointsPerRecord,
                              IWaveformCollection<SByte> waveforms);
```

COM Method Prototype

```
HRESULT Channels.Item().MultiRecordMeasurement.FetchMultiRecordWaveformInt8 (
                              [in] __int64 FirstRecord,
                              [in] __int64 NumRecords,
                              [in] __int64 OffsetWithinRecord,
                              [in] __int64 NumPointsPerRecord,
                              [in, out] SAFEARRAY(BYTE)* WaveformArray,
                              [in, out] __int64 * ActualRecords,
                              [in, out] SAFEARRAY(__int64)* ActualPoints,
                              [in, out] SAFEARRAY(__int64)* FirstValidPoint,
                              [in, out] SAFEARRAY(double)* InitialXOffset,
                              [in, out] SAFEARRAY(double)* InitialXTimeSeconds,
                              [in, out] SAFEARRAY(double)* InitialXTimeFraction,
                              [in, out] double* XIncrement,
                              [in, out] double* ScaleFactor,
                              [in, out] double* ScaleOffset);
```

C Prototype

```
ViStatus IviDigitizer_FetchMultiRecordWaveformInt8 (ViSession Vi,
                                                   ViConstString ChannelName,
                                                   ViInt64 FirstRecord,
                                                   ViInt64 NumRecords,
                                                   ViInt64 OffsetWithinRecord,
                                                   ViInt64 NumPointsPerRecord,
                                                   ViInt64 WaveformArraySize,
                                                   ViInt8 WaveformArray[],
                                                   ViInt64* ActualRecords,
                                                   ViInt64 ActualPoints[],
                                                   ViInt64 FirstValidPoint[],
                                                   ViReal64 InitialXOffset[],
                                                   ViReal64 InitialXTimeSeconds[],
                                                   ViReal64 InitialXTimeFraction[],
                                                   ViReal64* XIncrement,
                                                   ViReal64* ScaleFactor,
                                                   ViReal64* ScaleOffset);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`

`FirstRecord`	Specifies the number of the first record to read.	`ViInt64`

`NumRecords (C/COM)`
`NumberOfRecords (.NET)`
	Specifies the number of consecutive records to read. If NumRecords is greater than 1, this parameter allows full or partial (if OffsetWithinRecord and NumPointsPerRecord are specified accordingly) data to be retrieved from multiple digitizer records in a single Fetch call. If NumRecords is less than or equal to zero, an error will be returned.	`ViInt64`

`OffsetWithinRecord`	Specifies the start index within the record from which the data should be retrieved. While normally zero, this parameter allows users to retrieve partial records. Data that comes before the OffsetWithinRecord index will not be retrieved. This is perhaps most useful when retrieving very large data records because it allows records to be retrieved in several smaller chunks.	`ViInt64`

`NumPointsPerRecord (C/COM)`
`NumberOfPointsPerRecord (.NET)`
	Specifies the number of data points per record to return This number may be larger than the amount of data available. Use the ActualPoints parameter to determine how many data points were returned.	`ViInt64`

`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`
`waveforms (.NET)`	A Waveform collection object with a particular number of waveforms, each with data of a particular size needed only for reusing waveform object across reads. The waveform collection memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform collection object using the Create Waveform Collection method and set the waveforms parameter to that waveform collection. To allocate memory during the call to this method, set the waveforms parameter to `(IWaveformCollection<SByte>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error.	`IWaveformCollection<SByte>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViInt8[]`
`ActualRecords (C/COM)`	Indicates how many records in the acquisition completed successfully. The arrays `ActualPoints`, `FirstValidPoint`, `InitialXOffset`, `InitialXTimeSeconds` and `InitialXTimeFraction` have the corresponding first contiguous values valid.	`ViInt64`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument for each completed record. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array for each completed record. This value will often be simply the record index (zero to ActualRecords-1) times NumPointsPerRecord. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`

`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`XIncrement (C/COM)`	The time between points in the acquired waveform in seconds.	`ViReal64`
`ScaleFactor (C/COM)`	Scaling factor for the waveform data.	`ViReal64`
`ScaleOffset (C/COM)`	Scaling offset for the waveform data.	`ViReal64`
`Return Value (.NET)`	A waveform collection object with data from the channel. (In .NET, this is the return value of the method.) (The IVI-3.2: Inherent Capabilities Specification defines the IwaveformCollection interface.)	`IWaveformCollection<SByte>`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled

5.3.5 Fetch Multi-Record Waveform Real64

Description

This function operates identically to the Fetch Multi-Record Waveform Int16, with the only difference being the data type of the returned waveform array. Please see the definition of that function for details. Note that for this function, after completion each element in the `WaveformArray` parameter is the actual sampled voltage in Volts.

.NET Method Prototype

```
IWaveformCollection<Double>
            Channels[].MultiRecordMeasurement.FetchMultiRecordWaveform (
                             Int64 firstRecord,
                             Int64 numberOfRecords,
                             Int64 offsetWithinRecord,
                             Int64 numberOfPointsPerRecord,
                             IWaveformCollection<Double> waveforms);
```

COM Method Prototype

```
HRESULT Channels.Item().MultiRecordMeasurement.FetchMultiRecordWaveformReal64 (
                             [in] __int64 FirstRecord,
                             [in] __int64 NumRecords,
                             [in] __int64 OffsetWithinRecord,
                             [in] __int64 NumPointsPerRecord,
                             [in, out] SAFEARRAY(double)* WaveformArray,
                             [in, out] __int64 * ActualRecords,
                             [in, out] SAFEARRAY(__int64)* ActualPoints,
                             [in, out] SAFEARRAY(__int64)* FirstValidPoint,
                             [in, out] SAFEARRAY(double)* InitialXOffset,
                             [in, out] SAFEARRAY(double)* InitialXTimeSeconds,
                             [in, out] SAFEARRAY(double)* InitialXTimeFraction,
                             [in, out] double* XIncrement);
```

C Prototype

```
ViStatus IviDigitizer_FetchMultiRecordWaveformReal64 (ViSession Vi,
                                               ViConstString ChannelName,
                                               ViInt64 FirstRecord,
                                               ViInt64 NumRecords,
                                               ViInt64 OffsetWithinRecord,
                                               ViInt64 NumPointsPerRecord,
                                               ViInt64 WaveformArraySize,
                                               ViReal64 WaveformArray[],
                                               ViInt64* ActualRecords,
                                               ViInt64 ActualPoints[],
                                               ViInt64 FirstValidPoint[],
                                               ViReal64 InitialXOffset[],
                                               ViReal64 InitialXTimeSeconds[],
                                               ViReal64 InitialXTimeFraction[],
                                               ViReal64* XIncrement);
```

Parameters

Inputs	Description	Base Type
`Vi (C)`	Instrument handle.	`ViSession`
`ChannelName (C)`	Name of the channel from which to retrieve the data.	`ViConstString`
`FirstRecord`	Specifies the number of the first record to read.	`ViInt64`

`NumRecords (C/COM)` `NumberOfRecords (.NET)`	Specifies the number of consecutive records to read. If NumRecords is greater than 1, this parameter allows full or partial (if OffsetWithinRecord and NumPointsPerRecord are specified accordingly) data tc be retrieved from multiple digitizer records in a single Fetch call. If NumRecords is less than or equal to zero, an error will be returned.	`ViInt64`
`OffsetWithinRecord`	Specifies the start index within the record from which the data should be retrieved. While normally zero, this parameter allows users to retrieve partial records. Data that comes before the OffsetWithinRecord index will not be retrieved. This is perhaps most useful when retrieving very large data records because it allows records to be retrieved in several smaller chunks.	`ViInt64`
`NumPointsPerRecord (C/COM)` `NumberOfPointsPerRecord (.NET)`	Specifies the number of data points per record to return This number may be larger than the amount of data available. Use the ActualPoints parameter to determine how many data points were returned.	`ViInt64`
`WaveformArraySize (C)`	Specifies the allocated size of the WaveformArray buffer, in number of data points. If this value is smaller than the total number of points to be retrieved, the driver will fill the waveform buffer as fully as possible and return the actual number of points retrieved in the ActualPoints parameter. In IVI-COM it is the driver that allocates the memory buffer, unless a non-NULL SAFEARRAY is passed.	`ViInt64`
`waveforms (.NET)`	A Waveform collection object with a particular number of waveforms, each with data of a particular size needed only for reusing waveform object across reads. The waveform collection memory may be allocated before calling this method, or during the call to this method. To allocate memory before calling this method, create a waveform collection object using the Create Waveform Collection method and set the waveforms parameter to that waveform collection. To allocate memory during the call to this method, set the waveforms parameter to `(IWaveformCollection<Double>)null`. Note that this is critically different than setting waveforms to `null`, which generates a build error.	`IWaveformCollec tion<Double>`

Outputs	Description	Base Type
`WaveformArray (C/COM)`	Buffer into which the acquired waveform is stored. For IVI-C, this buffer is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated. To have the driver allocate the buffer, the user passes in a valid pointer to a NULL SAFEARRAY. Note that this is critically different than passing in a NULL pointer, which generates an error. When IVI-COM users pass in a pointer to a non-NULL SAFEARRAY, the driver fills the user-allocated array in the same fashion as with IVI-C.	`ViReal64[]`
`ActualRecords (C/COM)`	Indicates how many records in the acquisition completed successfully. The arrays `ActualPoints`, `FirstValidPoint`, `InitialXOffset`, `InitialXTimeSeconds` and `InitialXTimeFraction` have the corresponding first contiguous values valid.	`ViInt64`
`ActualPoints (C/COM)`	Indicates how many data points were actually retrieved from the instrument for each completed record. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`FirstValidPoint (C/COM)`	Indicates the index of the first valid data point in the output Data array for each completed record. This value will often be simply the record index (zero to ActualRecords-1) times NumPointsPerRecord. However, some digitizer hardware designs transfer data most efficiently when the data is aligned with specific memory address boundaries. In those cases, the hardware may return a few invalid data points at the beginning of a record. This eliminates the need to shift the data during the transfer, ensuring maximum data transfer rates. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViInt64[]`
`InitialXOffset (C/COM)`	The time in relation to the Trigger Event of the first point in the waveform in seconds. Negative values mean that the first point in the waveform array was acquired before the trigger event. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`

`InitialXTimeSeconds (C/COM)`	Specifies the seconds portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`InitialXTimeFraction (C/COM)`	Specifies the fractional portion of the absolute time at which the first data point was acquired. Note that the actual time is the sum of InitialXTimeSeconds and InitialXTimeFraction. The time is specified as the sum of two values because a single double-precision floating-point number does not have sufficient range and resolution to specify the time. This is an array of size at least `NumRecords` or a NULL pointer. For IVI-C, this array is always user allocated. For IVI-COM, this buffer may either be user allocated or driver allocated, with the same rules as for `WaveformArray`.	`ViReal64[]`
`XIncrement (C/COM)`	The time between points in the acquired waveform in seconds.	`ViReal64`
`Return Value (.NET)`	A waveform collection object with data from the channel. (In .NET, this is the return value of the method.) (The IVI-3.2: Inherent Capabilities Specification defines the IwaveformCollection interface.)	`IWaveformColle tion<Double>`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

This method can also throw these additional class-defined exceptions:

- Channel Not Enabled

5.4 IviDigitizerMultiRecordAcquisition Behavior Model

The IviDigitizerMultiRecordAcquisition extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model.*

5.5 IviDigitizerMultiRecordAcquisition Compliance Notes

For a specific driver to comply with the IviDigitizerMultiRecordAcquisition extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the functions listed in this section.

6. IviDigitizerBoardTemperature Extension Group

6.1 IviDigitizerBoardTemperature Overview

The IviDigitizerBoardTemperature extension group supports digitizers with the ability to report the temperature of the device as a whole.

6.2 IviDigitizerBoardTemperature Attributes

The IviDigitizerBoardTemperature extension group defines the following attributes:

- Board Temperature
- Temperature Units

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

6.2.1 Board Temperature

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	RO	N/A	None	Query Board Temperature (IVI-C only)

.NET Property Name

`Temperature.BoardTemperature`

COM Property Name

`Temperature.BoardTemperature`

C Constant Name

`IVIDIGITIZER_ATTR_BOARD_TEMPERATURE`

Description

Indicates the temperature of the entire board. The units are governed by the Temperature Units attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

6.2.2 Temperature Units

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	Configure Temperature Units (IVI-C only)

.NET Property Name

`Temperature.Units`

.NET Enumeration Name

`TemperatureUnits`

COM Property Name

`Temperature.Units`

COM Enumeration Name

`IviDigitizerTemperatureUnitsEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TEMPERATURE_UNITS`

Description

Specifies the temperature units returned by the Board Temperature and the Channel Temperature attributes.

Defined Values

Name	***Description***		
		Language	***Identifier***
Celsius	Temperature values returned from the digitizer are in degrees Celsius.		
		.NET	`TemperatureUnits.Celsius`
		C	`IVIDIGITIZER_VAL_CELSIUS`
		COM	`IviDigitizerTemperatureUnitsCelsius`
Fahrenheit	Temperature values returned from the digitizer are in degrees Fahrenheit.		
		.NET	`TemperatureUnits.Fahrenheit`
		C	`IVIDIGITIZER_VAL_FAHRENHEIT`
		COM	`IviDigitizerTemperatureUnitsFahrenheit`
Kelvin	Temperature values returned from the digitizer are in degrees Kelvin.		
		.NET	`TemperatureUnits.Kelvin`
		C	`IVIDIGITIZER_VAL_KELVIN`
		COM	`IviDigitizerTemperatureUnitsKelvin`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

6.3 IviDigitizerBoardTemperature Functions

The IviDigitizerBoardTemperature extension group defines the following function:

- Configure Temperature Units (IVI-C only)
- Query Board Temperature (IVI-C only)

This section describes the behavior and requirements of this function.

6.3.1 Configure Temperature Units (IVI-C Only)

Description

This function is used to configure the temperature units returned by the Board Temperature and the Channel Temperature attributes.

.NET Method Prototype

```
N/A
```

(Use the `Temperature.BoardTemperature` property)

COM Method Prototype

```
N/A
```

(Use the `Temperature.BoardTemperature` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureTemperatureUnits (ViSession Vi,
                                                  ViInt32 Units);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Units`	Temperature units to be used by the Board Temperature and Channel Temperature attributes. This value sets the Temperature Units attribute. See the attribute description for more details.	`ViInt32`

Defined Values for the Units parameter

Name	***Description***		
		Language	***Identifier***
Celsius	Temperature values returned from the digitizer are in degrees Celsius.		
		C	`IVIDIGITIZER_VAL_CELSIUS`
Fahrenheit	Temperature values returned from the digitizer are in degrees Fahrenheit.		
		C	`IVIDIGITIZER_VAL_FAHRENHEIT`
Kelvin	Temperature values returned from the digitizer are in degrees Kelvin.		
		C	`IVIDIGITIZER_VAL_KELVIN`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

6.3.2 Query Board Temperature (IVI-C Only)

Description

This function is used to query the temperature of the entire board. The units are governed by the Temperature Units attribute.

.NET Method Prototype

```
N/A
```

(Use the `Temperature.BoardTemperature` property)

COM Method Prototype

```
N/A
```

(Use the `Temperature.BoardTemperature` property)

C Prototype

```
ViStatus IviDigitizer_QueryBoardTemperature (ViSession Vi,
                                             ViReal64* Temperature);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Outputs	Description	Base Type
`Temperature`	Returns whether the `current temperature of the entire board. The units are governed by the Temperature Units attribute.`	`ViReal64`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

6.4 IviDigitizerBoardTemperature Behavior Model

The IviDigitizerBoardTemperature extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

6.5 IviDigitizerBoardTemperature Compliance Notes

For a specific driver to comply with the IviDigitizerBoardTemperature extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

7. IviDigitizerChannelFilter Extension Group

7.1 IviDigitizerChannelFilter Overview

The IviDigitizerChannelFilter extension group supports to control the input filter of the digitizer.

7.2 IviDigitizerChannelFilter Attributes

The IviDigitizerChannelFilter extension group defines the following attributes:

- Input Filter Bypass
- Input Filter Max Frequency
- Input Filter Min Frequency

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

7.2.1 Input Filter Bypass

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	Channel	None	N/A

.NET Property Name

`Channels[].Filter.Bypass`

COM Property Name

`Channels.Item().Filter.Bypass`

C Constant Name

`IVIDIGITIZER_ATTR_INPUT_FILTER_BYPASS`

Description

Specifies whether or not to bypass the input filter.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

7.2.2 Input Filter Max Frequency

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	Channel	Up	Configure Input Filter

.NET Property Name

`Channels[].Filter.MaxFrequency`

COM Property Name

`Channels.Item().Filter.MaxFrequency`

C Constant Name

`IVIDIGITIZER_ATTR_INPUT_FILTER_MAX_FREQUENCY`

Description

Specifies the maximum input filter frequency. Specifying a value of zero means that the device should be set to the full bandwidth that the filter can deliver without being bypassed.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

7.2.3 Input Filter Min Frequency

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	Channel	Down	Configure Input Filter

.NET Property Name

`Channels[].Filter.MinFrequency`

COM Property Name

`Channels.Item().Filter.MinFrequency`

C Constant Name

`IVIDIGITIZER_ATTR_INPUT_FILTER_MIN_FREQUENCY`

Description

Specifies the minimum input filter frequency.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

7.3 IviDigitizerChannelFilter Functions

The IviDigitizerChannelFilter extension group defines the following function:

- Configure Input Filter

This section describes the behavior and requirements of this function.

7.3.1 Configure Input Filter

Description

This function is used to configure the minimum and maximum input filter frequencies for a specified channel.

.NET Method Prototype

```
void Channels[].Filter.Configure (Double minFrequency,
                                  Double maxFrequency);
```

COM Method Prototype

```
HRESULT Channels.Item().Filter.Configure ([in] double MinFrequency,
                                          [in] double MaxFrequency);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureInputFilter (ViSession Vi,
                                            ViConstString ChannelName,
                                            ViReal64 MinFrequency,
                                            ViReal64 MaxFrequency);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`ChannelName`	Name of the channel to configure.	`ViConstString`
`MinFrequency`	Specifies the minimum input filter frequency. This value sets the Input Filter Min Frequency attribute. See the attribute description for more details.	`ViReal64`
`MaxFrequency`	Specifies the maximum input filter frequency. This value sets the Input Filter Max Frequency attribute. See the attribute description for more details.	`ViReal64`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

7.4 IviDigitizerChannelFilter Behavior Model

The IviDigitizerChannelFilter extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

7.5 IviDigitizerChannelFilter Compliance Notes

For a specific driver to comply with the IviDigitizerChannelFilter extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

8. IviDigitizerChannelTemperature Extension Group

8.1 IviDigitizerChannelTemperature Overview

The IviDigitizerChannelTemperature extension group supports digitizers with the ability to report the temperature of individual channels.

8.2 IviDigitizerChannelTemperature Attributes

The IviDigitizerChannelTemperature extension group defines the following attributes:

- Channel Temperature

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

8.2.1 Channel Temperature

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	RO	Channel	None	Query Channel Temperature (IVI-C only)

.NET Property Name

```
Channels[].Temperature
```

COM Property Name

```
Channels.Item().Temperature
```

C Constant Name

```
IVIDIGITIZER_ATTR_CHANNEL_TEMPERATURE
```

Description

Indicates the temperature of the channel. The units are governed by the Temperature Units attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

8.3 IviDigitizerChannelTemperature Functions

The IviDigitizerChannelTemperature extension group defines the following function:

- Query Channel Temperature (IVI-C only)

This section describes the behavior and requirements of this function.

8.3.1 Query Channel Temperature (IVI-C Only)

Description

This function is used to query the temperature of a specific channel. The units are governed by the Temperature Units attribute.

.NET Method Prototype

```
N/A
```

(use the `Channels[].Temperature property)`

COM Method Prototype

```
N/A
```

(use the `Channels.Item().Temperature property)`

C Prototype

```
ViStatus IviDigitizer_QueryChannelTemperature (ViSession Vi,
                                               ViConstString ChannelName,
                                               ViReal64* Temperature);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`ChannelName`	Name of the channel from which to read the temperature.	`ViConstString`

Outputs	Description	Base Type
`Temperature`	Returns whether the current temperature of the entire board. The units are governed by the Temperature Units attribute.	`ViReal64`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

8.4 IviDigitizerChannelTemperature Behavior Model

The IviDigitizerChannelTemperature extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

8.5 IviDigitizerChannelTemperature Compliance Notes

For a specific driver to comply with the IviDigitizerChannelTemperature extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

9. IviDigitizerTimeInterleavedChannels Extension Group

9.1 IviDigitizerTimeInterleavedChannels Overview

The IviDigitizerTimeInterleavedChannels extension group supports digitizers with the ability to combine two or more channels to achieve higher sample rates and/or greater memory depth. This is accomplished by routing the signal from a single physical input connector to multiple analog-to-digital converters, with each ADC's sample clock offset from the others. (The other physical input connectors are not used.)

Some multichannel digitizers offer the possibility of interleaving their channels: by combining the A/D converter and/or memory from several channels, interleaving allows the user to multiply the maximum sampling rate when not all input channels are required to be active. The number of channels available for signal acquisition is correspondingly reduced.

Digitizers usually support interleaving of 2 or 4 channels. Combining 2 channels doubles the maximum sampling rate and/or the maximum number of samples that can be acquired on one single input channel. The active channel may or may not be user selectable. For digitizers featuring segmentable memory (multiple records/segments per acquisition), depending on the design of the digitizer the maximum number of records may or may not increase when combining channels.

When combining channels, users should be careful to specify the desired sample rate. Since sample rate is defined separately, combining channels does not automatically change the sample rate – it only changes the maximum available sample rate.

9.2 IviDigitizerTimeInterleavedChannels Attributes

The IviDigitizerTimeInterleavedChannels extension group defines the following attributes:

- Time Interleaved Channel List
- Time Interleaved Channel List Auto

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

9.2.1 Time Interleaved Channel List

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	R/W	Channel	None	Configure Time Interleaved Channel List (IVI-C only)

.NET Property Name

`Channels[].TimeInterleavedChannelList`

COM Property Name

`Channels.Item().TimeInterleavedChannelList`

C Constant Name

`IVIDIGITIZER_ATTR_TIME_INTERLEAVED_CHANNEL_LIST`

Description

This attribute is used to combine this channel with one or more other channels to achieve higher effective sampling rates and/or greater memory depth. The string provided here specifies which channels should operate in combined mode with the current channel. This attribute is a comma-separated list of one or more channel names. Users may specify either physical or virtual repeated capability identifiers in this list. An empty string or VI_NULL can be used to indicate that no channels should be combined (or that none are currently combined, in the case of a query).

Setting this attribute on any channel disables automatic combined mode (Combined Channels Auto is set to False). Querying this attribute when Combined Channels Auto is True returns the list of channels (if any) the digitizer automatically combined with the current channel to satisfy the sample rate requirements. If a channel name specified in the list is not recognized, the driver generates the Unknown Channel Name error. If a channel name specified in the list is not enabled, this attribute generates the error Channel Not Enabled.

Setting this attribute on a channel designates that channel as the one on which subsequent channel-based operations should be made. This includes configuration operations such as setting the Vertical Coupling, Vertical Offset, and Vertical Range attributes, It also includes fetch and read operations.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

9.2.2 Time Interleaved Channel List Auto

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	N/A	None	N/A

.NET Property Name

`Acquisition.TimeInterleavedChannelListAuto`

COM Property Name

`Acquisition.TimeInterleavedChannelListAuto`

C Constant Name

`IVIDIGITIZER_ATTR_TIME_INTERLEAVED_CHANNEL_LIST_AUTO`

Description

Specifies whether or not the instrument should automatically combine enabled channels to satisfy user-specified sample rates. When set to True, the instrument will automatically combine channels to meet the sample rate requirements specified via the Sample Rate attribute. Use the Time Interleaved Channel List attribute to query which channels (if any) have been combined.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

9.3 IviDigitizerTimeInterleavedChannels Functions

The IviDigitizerTimeInterleavedChannels extension group defines the following function:

- Configure Time Interleaved Channel List (IVI-C only)

This section describes the behavior and requirements of this function.

9.3.1 Configure Time Interleaved Channel List (IVI-C Only)

Description

This function is used to combine this channel with one or more other channels to achieve higher effective sampling rates and/or greater memory depth. Calling this function disables automatic combined mode (Combined Channels Auto is set to False).

Calling this function on a channel designates that channel as the one on which subsequent channel-based operations should be made. This includes configuration operations such as setting the Vertical Coupling, Vertical Offset, and Vertical Range attributes. It also includes fetch and read operations.

.NET Method Prototype

```
N/A
```

(use the `Channels[].TimeInterleavedChannelList` property)

COM Method Prototype

```
N/A
```

(use the `Channels.Item().TimeInterleavedChannelList` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureTimeInterleavedChannelList (ViSession Vi,
                                                ViConstString ChannelName,
                                                ViConstString ChannelList);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`ChannelName`	Name of the channel to be combined with those in ChannelList.	`ViConstString`
`ChannelList`	A comma-separated list of one or more channel names to combine with the channel specified by ChannelName. This value sets the Time Interleaved Channel List attribute. See the attribute description for more details.	`ViConstString`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled

9.4 IviDigitizerTimeInterleavedChannels Behavior Model

The IviDigitizerTimeInterleavedChannels extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

9.5 IviDigitizerTimeInterleavedChannels Compliance Notes

For a specific driver to comply with the IviDigitizerTimeInterleavedChannels extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

10. IviDigitizerDataInterleavedChannels Extension Group

10.1 IviDigitizerDataInterleavedChannels Overview

The IviDigitizerDataInterleavedChannels extension group supports digitizers with the ability to interleave data samples from different channels into a single set of data. This feature allows data from multiple digitizer channels to be retured by the digitizer in a single data retrieval operation. The retrieved data values from each combined channel are interleaved so that all data points taken at a single instant follow one another before the next time samples begin.

This feature is most common in digitizers that are designed to read I/Q data. The ‘I’ signal is attached to one digitizer channel and the ‘Q’ data to another. The channels may be combined using the IviDigitizerDataInterleavedChannels extension group so that a single data retrieval call returns complex I/Q data from both channels.

10.2 IviDigitizerDataInterleavedChannels Attributes

The IviDigitizerDataInterleavedChannels extension group defines the following attributes:

- Data Interleaved Channel List

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

10.2.1 Data Interleaved Channel List

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	R/W	Channel	None	Configure Data Interleaved Channel List (IVI-C only)

.NET Property Name

`Channels[].DataInterleavedChannelList`

COM Property Name

`Channels.Item().DataInterleavedChannelList`

C Constant Name

`IVIDIGITIZER_ATTR_DATA_INTERLEAVED_CHANNEL_LIST`

Description

This attribute is used to combine this channel with one or more other channels to interleave the returned data. The string provided here specifies which channels should operate in combined mode with the current channel. This attribute is a comma-separated list of one or more channel names. Users may specify either physical or virtual repeated capability identifiers in this list. An empty string or VI_NULL can be used to indicate that no channels should be combined (or that none are currently combined, in the case of a query).

If a channel name specified in the list is not recognized, the driver generates the Unknown Channel Name error. If a channel name specified in the list is not enabled, this attribute generates the error Channel Not Enabled.

Setting this attribute on a channel designates that channel as the one on which subsequent fetch and read operations should be made. Fetch and read operations will return the data from multiple channels, so care should be taken to ensure that sufficient memory is allocated.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

10.3 IviDigitizerDataInterleavedChannels Functions

The IviDigitizerDataInterleavedChannels extension group defines the following function:

- Configure Data Interleaved Channel List (IVI-C only)

This section describes the behavior and requirements of this function.

10.3.1 Configure Data Interleaved Channel List (IVI-C Only)

Description

This attribute is used to combine this channel with one or more other channels to interleave the returned data. The string provided here specifies which channels should operate in combined mode with the current channel. This attribute is a comma-separated list of one or more channel names. Users may specify either physical or virtual repeated capability identifiers in this list. An empty string or VI_NULL can be used to indicate that no channels should be combined (or that none are currently combined, in the case of a query).

If a channel name specified in the list is not recognized, the driver generates the Unknown Channel Name error. If a channel name specified in the list is not enabled, this attribute generates the error Channel Not Enabled.

Setting this attribute on a channel designates that channel as the one on which subsequent fetch and read operations should be made. Fetch and read operations will return the data from multiple channels, so care should be taken to ensure that sufficient memory is allocated.

.NET Method Prototype

`N/A`

(use the `Channels[].DataInterleavedChannelList` property)

COM Method Prototype

`N/A`

(use the `Channels.Item().DataInterleavedChannelList` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureDataInterleavedChannelList (ViSession Vi,
                                                           ViConstString ChannelName,
                                                           ViConstString ChannelList);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`ChannelName`	Name of the channel to be combined with those in ChannelList.	`ViConstString`
`ChannelList`	A comma-separated list of one or more channel names to combine with the channel specified by ChannelName. This value sets the Data Interleaved Channel List attribute. See the attribute description for more details.	`ViConstString`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Channel Not Enabled

10.4 IviDigitizerDataInterleavedChannels Behavior Model

The IviDigitizerDataInterleavedChannels extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

10.5 IviDigitizerDataInterleavedChannels Compliance Notes

For a specific driver to comply with the IviDigitizerDataInterleavedChannels extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

11. IviDigitizerReferenceOscillator Extension Group

11.1 IviDigitizerReferenceOscillator Overview

The IviDigitizerReferenceOscillator extension group supports digitizers with the ability to use an external reference oscillator.

11.2 IviDigitizerReferenceOscillator Attributes

The IviDigitizerReferenceOscillator extension group defines the following attributes:

- Reference Oscillator External Frequency
- Reference Oscillator Output Enabled
- Reference Oscillator Source

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

11.2.1 Reference Oscillator External Frequency

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	N/A	None	Configure Reference Oscillator

.NET Property Name

`ReferenceOscillator.ExternalFrequency`

COM Property Name

`ReferenceOscillator.ExternalFrequency`

C Constant Name

`IVIDIGITIZER_ATTR_REFERENCE_OSCILLATOR_EXTERNAL_FREQUENCY`

Description

Specifies the frequency of the external signal which is used as a frequency reference. This value is used only if the Reference Oscillator Source attribute is set to External. The units are Hertz.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

11.2.2 Reference Oscillator Output Enabled

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	N/A	None	Configure Reference Oscillator Output Enabled (IVI-C only)

.NET Property Name

`ReferenceOscillator.OutputEnabled`

COM Property Name

`ReferenceOscillator.OutputEnabled`

C Constant Name

`IVIDIGITIZER_ATTR_REFERENCE_OSCILLATOR_OUTPUT_ENABLED`

Description

Specifies whether or not the reference frequency signal appears at an output of the digitizer.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

11.2.3 Reference Oscillator Source

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	Configure Reference Oscillator

.NET Property Name

`ReferenceOscillator.Source`

.NET Enumeration Name

`ReferenceOscillatorSource`

COM Property Name

`ReferenceOscillator.Source`

COM Enumeration Name

`IviDigitizerReferenceOscillatorSourceEnum`

C Constant Name

`IVIDIGITIZER_ATTR_REFERENCE_OSCILLATOR_SOURCE`

Description

Specifies the reference frequency source used.

Defined Values

Name	***Description***		
		Language	***Identifier***
Internal	The internal reference oscillator is used.		
		.NET	`ReferenceOscillatorSource.Internal`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_INTERNAL`
		COM	`IviDigitizerReferenceOscillatorSourceInternal`
External	An external reference oscillator is used.		
		.NET	`ReferenceOscillatorSource.External`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_EXTERNAL`
		COM	`IviDigitizerReferenceOscillatorSourceExternal`
PXIClk10	`The PXI/PXIe backplane 10 MHz reference.`		
		.NET	`ReferenceOscillatorSource.PxiClk10`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXI_CLK10`
		COM	`IviDigitizerReferenceOscillatorSourcePXIClk10`
PXIeClk100	`The PXIe backplane 100 MHz reference.`		
		.NET	`ReferenceOscillatorSource.PxiExpressClk100`

		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXIE_CLK100`
		COM	`IviDigitizerReferenceOscillatorSourcePXIeClk100`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

11.3 IviDigitizerReferenceOscillator Functions

The IviDigitizerReferenceOscillator extension group defines the following functions:

- Configure Reference Oscillator
- Configure Reference Oscillator Output Enabled (IVI-C only)

This section describes the behavior and requirements of this function.

11.3.1 Configure Reference Oscillator

Description

Configures the digitizer's reference oscillator.

.NET Method Prototype

```
void ReferenceOscillator.Configure (ReferenceOscillatorSource source,
                                    Double frequency);
```

COM Method Prototype

```
HRESULT ReferenceOscillator.Configure (
                         [in] IviDigitizerReferenceOscillatorSourceEnum Source,
                         [in] double Frequency);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureReferenceOscillator (ViSession Vi,
                                                     ViInt32 Source,
                                                     ViReal64 Frequency);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the source of the reference frequency signal. The driver uses this value to set the Reference Oscillator Source attribute. See the attribute description for more details.	`ViInt32`
`Frequency`	Specifies the frequency of the external reference oscillator. This parameter is only used if the Source is set to External. The driver uses this value to set the Reference Oscillator Frequency attribute. See the attribute description for more details.	`ViReal64`

Defined Values for the Source Parameter

Name	*Description*		
		Language	*Identifier*
Internal	The internal reference oscillator is used.		
		.NET	`ReferenceOscillatorSource.Internal`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_INTERNAL`
		COM	`IviDigitizerReferenceOscillatorSourceInternal`
External	An external reference oscillator is used.		
		.NET	`ReferenceOscillatorSource.External`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_EXTERNAL`
		COM	`IviDigitizerReferenceOscillatorSourceExternal`
PXIClk10	`The PXI/PXIe backplane 10 MHz reference.`		

		.NET	`ReferenceOscillatorSource.PxiClk10`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXI_CLK10`
		COM	`IviDigitizerReferenceOscillatorSourcePXIClk10`
PXIeClk100	`The PXIe backplane 100 MHz reference.`		
		.NET	`ReferenceOscillatorSource.PxiExpressClk100`
		C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXIE_CLK100`
		COM	`IviDigitizerReferenceOscillatorSourcePXIeClk100`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

11.3.2 Configure Reference Oscillator Output Enabled (IVI-C Only)

Description

Configures whether or not the reference frequency signal appears at an output of the digitizer.

.NET Method Prototype

```
(N/A)
```

(use the `ReferenceOscillator.OutputEnabled` property)

COM Method Prototype

```
(N/A)
```

(use the `ReferenceOscillator.OutputEnabled` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureReferenceOscillatorOutputEnabled(ViSession Vi,
                                                                ViBoolean Enabled);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Enabled`	Specifies whether or not the reference frequency signal appears at a reference output of the digitizer. This value sets the Reference Oscillator Output Enabled attribute. See the attribute description for more details.	`ViBoolean`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

11.4 IviDigitizerReferenceOscillator Behavior Model

The IviDigitizerReferenceOscillator extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

11.5 IviDigitizerReferenceOscillator Compliance Notes

For a specific driver to comply with the IviDigitizerReferenceOscillator extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

12. IviDigitizerSampleClock Extension Group

12.1 IviDigitizerSampleClock Overview

The IviDigitizerSampleClock extension group supports with the ability to use (or provide) an external sample clock.

12.2 IviDigitizerSampleClock Attributes

The IviDigitizerSampleClock extension group defines the following attributes:

- Sample Clock External Divider
- Sample Clock External Frequency
- Sample Clock Source
- Sample Clock Output Enabled

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

12.2.1 Sample Clock External Divider

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	N/A	Down	Configure Sample Clock

.NET Property Name

`SampleClock.ExternalDivider`

COM Property Name

`SampleClock.ExternalDivider`

C Constant Name

`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_EXTERNAL_DIVIDER`

Description

Specifies the value by which the external sample clock should be divided. This value is used only if the Sample Clock Source attribute is set to External.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

12.2.2 Sample Clock External Frequency

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	N/A	None	Configure Sample Clock

.NET Property Name

`SampleClock.ExternalFrequency`

COM Property Name

`SampleClock.ExternalFrequency`

C Constant Name

`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_EXTERNAL_FREQUENCY`

Description

Specifies the frequency of the external signal which is used as a sample clock. This value is used only if the Sample Clock Source attribute is set to External. The units are Hertz.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

12.2.3 Sample Clock Source

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	Configure Sample Clock

.NET Property Name

`SampleClock.Source`

.NET Enumeration Name

`SampleClockSource`

COM Property Name

`SampleClock.Source`

COM Enumeration Name

`IviDigitizerSampleClockSourceEnum`

C Constant Name

`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_SOURCE`

Description

Specifies the clock used to pace acquisition sampling.

Defined Values

Name	*Description*		
		Language	*Identifier*
Internal	The internal sample clock is used.		
		.NET	`SampleClockSource.Internal`
		C	`IVIDIGITIZER_VAL_SAMPLE_CLOCK_SOURCE_INTERNAL`
		COM	`IviDigitizerSampleClockSourceInternal`
External	An external sample clock is used.		
		.NET	`SampleClockSource.External`
		C	`IVIDIGITIZER_VAL_SAMPLE_CLOCK_SOURCE_EXTERNAL`
		COM	`IviDigitizerSampleClockSourceExternal`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

12.2.4 Sample Clock Output Enabled

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	N/A	None	Configure Sample Clock Output Enabled (IVI-C only)

.NET Property Name

`SampleClock.OutputEnabled`

COM Property Name

`SampleClock.OutputEnabled`

C Constant Name

`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_OUTPUT_ENABLED`

Description

Specifies whether or not the sample clock appears at a reference output of the digitizer.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

12.3 IviDigitizerSampleClock Functions

The IviDigitizerSampleClock extension group defines the following functions:

- Configure Sample Clock
- Configure Sample Clock Output Enabled (IVI-C only)

This section describes the behavior and requirements of this function.

12.3.1 Configure Sample Clock

Description

Configures the digitizer's sample clock.

.NET Method Prototype

```
void SampleClock.Configure (SampleClockSource source,
                            Double frequency,
                            Double divider);
```

COM Method Prototype

```
HRESULT SampleClock.Configure (
                         [in] IviDigitizerSampleClockSourceEnum Source,
                         [in] double Frequency,
                         [in] double Divider);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureSampleClock (ViSession Vi,
                                            ViInt32 Source,
                                            ViReal64 Frequency,
                                            ViReal64 Divider);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the source of the sample clock signal. The driver uses this value to set the Sample Clock Source attribute. See the attribute description for more details.	`ViInt32`
`Frequency`	Specifies the frequency of the external sample clock. This parameter is only used if the Sample Clock Source attribute is set to External. The driver uses this value to set the External Sample Clock Frequency attribute. See the attribute description for more details.	`ViReal64`
`Divider`	Specifies the value by which the external sample clock should be divided. This value is used only if the Sample Clock Source attribute is set to External.	`ViReal64`

Defined Values for the Source Parameter

Name	*Description*		
		Language	*Identifier*
Internal	The internal reference oscillator is used.		
		.NET	`SampleClockSource.Internal`
		C	`IVIDIGITIZER_VAL_SAMPLE_CLOCK_SOURCE_INTERNAL`
		COM	`IviDigitizerSampleClockSourceInternal`
External	An external reference oscillator is used.		

		.NET	`SamplesClockSource.External`
		C	`IVIDIGITIZER_VAL_SAMPLE_CLOCK_SOURCE_EXTERNAL`
		COM	`IviDigitizerSampleClockSourceExternal`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

12.3.2 Configure Sample Clock Output Enabled (IVI-C Only)

Description

Configures whether or not the sample clock appears at a reference output of the digitizer.

.NET Method Prototype

```
(N/A)
```

(use the `SampleClock.OutputEnabled` property)

COM Method Prototype

```
(N/A)
```

(use the `SampleClock.OutputEnabled` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureSampleClockOutputEnabled(ViSession Vi,
                                                        ViBoolean Enabled);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Enabled`	Specifies whether or not the sample clock appears at a reference output of the digitizer. This value sets the Sample Clock Output Enabled attribute. See the attribute description for more details.	`ViBoolean`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

12.4 IviDigitizerSampleClock Behavior Model

The IviDigitizerSampleClock extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

12.5 IviDigitizerSampleClock Compliance Notes

For a specific driver to comply with the IviDigitizerSampleClock extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

13. IviDigitizerSampleMode Extension Group

13.1 IviDigitizerSampleMode Overview

In addition to the fundamental capabilities, the IviDigitizerSampleMode extension group supports digitizers with the ability to control whether the digitizer is using real-time or equivalent-time sampling.

13.2 IviDigitizerSampleMode Attributes

The IviDigitizerSampleMode extension group defines the following attributes:

- Sample Mode

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

13.2.1 Sample Mode

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	Configure Sample Mode (IVI-C only)

.NET Property Name

`Acquisition.SampleMode`

.NET Enumeration Name

`SampleMode`

COM Property Name

`Acquisition.SampleMode`

COM Enumeration Name

`IviDigitizerSampleModeEnum`

C Constant Name

`IVIDIGITIZER_ATTR_SAMPLE_MODE`

Description

Specifies whether the digitizer uses real-time or equivalent-time sampling.

Defined Values

Name	***Description***		
		Language	***Identifier***
Real Time	Specifies real-time sampling.		
		.NET	`SampleMode.RealTime`
		C	`IVIDIGITIZER_VAL_SAMPLE_MODE_REAL_TIME`
		COM	`IviDigitizerSampleModeRealTime`
Equivalent Time	Specifies equivalent-time sampling.		
		.NET	`SampleMode.EquivalentTime`
		C	`IVIDIGITIZER_VAL_SAMPLE_MODE_EQUIVALENT_TIME`
		COM	`IviDigitizerSampleModeEquivalentTime`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

13.3 IviDigitizerSampleMode Functions

The IviDigitizerSampleMode extension group defines the following function:

- Configure Sample Mode (IVI-C only)

This section describes the behavior and requirements of this function.

13.3.1 Configure Sample Mode (IVI-C Only)

Description

Configures the digitizer sample mode.

.NET Method Prototype

```
(N/A)
```

(use the `Acquisition.SampleMode` property)

COM Method Prototype

```
(N/A)
```

(use the `Acquisition.SampleMode` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureSampleMode (ViSession Vi,
                                           ViInt32 SampleMode);
```

Parameters

Inputs	**Description**	**Base Type**
`Vi`	Instrument handle.	`ViSession`
`SampleMode`	Specifies the sample mode used by the digitizer. This value sets the Sample Mode attribute. See the attribute description for more details.	`ViInt32`

Defined Values for the Condition Parameter

Name	***Description***		
		Language	***Identifier***
Real Time	Specifies real-time sampling.		
		C	`IVIDIGITIZER_VAL_REAL_TIME`
Equivalent Time	Specifies equivalent-time sampling.		
		C	`IVIDIGITIZER_VAL_EQUIVALENT_TIME`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

13.4 IviDigitizerSampleMode Behavior Model

The IviDigitizerSampleMode extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

13.5 IviDigitizerSampleMode Compliance Notes

For a specific driver to comply with the IviDigitizerSampleMode extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

14. IviDigitizerSelfCalibration Extension Group

14.1 IviDigitizerSelfCalibration Overview

The IviDigitizerSelfCalibration extension group supports digitizers with the ability to perform self calibration.

14.2 IviDigitizerSelfCalibration Functions

The IviDigitizerSelfCalibration extension group defines the following function:

- Self Calibrate

This section describes the behavior and requirements of this function.

14.2.1 Self Calibrate

Description

Executes all internal calibrations. If the digitizer does not support self-calibration, this function silently succeeds.

.NET Method Prototype

```
void Calibration.SelfCalibrate ();
```

COM Method Prototype

```
HRESULT Calibration.SelfCalibrate ();
```

C Prototype

```
ViStatus IviDigitizer_SelfCalibrate (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

14.3 IviDigitizerSelfCalibration Behavior Model

The IviDigitizerSelfCalibration extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.3, *IviDigitizerBase Behavior Model*.

14.4 IviDigitizerSelfCalibration Compliance Notes

For a specific driver to comply with the IviDigitizerSelfCalibration extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

15. IviDigitizerDownconversion Extension Group

15.1 IviDigitizerDownconversion Overview

The IviDigitizerDownconversion extension group supports digitizers with the ability to do frequency translation or downconversion in hardware. This allows the user to only return the frequency-translated data at the potentially lower sample rate. For example, enabling this would allow the digitizer in a Synthetic Instrument system to acquire an IF signal and return I/Q data.

When downconversion is enabled, the digitizer Read and Fetch functions return interleaved data, and thus, twice as many points. When enabled, the Sample Rate attribute specifies the downconverted sample rate not the ADC sample rate.

15.2 IviDigitizerDownconversion Attributes

The IviDigitizerDownconversion extension group defines the following attributes:

- Downconversion Enabled
- Downconversion Center Frequency
- Downconversion IQ Interleaved

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

15.2.1 Downconversion Enabled

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	Channel	None	Configure Downconversion

.NET Property Name

`Channels[].Downconversion.Enabled`

COM Property Name

`Channels.Item().Downconversion.Enabled`

C Constant Name

`IVIDIGITIZER_ATTR_DOWNCONVERSION_ENABLED`

Description

Enables or disables downconversion. When enabled, the Read and Fetch functions return data according to the setting of the Fetch IQ Interleaved Data attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

15.2.2 Downconversion Center Frequency

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	Channel	None	Configure Downconversion

.NET Property Name

`Channels[].Downconversion.CenterFrequency`

COM Property Name

`Channels.Item().Downconversion.CenterFrequency`

C Constant Name

`IVIDIGITIZER_ATTR_DOWNCONVERSION_CENTER_FREQUENCY`

Description

Specifies the center frequency, in Hz, from which the digitizer should downconvert.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

15.2.3 Downconversion IQ Interleaved

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	Channel	None	N/A

.NET Property Name

`Channels[].Downconversion.IQInterleaved`

COM Property Name

`Channels.Item().Downconversion.IQInterleaved`

C Constant Name

`IVIDIGITIZER_ATTR_DOWNCONVERSION_IQ_INTERLEAVED`

Description

Controls how the Read and Fetch functions return data when downconversion is enabled. When this attribute is True and downconversion is enabled, the data returned from the Read and Fetch functions is interleaved I-Q data points. When this attribute is False and downconversion is enabled, the Read and Fetch functions return data with all I data points in sequence followed by all Q data points. This attribute has no effect when the Downconversion Enabled attribute is False.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

15.3 IviDigitizerDownconversion Functions

The IviDigitizerDownconversion extension group defines the following function:

- Configure Downconversion

This section describes the behavior and requirements of this function.

15.3.1 Configure Downconversion

Description

Configures how the digitizer performs downconversion.

.NET Method Prototype

```
void Channels[].Downconversion.Configure (Boolean enabled,
                                          Double centerFrequency);
```

COM Method Prototype

```
HRESULT Channels.Item().Downconversion.Configure ([in] VARIANT_BOOL Enabled,
                                                  [in] double CenterFrequency);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureDownconversion (ViSession Vi,
                                               ViConstString ChannelName,
                                               ViBoolean Enabled,
                                               ViReal64 CenterFrequency);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`ChannelName`	Name of the channel to configure.	`ViConstString`
`Enabled`	Enables or disables downversion. The driver uses this value to set the Downconversion Enabled attribute. See the attribute description for more details.	`ViBoolean`
`CenterFrequency`	Specifies the center frequency, in Hz, from which the digitizer should downconvert. The driver uses this value to set the Downconversion Center Frequency attribute. See the attribute description for more details.	`ViReal64`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

16. IviDigitizerArm Extension Group

16.1 IviDigitizerArm Overview

The IviDigitizerArm extension group supports instruments that can arm the digitizer based on an edge detected in a source signal. Digitizers that support arming can typically arm on the same kinds of events on which it can trigger. Consequently, the attributes and functions in the IviDigitizerArm mimic those describing edge triggering in the trigger sub-system of the IviDigitizerBase capability group.

16.2 IviDigitizerArm Attributes

The IviDigitizerArm extension group defines the following attributes:

- Active Arm Source
- Arm Count
- Arm Coupling
- Arm Delay
- Arm Hysteresis
- Arm Level
- Arm Output Enabled
- Arm Slope
- Arm Source Count
- Arm Source Item (IVI-COM & IVI.NET only)
- Arm Source Name (IVI-COM & IVI.NET only)
- Arm Type

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

16.2.1 Active Arm Source

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	R/W	N/A	None	N/A

.NET Property Name

`Arm.ActiveSource`

COM Property Name

`Arm.ActiveSource`

C Constant Name

`IVIDIGITIZER_ATTR_ACTIVE_ARM_SOURCE`

Description

Specifies the source the digitizer monitors for the arm event. The value specified here must be one of the valid repeated capability names for the ArmSource repeated capability.

If an IVI driver supports an arm source and the arm source is listed in IVI-3.3 *Cross Class Capabilities Specification*, Section 3, then the IVI driver shall accept the standard string for that arm source. This attribute is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new arm source strings for arm sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

This attribute only affects instrument behavior when either the IviDigitizerMultiArm extension group is not supported or the Arm Source Operator is set to None.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.2 Arm Count

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	N/A

.NET Property Name

`Arm.Count`

COM Property Name

`Arm.Count`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_COUNT`

Description

Specifies the number of times the arm has to occur to complete the arm loop; that is, the number of arms that are accepted before the acquisition must be initiated again.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.3 Arm Coupling

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	N/A

.NET Property Name

`Arm.Sources[].Coupling`

.NET Enumeration Name

`TriggerCoupling`

COM Property Name

`Arm.Sources.Item().Coupling`

COM Enumeration Name

`IviDigitizerTriggerCouplingEnum`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_COUPLING`

Description

Specifies how the digitizer couples the arm source.

Defined Values

Name	*Description*		
		Language	*Identifier*
AC	The digitizer AC couples the arm signal.		
		.NET	`TriggerCoupling.AC`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_AC`
		COM	`IviDigitizerTriggerCouplingAC`
DC	The digitizer DC couples the arm signal.		
		.NET	`TriggerCoupling.DC`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_DC`
		COM	`IviDigitizerTriggerCouplingDC`
HF Reject	The digitizer filters out the high frequencies from the arm signal.		
		.NET	`TriggerCoupling.HFReject`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING HF_REJECT`
		COM	`IviDigitizerTriggerCouplingHFReject`
LF Reject	The digitizer filters out the low frequencies from the arm signal.		
		.NET	`TriggerCoupling.LFReject`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING LF_REJECT`

		COM	`IviDigitizerTriggerCouplingLFReject`
Noise Reject	The digitizer filters out the noise from the arm signal.		
		.NET	`TriggerCoupling.NoiseReject`
		C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING NOISE_REJECT`
		COM	`IviDigitizerTriggerCouplingNoiseReject`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.4 Arm Delay

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	N/A	Down	N/A

.NET Property Name

`Arm.Delay`

COM Property Name

`Arm.Delay`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_DELAY`

Description

Specifies the delay from when the arm logic satisfied until the waiting for trigger state is entered. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.5 Arm Hysteresis

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	ArmSource	None	N/A

.NET Property Name

`Arm.Sources[].Hysteresis`

COM Property Name

`Arm.Sources.Item().Hysteresis`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_HYSTERESIS`

Description

Specifies the arm hysteresis in Volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.6 Arm Level

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	ArmSource	None	Configure Edge Arm Source Configure Glitch Arm Source Configure Width Arm Source

.NET Property Name

`Arm.Sources[].Level`

COM Property Name

`Arm.Sources.Item().Level`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_LEVEL`

Description

Specifies the voltage threshold for the arm sub-system. The units are Volts. This attribute affects instrument behavior only when the Arm Type is set to one of the following values: Arm Edge, Arm Glitch, or Arm Width. This attribute, along with the Arm Slope, Arm Source, and Arm Coupling attributes, defines the arm event when the Arm Type is set to one of the following values: Arm Edge, Arm Glitch, or Arm Width.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.7 Arm Output Enabled

Data Type	Access	Applies To	Coercion	High Level Functions
`ViBoolean`	R/W	N/A	None	N/A

.NET Property Name

`Arm.OutputEnabled`

COM Property Name

`Arm.OutputEnabled`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_OUTPUT_ENABLED`

Description

Specifies whether or not an accepted arm appears at an output of the digitizer.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.8 Arm Slope

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure Edge Arm Source

.NET Property Name

`Arm.Sources[].Edge.Slope`

.NET Enumeration Name

`Slope`

COM Property Name

`Arm.Sources.Item().Edge.Slope`

COM Enumeration Name

`IviDigitizerTriggerSlopeEnum`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_SLOPE`

Description

Specifies whether a rising or a falling edge arms the digitizer. This attribute affects instrument operation only when the Arm Type attribute is set to Arm Edge.

Defined Values

Name	***Description***		
		Language	***Identifier***
Negative	A negative (falling) edge passing through the arm level arms the digitizer.		
		.NET	`Slope.Negative`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`
		COM	`IviDigitizerTriggerSlopeNegative`
Positive	A positive (rising) edge passing through the arm level arms the digitizer.		
		.NET	`Slope.Positive`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`
		COM	`IviDigitizerTriggerSlopePositive`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.9 Arm Source Count

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	RO	N/A	None	N/A

.NET Property Name

`Arm.Sources.Count`

This property is inherited from `IIviRepeatedCapabilityCollection`.

COM Property Name

`Arm.Sources.Count`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_SOURCE_COUNT`

Description

Returns the number of arm sources available on the device.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.10 Arm Source Item (IVI-COM & IVI.NET only)

Data Type	Access	Applies To	Coercion	High Level Functions
`IIviDigitizerArmSource`	RO	ArmSource	None	N/A

.NET Property Name

```
IIviDigitizerSource Arm.Sources[String name];
```

This indexer is inherited from `IIviRepeatedCapabilityCollection`. The name parameter uniquely identifies a particular arm source in the arm sources collection.

COM Property Name

```
Arm.Sources.Item([in] BSTR Name)
```

C Constant Name

```
N/A
```

Description

`Arm` Source Item uniquely identifies an arm source in the arm sources collection. It returns an interface pointer which can be used to control the attributes and other functionality of that arm source.

The Item property takes an arm source name. If the user passes an invalid value for the `Name` parameter, the property returns an error.

Valid names include physical repeated capability identifiers and virtual repeated capability identifiers.

If an IVI driver supports an arm source and the arm source is listed in IVI-3.3 *Cross Class Capabilities Specification*, Section 3, then the IVI driver shall accept the standard string for that arm source. This attribute is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new arm source strings for arm sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.11 Arm Source Name (IVI-COM & IVI.NET only)

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	RO	ArmSource	None	N/A

.NET Property Name

`Arm.Sources[].Name`

This property is inherited from `IIviRepeatedCapabilityIdentification`.

COM Property Name

`Arm.Sources.Name([in] LONG Index)`

C Constant Name

`N/A`

(Use the `GetArmSourceName` function.)

Description

This property returns the physical arm source identifier that corresponds to the index that the user specifies. If the driver defines a qualified arm source name, this property returns the qualified name.

In COM, the index is one-based. In .NET, the index is zero-based.

For COM, if the value that the user passes for the Index parameter is less than one or greater than the value of the Arm Source Count attribute, the property returns an empty string in the Name parameter and returns the Invalid Value error.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.2.12 Arm Type

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	N/A

.NET Property Name

`Arm.Sources[].Type`

.NET Enumeration Name

`ArmType`

COM Property Name

`Arm.Sources.Item().Type`

COM Enumeration Name

`IviDigitizerArmTypeEnum`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_TYPE`

Description

The kind of event that arms the digitizer.

Defined Values

Name	***Description***		
		Language	***Identifier***
Arm Edge	Configures the digitizer for edge arming. An edge arm occurs when the arm signal specified with the Arm Source attribute passes the voltage threshold specified with the Arm Level attribute and has the slope specified with the Arm Slope attribute.		
		.NET	`ArmType.Edge`
		C	`IVIDIGITIZER_VAL_EDGE_ARM`
		COM	`IviDigitizerArmEdge`
Arm Width	Configures the digitizer for width arming. Use the IviDigitizerWidthArm extension properties and methods to configure the arm.		
		.NET	`ArmType.Width`
		C	`IVIDIGITIZER_VAL_WIDTH_ARM`
		COM	`IviDigitizerArmWidth`
Arm Runt	Configures the digitizer for runt arming. Use the IviDigitizerRuntArm extension properties and methods to configure the arm.		
		.NET	`ArmType.Runt`
		C	`IVIDIGITIZER_VAL_RUNT_ARM`
		COM	`IviDigitizerArmRunt`

Arm Glitch	Configures the digitizer for glitch arming. Use the IviDigitizerGlitchArm extension properties and methods to configure the arm.		
		.NET	`ArmType.Glitch`
		C	`IVIDIGITIZER_VAL_GLITCH_ARM`
		COM	`IviDigitizerArmGlitch`
Arm TV	Configures the digitizer for arming on TV signals. Use the IviDigitizerTVArm extension properties and methods to configure the arm.		
		.NET	`ArmType.TV`
		C	`IVIDIGITIZER_VAL_TV_ARM`
		COM	`IviDigitizerArmTV`
Arm Window	Configures the digitizer for window arming. Use the IviDigitizerWindowArm extension properties and methods to configure the arm.		
		.NET	`ArmType.Window`
		C	`IVIDIGITIZER_VAL_WINDOW_ARM`
		COM	`IviDigitizerArmWindow`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

16.3 IviDigitizerArm Functions

The IviDigitizerArm extension group defines the following functions:

- Configure Edge Arm Source
- Get Arm Source Name (IVI-C only)

This section describes the behavior and requirements of these functions.

16.3.1 Configure Edge Arm Source

Description

This function sets the edge arming attributes. An edge occurs when the arm signal that the end-user specifies with the Source parameter passes through the voltage threshold that the end-user specifies with the Level parameter and has the slope that the end-user specifies with the Slope parameter. This function affects instrument behavior only if the Arm Type is Arm Edge. Set the Arm Type and Arm Coupling before calling this function. If the arm source is one of the analog input channels, an application program should configure the vertical range, vertical coupling, and the maximum input frequency before calling this function.

.NET Method Prototype

```
void Arm.Sources[].Edge.Configure (Double level,
                                   Slope slope);
```

COM Method Prototype

```
HRESULT Arm.Sources.Item().Edge.Configure ([in] double Level,
                                           [in] IviDigitizerTriggerSlopeEnum Slope);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureEdgeArmSource (ViSession Vi,
                                              ViConstString Source,
                                              ViReal64 Level,
                                              ViInt32 Slope);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the arm source that is to be configured.	`ViString`
`Level`	Specifies the arm level. This value sets the Arm Level attribute.	`ViReal64`
`Slope`	Specifies the arm slope. This value sets the Arm Slope attribute.	`ViInt32`

Defined Values for the Slope Parameter

Name	***Description***		
		Language	***Identifier***
Negative	A negative (falling) edge passing through the arm level arms the digitizer.		
		.NET	`Slope.Negative`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`
		COM	`IviDigitizerTriggerSlopeNegative`
Positive	A positive (rising) edge passing through the arm level arms the digitizer.		
		.NET	`Slope.Positive`
		C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`

		COM	`IviDigitizerTriggerSlopePositive`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

16.3.2 Get Arm Source Name (IVI-C Only)

Description

This function returns the specific driver defined arm source name that corresponds to the one-based index that the user specifies. If the driver defines a qualified arm source name, this function returns the qualified name. If the value that the user passes for the `Index` parameter is less than one or greater than the value of the Arm Source Count attribute, the function returns an empty string in the `Name` parameter and returns the Invalid Value error.

.NET Method Prototype

N/A

(use the `Arm.Sources[].Name` property)

COM Method Prototype

N/A

(use the `Arm.Sources.Item().Name` property)

C Prototype

```
ViStatus IviDigitizer_GetArmSourceName (ViSession Vi,
                                        ViInt32 SourceIndex,
                                        ViInt32 SourceNameBufferSize,
                                        ViChar SourceName[]);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`SourceIndex`	A one-based index that defines which name to return	`ViInt32`
`SourceNameBufferSize`	The number of bytes in the ViChar array that the user specifies for the Source`Name` parameter.	`ViInt32`

Outputs	Description	Base Type
`SourceName`	A user-allocated (for IVI-C) or driver-allocated (for IVI-COM) buffer into which the driver stores the arm source name. The caller may pass `VI_NULL` for this parameter if the Source`NameBufferSize` parameter is 0.	`ViChar[]`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

16.4 IviDigitizerArm Behavior Model

The IviDigitizerArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.1.3, *IviDigitizerBase Arm and Trigger States*.
.
The main feature that the IviDigitizerArm capability group adds to the IviDigitizerBase capability group is that the device does not necessarily return to the *Idle* state after the desired number of records have been acquired. Rather, an arm count specifies the number of times the arm has to occur to complete the arm loop; that is, the number of arms that are accepted before the acquisition must be initiated again.

16.5 IviDigitizerArm Compliance Notes

For a specific driver to comply with the IviDigitizerArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

17. IviDigitizerMultiArm Extension Group

17.1 IviDigitizerMultiArm Overview

The IviDigitizerMultiArm extension group supports digitizers with the ability to arm on combinations of arm sources and their associated arm conditions. The user has the ability to logically OR arm sources together or to logically AND them together and specify the result as the overall arm source.

17.2 IviDigitizerMultiArm Attributes

The IviDigitizerMultiArm extension group defines the following attributes:

- Arm Source List
- Arm Source Operator

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

17.2.1 Arm Source List

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	R/W	N/A	None	Configure Multi Arm

.NET Property Name

`Arm.MultiArm.SourceList`

COM Property Name

`Arm.MultiArm.SourceList`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_SOURCE_LIST`

Description

A comma separated list of source names to be used in a multi arm scenario. This attribute only affects instrument behavior when the Arm Source Operator attribute is set to AND or OR. When Arm Source Operator is set to AND, the arm conditions associated with each source in this list must be simultaneously satisfied in order to arm the digitizer. When Arm Source Operator is set to OR, the first arm source in the list that satisfies its arm conditions will arm the digitizer. Any valid name used for the Arm Source attribute may be used in this list. An arm source may appear only once in the list. If a name in the list is not recognized, the driver returns the Unknown Channel Name error. See IVI-3.2 for the definition of this error.

If an IVI driver supports an arm source and the arm source is listed in IVI-3.3 *Cross Class Capabilities Specification*, Section 3, then the IVI driver shall accept the standard string for that arm source. This attribute is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new arm source strings for arm sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

17.2.2 Arm Source Operator

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	Configure Multi Arm

.NET Property Name

`Arm.MultiArm.SourceOperator`

.NET Enumeration Name

`ArmSourceOperator`

COM Property Name

`Arm.MultiArm.SourceOperator`

COM Enumeration Name

`IviDigitizerArmSourceOperatorEnum`

C Constant Name

`IVIDIGITIZER_ATTR_ARM_SOURCE_OPERATOR`

Description

Specifies the boolean operation to apply to the arm sources specified by the Arm Source List attribute. See the definition of that attribute for details.

Defined Values

Name	***Description***		
		Language	***Identifier***
AND	Arm sources are AND'd together. The digitizer arms when all configured arm source conditions are satisfied.		
		.NET	`ArmSourceOperator.And`
		C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_AND`
		COM	`IviDigitizerArmSourceOperatorAND`
OR	Arm sources are OR'd together. The digitizer arms when the first configured arm source condition is satisfied.		
		.NET	`ArmSourceOperator.Or`
		C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_OR`
		COM	`IviDigitizerArmSourceOperatorOR`
None	No operator is applied to the configured list of arm sources. The arm source list is ignored, and the digitizer arms when the active arm source, given by the Active Arm Source attribute, and its associated arm conditions are satisfied.		
		.NET	`ArmSourceOperator.None`
		C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_NONE`

		COM	`IviDigitizerArmSourceOperatorNone`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

17.3 IviDigitizerMultiArm Functions

The IviDigitizerMultiArm extension group defines the following function:

- Configure Multi Arm

This section describes the behavior and requirements of this function.

17.3.1 Configure Multi Arm

Description

Configures the digitizer to arm based on multiple arm sources. The digitizer can be instructed to arm when any one of multiple arm source conditions are met or when all specified arm source conditions are met.

.NET Method Prototype

```
void Arm.MultiArm.Configure (String sourceList,
                             ArmSourceOperator operator);
```

COM Method Prototype

```
HRESULT Arm.MultiArm.Configure ([in] BSTR SourceList,
                            [in] IviDigitizerArmSourceOperatorEnum Operator);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureMultiArm (ViSession Vi,
                                         ViConstString SourceList,
                                         ViInt32 Operator);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`SourceList`	A comma separated list of source names to be used in a multi arm scenario. The driver uses this value to set the Arm Source List attribute. See the attribute description for more details.	`ViConstString`
`Operator`	Specifies the boolean operation to apply to the arm sources specified by the SourceList parameter. The driver uses this value to set the Arm Source Operator attribute. See the attribute description for more details.	`ViInt32`

Defined Values for the Operator Parameter

Name	*Description*		
		Language	*Identifier*
AND	Arm sources are AND'd together. The digitizer arms when all configured arm source conditions are satisfied.		
		.NET	`ArmSourceOperator.And`
		C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_AND`
		COM	`IviDigitizerArmSourceOperatorAND`
OR	Arm sources are OR'd together. The digitizer arms when the first configured arm source condition is satisfied.		
		.NET	`ArmSourceOperator.Or`
		C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_OR`
		COM	`IviDigitizerArmSourceOperatorOR`

None	No operator is applied to the configured list of arm sources. The arm source list is ignored, and the digitizer arms when the active arm source, given by the Active Arm Source attribute, and its associated arm conditions are satisfied.		
		.NET	`ArmSourceOperator.None`
		C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_NONE`
		COM	`IviDigitizerArmSourceOperatorNone`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

17.4 IviDigitizerMultiArm Behavior Model

The IviDigitizerMultiArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

17.5 IviDigitizerMultiArm Compliance Notes

For a specific driver to comply with the IviDigitizerMultiArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

18. IviDigitizerGlitchArm Extension Group

18.1 IviDigitizerGlitchArm Overview

In addition to the fundamental capabilities, the IviDigitizerGlitchArm extension group defines extensions for digitizers that can arm on a “glitch” pulses. The device arms in response to glitches in the same way that it triggers in response to glitches. See Section 28 for details on glitches.

18.2 IviDigitizerGlitchArm Attributes

The IviDigitizerGlitchArm extension group defines the following attributes:

- Glitch Arm Condition
- Glitch Arm Polarity
- Glitch Arm Width

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

18.2.1 Glitch Arm Condition

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure Glitch Arm Source

.NET Property Name

`Arm.Sources[].Glitch.Condition`

.NET Enumeration Name

`GlitchCondition`

COM Property Name

`Arm.Sources.Item().Glitch.Condition`

COM Enumeration Name

`IviDigitizerGlitchConditionEnum`

C Constant Name

`IVIDIGITIZER_ATTR_GLITCH_ARM_CONDITION`

Description

Specifies the glitch condition. This attribute determines whether the glitch arm happens when the digitizer detects a pulse with a width less than or greater than the width value.

Defined Values

Name	***Description***		
		Language	***Identifier***
Less Than	The digitizer arms when the pulse width is less than the value you specify with the Glitch Arm Width attribute.		
		.NET	`GlitchCondition.LessThan`
		C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`
		COM	`IviDigitizerGlitchConditionLessThan`
Greater Than	The digitizer arms when the pulse width is greater than the value you specify with the Glitch Arm Width attribute.		
		.NET	`GlitchCondition.GreaterThan`
		C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`
		COM	`IviDigitizerGlitchConditionGreaterThan`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

18.2.2 Glitch Arm Polarity

Data Type	Access	Applies to	Coercion	High Level Functions
`ViInt32`	`R/W`	ArmSource	None	Configure Glitch Arm Source

COM Property Name

`Arm.Sources[].Glitch.Polarity`

COM Enumeration Name

`GlitchPolarity`

COM Property Name

`Arm.Sources.Item().Glitch.Polarity`

COM Enumeration Name

`IviDigitizerGlitchPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_GLITCH_ARM_POLARITY`

Description

Specifies the polarity of the glitch that arms the digitizer.

Defined Values

Name	***Description***		
		Language	***Identifier***
Glitch Positive	The digitizer arms on a positive glitch.		
		.NET	`GlitchPolarity.Positive`
		C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`
		COM	`IviDigitizerGlitchPolarityPositive`
Glitch Negative	The digitizer arms on a negative glitch.		
		.NET	`GlitchPolarity.Negative`
		C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`
		COM	`IviDigitizerGlitchPolarityNegative`
Glitch Either	The digitizer arms on either a positive or negative glitch.		
		.NET	`GlitchPolarity.Either`
		C	`IVIDIGITIZER_VAL_GLITCH_EITHER`
		COM	`IviDigitizerGlitchPolarityEither`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

18.2.3 Glitch Arm Width

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	ArmSource	None	Configure Glitch Arm Source

.NET Property Name

`Arm.Sources[].Glitch.Width`

COM Property Name

`Arm.Sources.Item().Glitch.Width`

C Constant Name

`IVIDIGITIZER_ATTR_GLITCH_ARM_WIDTH`

Description

Specifies the glitch width. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan. The digitizer arms when it detects a pulse with a width less than or greater than this value, depending on the Glitch Arm Condition attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

18.3 IviDigitizerGlitchArm Functions

The IviDigitizerGlitchArm extension group defines the following function:

- Configure Glitch Arm Source

This section describes the behavior and requirements of this function.

18.3.1 Configure Glitch Arm Source

Description

This function configures the glitch arm. A glitch arm occurs when the arm signal has a pulse with a width that is less than or greater than the glitch width. The end user specifies which comparison criterion to use with the GlitchCondition parameter. The end-user specifies the glitch width with the GlitchWidth parameter. The end-user specifies the polarity of the pulse with the GlitchPolarity parameter. The arm does not actually occur until the edge of a pulse that corresponds to the GlitchWidth and GlitchPolarity crosses the threshold the end-user specifies in the TriggerLevel parameter. This function affects instrument behavior only if the arm type is Glitch Arm. Set the arm type and arm coupling before calling this function.

.NET Method Prototype

```
void Arm.Sources[].Glitch.Configure (Double level,
                                     PrecisionTimeSpan width,
                                     GlitchPolarity polarity,
                                     GlitchCondition condition);
```

COM Method Prototype

```
HRESULT Arm.Sources.Item().Glitch.Configure ([in] double Level,
                               [in] double Width,
                               [in] IviDigitizerGlitchPolarityEnum Polarity,
                               [in] IviDigitizerGlitchConditionEnum Condition);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureGlitchArmSource (ViSession Vi,
                                                ViConstString Source,
                                                ViReal64 Level,
                                                ViReal64 Width,
                                                ViInt32 Polarity,
                                                ViInt32 Condition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the arm source that is to be configured.	`ViString`
`Level`	Specifies the arm level. This value sets the Arm Level attribute.	`ViReal64`
`Width`	Specifies the glitch arming glitch width. This value sets the Glitch Arm Width attribute.	`ViReal64(C/COM)` `PrecisionTimeSpan (.NET)`
`Polarity`	Specifies the glitch polarity. This value sets the Glitch Arm Polarity attribute.	`ViInt32`
`Condition`	Specifies the glitch condition. This value sets the Glitch Arm Condition attribute.	`ViInt32`

Defined Values for the Polarity Parameter

Name	*Description*		
		Language	*Identifier*
Positive	The digitizer arms on a positive glitch.		
		.NET	`GlitchPolarity.Positive`
		C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`
		COM	`IviDigitizerGlitchPolarityPositive`
Negative	The digitizer arms on a negative glitch.		
		.NET	`GlitchPolarity.Negative`
		C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`
		COM	`IviDigitizerGlitchPolarityNegative`
Either	The digitizer arms on either a positive or negative glitch.		
		.NET	`GlitchPolarity.Either`
		C	`IVIDIGITIZER_VAL_GLITCH_EITHER`
		COM	`IviDigitizerGlitchPolarityEither`

Defined Values for the Condition Parameter

Name	*Description*		
		Language	*Identifier*
Less Than	The digitizer arms when the pulse width is less than the value you specify with the Glitch Arm Width attribute.		
		.NET	`GlitchCondition.LessThan`
		C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`
		COM	`IviDigitizerGlitchConditionLessThan`
Greater Than	The digitizer arms when the pulse width is greater than the value you specify with the Glitch Arm Width attribute.		
		.NET	`GlitchCondition.GreaterThan`
		C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`
		COM	`IviDigitizerGlitchConditionGreaterThan`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

18.4 IviDigitizerGlitchArm Behavior Model

The IviDigitizerGlitchArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

18.5 IviDigitizerGlitchArm Compliance Notes

For a specific driver to comply with the IviDigitizerGlitchArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

19. IviDigitizerRuntArm Extension Group

19.1 IviDigitizerRuntArm Overview

In addition to the fundamental capabilities, the IviDigitizerRuntArm extension group defines extensions for digitizers that can arm on “runt” pulses. The device arms in response to runts in the same way that it triggers in response to runts. See Section 29 for details on runts.

19.2 IviDigitizerRuntArm Attributes

The IviDigitizerRuntArm extension group defines the following attributes:

- Runt Arm High Threshold
- Runt Arm Low Threshold
- Runt Arm Polarity

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

19.2.1 Runt Arm High Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	ArmSource	None	Configure Runt Arm Source

.NET Property Name

`Arm.Sources[].Runt.ThresholdHigh`

COM Property Name

`Arm.Sources.Item().Runt.ThresholdHigh`

C Constant Name

`IVIDIGITIZER_ATTR_RUNT_ARM_HIGH_THRESHOLD`

Description

The high threshold the digitizer uses for runt arming. The units are volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

19.2.2 Runt Arm Low Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	ArmSource	None	Configure Runt Arm Source

.NET Property Name

`Arm.Sources[].Runt.ThresholdLow`

COM Property Name

`Arm.Sources.Item().Runt.ThresholdLow`

C Constant Name

`IVIDIGITIZER_ATTR_RUNT_ARM_LOW_THRESHOLD`

Description

The low threshold the digitizer uses for runt arming. The units are volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

19.2.3 Runt Arm Polarity

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure Runt Arm Source

.NET Property Name

`Arm.Sources[].Runt.Polarity`

.NET Enumeration Name

`RuntPolarity`

COM Property Name

`Arm.Sources.Item().Runt.Polarity`

COM Enumeration Name

`IviDigitizerRuntPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_RUNT_ARM_POLARITY`

Description

Specifies the polarity of the runt that arms the digitizer.

Defined Values

Name	***Description***		
		Language	***Identifier***
Positive	The digitizer arms on a positive runt. A positive runt occurs when a rising edge crosses the low runt threshold and does not cross the high runt threshold before re-crossing the low runt threshold.		
		.NET	`RuntPolarity.Positive`
		C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`
		COM	`IviDigitizerRuntPolarityPositive`
Negative	The digitizer arms on a negative runt. A negative runt occurs when a falling edge crosses the high runt threshold and does not cross the low runt threshold before re-crossing the high runt threshold.		
		.NET	`RuntPolarity.Negative`
		C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`
		COM	`IviDigitizerRuntPolarityNegative`
Either	The digitizer arms on either a positive or negative runt.		
		.NET	`RuntPolarity.Either`
		C	`IVIDIGITIZER_VAL_RUNT_EITHER`
		COM	`IviDigitizerRuntPolarityEither`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

19.3 IviDigitizerRuntArm Functions

The IviDigitizerRuntArm extension group defines the following function:

- Configure Runt Arm Source

This section describes the behavior and requirements of this function.

19.3.1 Configure Runt Arm Source

Description

This function configures the runt arm. A runt occurs when the arm signal crosses one of the runt thresholds twice without crossing the other runt threshold. The end-user specifies the runt thresholds with the RuntLowThreshold and RuntHighThreshold parameters. The end-user specifies the polarity of the runt with the RuntPolarity parameter. This function affects instrument behavior only if the arm type is Runt Trigger. Set the arm type and trigger coupling before calling this function.

.NET Method Prototype

```
void Arm.Sources[].Runt.Configure (Double thresholdLow,
                                   Double thresholdHigh,
                                   RuntPolarity polarity);
```

COM Method Prototype

```
HRESULT Arm.Sources.Item().Runt.Configure ([in] double ThresholdLow,
                                   [in] double ThresholdHigh,
                                   [in] IviDigitizerRuntPolarityEnum Polarity);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureRuntArmSource (ViSession Vi,
                                              ViConstString Source,
                                              ViReal64 ThresholdLow,
                                              ViReal64 ThresholdHigh,
                                              ViInt32 Polarity);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the arm source that is to be configured.	`ViString`
`ThresholdLow`	Sets the runt triggering low threshold in volts. This value sets the Runt Arm Low Threshold attribute.	`ViReal64`
`ThresholdHigh`	Sets the runt triggering high threshold in volts. This value sets the Runt Arm High Threshold attribute.	`ViReal64`
`Polarity`	Sets the runt polarity. This value sets the Runt Arm Polarity attribute.	`ViInt32`

Defined Values for the Polarity Parameter

Name	*Description*		
		Language	*Identifier*
Positive	The digitizer arms on a positive runt. A positive runt occurs when a rising edge crosses the low runt threshold and does not cross the high runt threshold before re-crossing the low runt threshold.		
		.NET	`RuntPolarity.Positive`
		C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`
		COM	`IviDigitizerRuntPolarityPositive`
Negative	The digitizer arms on a negative runt. A negative runt occurs when a falling edge crosses the high runt threshold and does not cross the low runt threshold before re-crossing the high runt threshold.		
		.NET	`RuntPolarity.Negative`
		C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`
		COM	`IviDigitizerRuntPolarityNegative`
Either	The digitizer arms on either a positive or negative runt.		
		.NET	`RuntPolarity.Either`
		C	`IVIDIGITIZER_VAL_RUNT_EITHER`
		COM	`IviDigitizerRuntPolarityEither`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

19.4 IviDigitizerRuntArm Behavior Model

The IviDigitizerRuntArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

19.5 IviDigitizerRuntArm Compliance Notes

For a specific driver to comply with the IviDigitizerRuntArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

20. IviDigitizerSoftwareArm Extension Group

20.1 IviDigitizerSoftwareArm Overview

The IviDigitizerSoftwareArm extension group supports digitizers with the ability to arm an acquisition.

20.2 IviDigitizerSoftwareArm Functions

The IviDigitizerSoftwareArm extension group defines the following function:

- Send Software Arm

This section describes the behavior and requirements of this function.

20.2.1 Send Software Arm

Description

This function sends a software-generated arm to the instrument. It is only applicable for instruments using interfaces or protocols which support an explicit arm function.

Since instruments interpret a software-generated arm in a wide variety of ways, the precise response of the instrument to this arm is not defined.

This function should not use resources which are potentially shared by other devices (for example, the VXI trigger lines). Use of such shared resources may have undesirable effects on other devices.

This function should not check the instrument status. Typically, the end-user calls this function only in a sequence of calls to other low-level driver functions. The sequence performs one operation. The end-user uses the low-level functions to optimize one or more aspects of interaction with the instrument. To check the instrument status, call the appropriate error query function at the conclusion of the sequence.

The arm source attribute must accept Software Arm as a valid setting for this function to work. If the arm source is not set to Software Arm, this function does nothing and returns the error Arm Not Software.

.NET Method Prototype

```
void Arm.SendSoftwareArm ();
```

COM Method Prototype

```
HRESULT Arm.SendSoftwareArm ();
```

C Prototype

```
ViStatus IviDigitizer_SendSoftwareArm (ViSession Vi);
```

Parameters

Inputs	**Description**	**Base Type**
`Vi`	Instrument handle.	`ViSession`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional class-defined status code:

- Arm Not Software

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method. This method can also return this additional class-defined exception:

- Arm Not Software

20.3 IviDigitizerSoftwareArm Behavior Model

The IviDigitizerSoftwareArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

20.4 IviDigitizerSoftwareArm Compliance Notes

For a specific driver to comply with the IviDigitizerSoftwareArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

21. IviDigitizerTVArm Extension Group

21.1 IviDigitizerTVArm Overview

In addition to the fundamental capabilities, the IviDigitizerTVArm extension group defines extensions for digitizers that can arm on standard TV signals. The device arms in response to TV signals in the same way that it triggers in response to TV signals. See Section 31 for details on TV triggers.

21.2 IviDigitizerTVArm Attributes

The IviDigitizerTVArm extension group defines the following attributes:

- TV Arm Event
- TV Arm Line Number
- TV Arm Polarity
- TV Arm Signal Format

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

21.2.1 TV Arm Event

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure TV Arm Source

.NET Property Name

`Arm.Sources[].TV.TriggerEvent`

.NET Enumeration Name

`TVTriggerEvent`

COM Property Name

`Arm.Sources.Item().TV.Event`

COM Enumeration Name

`IviDigitizerTVTriggerEventEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TV_ARM_EVENT`

Description

Specifies the event on which the digitizer arms.

Defined Values

Name	*Description*		
		Language	*Identifier*
Field1	Sets the digitizer to arm on field 1 of the video signal.		
		.NET	`TVTriggerEvent.Field1`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`
		COM	`IviDigitizerTVTriggerEventField1`
Field2	Sets the digitizer to arm on field 2 of the video signal.		
		.NET	`TVTriggerEvent.Field2`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`
		COM	`IviDigitizerTVTriggerEventField2`
Any Field	Sets the digitizer to arm on any field.		
		.NET	`TVTriggerEvent.AnyField`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`
		COM	`IviDigitizerTVTriggerEventAnyField`
Any Line	Sets the digitizer to arm on any line.		
		.NET	`TVTriggerEvent.AnyLine`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`

		COM	`IviDigitizerTVTriggerEventAnyLine`
Line Number	Sets the digitizer to arm on a specific line number you specify with the TV Trigger Line Number attribute.		
		.NET	`TVTriggerEvent.LineNumber`
		C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`
		COM	`IviDigitizerTVTriggerEventLineNumber`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

21.2.2 TV Arm Line Number

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	N/A

COM Property Name

`Arm.Sources[].TV.LineNumber`

COM Property Name

`Arm.Sources.Item().TV.LineNumber`

C Constant Name

`IVIDIGITIZER_ATTR_TV_ARM_LINE_NUMBER`

Description

Specifies the line on which the digitizer arms. The driver uses this attribute when the TV Arm Event is set to TV Event Line Number. The line number setting is independent of the field. This means that to arm on the first line of the second field, the user must configure the line number to the value of 263 (if we presume that field one had 262 lines).

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

21.2.3 TV Arm Polarity

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure TV Arm Source

.NET Property Name

`Arm.Sources[].TV.Polarity`

.NET Enumeration Name

`TVTriggerPolarity`

COM Property Name

`Arm.Sources.Item().TV.Polarity`

COM Enumeration Name

`IviDigitizerTVTriggerPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TV_ARM_POLARITY`

Description

Specifies the polarity of the TV signal.

Defined Values

Name	*Description*		
		Language	*Identifier*
Positive	Configures the digitizer to arm on a positive video sync pulse.		
		.NET	`TVTriggerPolarity.Positive`
		C	`IVIDIGITIZER_VAL_TV_POSITIVE`
		COM	`IviDigitizerTVTriggerPolarityPositive`
Negative	Configures the digitizer to arm on a negative video sync pulse.		
		.NET	`TVTriggerPolarity.Negative`
		C	`IVIDIGITIZER_VAL_TV_NEGATIVE`
		COM	`IviDigitizerTVTriggerPolarityNegative`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

21.2.4 TV Arm Signal Format

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure TV Arm Source

.NET Property Name

`Arm.Sources[].TV.SignalFormat`

.NET Enumeration Name

`TVSignalFormat`

COM Property Name

`Arm.Sources.Item().TV.SignalFormat`

COM Enumeration Name

`IviDigitizerTVSignalFormatEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TV_ARM_SIGNAL_FORMAT`

Description

Specifies the format of TV signal on which the digitizer arms.

Defined Values

Name	*Description*		
		Language	*Identifier*
NTSC	Configures the digitizer to arm on the NTSC signal format.		
		.NET	`TVSignalFormat.Ntsc`
		C	`IVIDIGITIZER_VAL_NTSC`
		COM	`IviDigitizerTVSignalFormatNTSC`
PAL	Configures the digitizer to arm on the PAL signal format		
		.NET	`TVSignalFormat.Pal`
		C	`IVIDIGITIZER_VAL_PAL`
		COM	`IviDigitizerTVSignalFormatPAL`
SECAM	Configures the digitizer to arm on the SECAM signal format		
		.NET	`TVSignalFormat.Secam`
		C	`IVIDIGITIZER_VAL_SECAM`
		COM	`IviDigitizerTVSignalFormatSECAM`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

21.3 IviDigitizerTVArm Functions

The IviDigitizerTVArm extension group defines the following function:

- ConfigureTV Arm Source

This section describes the behavior and requirements of this function.

21.3.1 ConfigureTV Arm Source

Description

This function configures the digitizer for TV arming. It configures the TV signal format, the event and the signal polarity. This function affects instrument behavior only if the arm type is TV Arm. Set the Arm Type and Arm Coupling before calling this function.

.NET Method Prototype

```
void Arm.Sources[].TV.Configure (TVSignalFormat signalFormat,
                                 TVTriggerEvent event,
                                 TVTriggerPolarity polarity);
```

COM Method Prototype

```
HRESULT Arm.Sources.Item().TV.Configure (
                              [in] IviDigitizerTVSignalFormatEnum SignalFormat,
                              [in] IviDigitizerTVTriggerEventEnum Event,
                              [in] IviDigitizerTVTriggerPolarityEnum Polarity);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureTVArmSource (ViSession Vi,
                                            ViConstString Source,
                                            ViInt32 SignalFormat,
                                            ViInt32 Event,
                                            ViInt32 Polarity);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the arm source that is to be configured.	`ViString`
`SignalFormat`	Specifies the TV arm signal format. This value sets the TV Arm Signal Format attribute.	`ViInt32`
`Event`	Specifies the TV arm event. This value sets the TV Arm Event attribute.	`ViInt32`
`Polarity`	Specifies the polarity of the TV arm signal. This value sets the TV Arm Polarity attribute.	`ViInt32`

Defined Values for the SignalFormat Parameter

Name	***Description***		
		Language	***Identifier***
NTSC	Configures the digitizer to arm on the NTSC signal format.		
		.NET	`TVSignalFormat.Ntsc`
		C	`IVIDIGITIZER_VAL_NTSC`
		COM	`IviDigitizerTVSignalFormatNTSC`
PAL	Configures the digitizer to arm on the PAL signal format		
		.NET	`TVSignalFormat.Pal`
		C	`IVIDIGITIZER_VAL_PAL`

		COM	`IviDigitizerTVSignalFormatPAL`
SECAM	Configures the digitizer to arm on the SECAM signal format		
		.NET	`TVSignalFormat.Secam`
		C	`IVIDIGITIZER_VAL_SECAM`
		COM	`IviDigitizerTVSignalFormatSECAM`

Defined Values for the Event Parameter

Name	***Description***		
		Language	***Identifier***
Field1	Sets the digitizer to arm on field 1 of the video signal.		
		.NET	`TVTriggerEvent.Field1`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`
		COM	`IviDigitizerTVTriggerEventField1`
Field2	Sets the digitizer to arm on field 2 of the video signal.		
		.NET	`TVTriggerEvent.Field2`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`
		COM	`IviDigitizerTVTriggerEventField2`
Any Field	Sets the digitizer to arm on any field.		
		.NET	`TVTriggerEvent.AnyField`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`
		COM	`IviDigitizerTVTriggerEventAnyField`
Any Line	Sets the digitizer to arm on any line.		
		.NET	`TVTriggerEvent.AnyLine`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`
		COM	`IviDigitizerTVTriggerEventAnyLine`
Line Number	Sets the digitizer to arm on a specific line number you specify with the TV Trigger Line Number attribute.		
		.NET	`TVTriggerEvent.LineNumber`
		C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`
		COM	`IviDigitizerTVTriggerEventLineNumber`

Defined Values for the Polarity Parameter

Name	***Description***		
		Language	***Identifier***
Positive	Configures the digitizer to arm on a positive video sync pulse.		
		.NET	`TVTriggerPolarity.Positive`
		C	`IVIDIGITIZER_VAL_TV_POSITIVE`
		COM	`IviDigitizerTVTriggerPolarityPositive`
Negative	Configures the digitizer to arm on a negative video sync pulse.		

		.NET	`TVTriggerPolarity.Negative`
		C	`IVIDIGITIZER_VAL_TV_NEGATIVE`
		COM	`IviDigitizerTVTriggerPolarityNegative`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

21.4 IviDigitizerTVArm Behavior Model

The IviDigitizerTVArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

21.5 IviDigitizerTVArm Compliance Notes

For a specific driver to comply with the IviDigitizerTVArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

22. IviDigitizerWidthArm Extension Group

22.1 IviDigitizerWidthArm Overview

In addition to the fundamental capabilities, the IviDigitizerWidthArm extension group defines extensions for digitizers that can arm on user-specified pulse widths. The device arms in response to pulses in the same way that it triggers in response to pulses. See Section32 for details on pulse width triggering.

22.2 IviDigitizerWidthArm Attributes

The IviDigitizerWidthArm extension group defines the following attributes:

- Width Arm Condition
- Width Arm High Threshold
- Width Arm Lowthreshold
- Width Arm Polarity

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

22.2.1 Width Arm Condition

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure Width Arm Source

.NET Property Name

`Arm.Sources[].Width.Condition`

.NET Enumeration Name

`WidthCondition`

COM Property Name

`Arm.Sources.Item().Width.Condition`

COM Enumeration Name

`IviDigitizerWidthConditionEnum`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_ARM_CONDITION`

Description

Specifies whether a pulse that is within or outside the high and low thresholds arms the digitizer. The end-user specifies the high and low thresholds with the Width High Threshold and Width Low Threshold attributes.

Defined Values

Name	*Description*		
		Language	*Identifier*
Within	Configures the digitizer to arm on pulses that have a width that is less than the high threshold and greater than the low threshold. The end-user specifies the high and low thresholds with the Width Arm High Threshold and Width Arm Low Threshold properties.		
		.NET	`WidthCondition.Within`
		C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`
		COM	`IviDigitizerWidthConditionWithin`
Outside	Configures the digitizer to arm on pulses that have a width that is either greater than the high threshold or less than a low threshold. The end-user specifies the high and low thresholds with the Width Arm High Threshold and Width Arm Low Threshold properties.		
		.NET	`WidthCondition.Outside`
		C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`
		COM	`IviDigitizerWidthConditionOutside`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

22.2.2 Width Arm High Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	ArmSource	None	Configure Width Arm Source

.NET Property Name

`Arm.Sources[].Width.ThresholdHigh`

COM Property Name

`Arm.Sources.Item().Width.ThresholdHigh`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_ARM_HIGH_THRESHOLD`

Description

Specifies the high width threshold time. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

22.2.3 Width Arm Low Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	ArmSource	None	Configure Width Arm Source

.NET Property Name

`Arm.Sources[].Width.ThresholdLow`

COM Property Name

`Arm.Sources.Item().Width.ThresholdLow`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_ARM_LOW_THRESHOLD`

Description

Specifies the low width threshold time. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

22.2.4 Width Arm Polarity

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure Width Arm Source

.NET Property Name

`Arm.Sources[].Width.Polarity`

.NET Enumeration Name

`WidthPolarity`

COM Property Name

`Arm.Sources.Item().Width.Polarity`

COM Enumeration Name

`IviDigitizerWidthPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_ARM_POLARITY`

Description

Specifies the polarity of the pulse that arms the digitizer.

Defined Values

Name	*Description*		
		Language	*Identifier*
Positive	Configures the digitizer to arm on positive pulses that have a width that meets the condition the user specifies with the Width Arm Condition attribute.		
		.NET	`WidthPolarity.Positive`
		C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`
		COM	`IviDigitizerWidthPolarityPositive`
Negative	Configures the digitizer to arm on negative pulses that have a width that meets the condition the user specifies with the Width Arm Condition attribute.		
		.NET	`WidthPolarity.Negative`
		C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`
		COM	`IviDigitizerWidthPolarityNegative`
Either	Configures the digitizer to arm on either positive or negative pulses that have a width that meets the condition the user specifies with the Width Arm Condition attribute.		
		.NET	`WidthPolarity.Either`
		C	`IVIDIGITIZER_VAL_WIDTH_EITHER`
		COM	`IviDigitizerWidthPolarityEither`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

22.3 IviDigitizerWidthArm Functions

The IviDigitizerWidthArm extension group defines the following function:

- Configure Width Arm Source

This section describes the behavior and requirements of this function.

22.3.1 Configure Width Arm Source

Description

Configures the width arm Source, Level, ThresholdLow, ThresholdHigh, Polarity, and Condition. A width arm occurs when a pulse, that passes through Level, with a width between or outside, the width thresholds is detected.

.NET Method Prototype

```
void Arm.Sources[].Width.Configure (Double level,
                                    PrecisionTimeSpan thresholdLow,
                                    PrecisionTimeSpan thresholdHigh,
                                    WidthPolarity polarity,
                                    WidthCondition condition);
```

COM Method Prototype

```
HRESULT Arm.Sources.Item().Width.Configure ([in] double Level,
                                [in] double ThresholdLow,
                                [in] double ThresholdHigh,
                                [in] IviDigitizerWidthPolarityEnum Polarity,
                                [in] IviDigitizerWidthConditionEnum Condition);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureWidthArmSource (ViSession Vi,
                                               ViConstString Source,
                                               ViReal64 Level,
                                               ViReal64 ThresholdLow,
                                               ViReal64 ThresholdHigh,
                                               ViInt32 Polarity,
                                               ViInt32 Condition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the arm source that is to be configured.	`ViString`
`Level`	Arm Level. This value sets the Arm Level attribute.	`ViReal64`
`ThresholdLow`	Sets the width arming low threshold. This value sets the Width Arm Low Threshold attribute.	`ViReal64(C/COM)` `PrecisionTimeSpan (.NET)`
`ThresholdHigh`	Sets the width arming high threshold. This value sets the Width Arm High Threshold attribute.	`ViReal64(C/COM)` `PrecisionTimeSpan (.NET)`
`Polarity`	Sets the width polarity. This value sets the Width Arm Polarity attribute.	`ViInt32`
`Condition`	Specifies whether a pulse that is within or outside the user-specified thresholds arms the digitizer. This value sets the Width Arm Condition attribute.	`ViInt32`

Defined Values for the Polarity Parameter

Name	*Description*		
		Language	*Identifier*
Positive	Configures the digitizer to arm on positive pulses that have a width that meets the condition the user specifies with the Width Arm Condition attribute.		
		.NET	`WidthPolarity.Positive`
		C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`
		COM	`IviDigitizerWidthPolarityPositive`
Negative	Configures the digitizer to arm on negative pulses that have a width that meets the condition the user specifies with the Width Arm Condition attribute.		
		.NET	`WidthPolarity.Negative`
		C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`
		COM	`IviDigitizerWidthPolarityNegative`
Either	Configures the digitizer to arm on either positive or negative pulses that have a width that meets the condition the user specifies with the Width Arm Condition attribute.		
		.NET	`WidthPolarity.Either`
		C	`IVIDIGITIZER_VAL_WIDTH_EITHER`
		COM	`IviDigitizerWidthPolarityEither`

Defined Values for the Condition Parameter

Name	*Description*		
		Language	*Identifier*
Within	Configures the digitizer to arm on pulses that have a width that is less than the high threshold and greater than the low threshold. The end-user specifies the high and low thresholds with the Width Arm High Threshold and Width Arm Low Threshold properties.		
		.NET	`WidthCondition.Within`
		C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`
		COM	`IviDigitizerWidthConditionWithin`
Outside	Configures the digitizer to arm on pulses that have a width that is either greater than the high threshold or less than a low threshold. The end-user specifies the high and low thresholds with the Width Arm High Threshold and Width Arm Low Threshold properties.		
		.NET	`WidthCondition.Outside`
		C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`
		COM	`IviDigitizerWidthConditionOutside`

Return Values

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

22.4 IviDigitizerWidthArm Behavior Model

The IviDigitizerWidthArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

22.5 IviDigitizerWidthArm Compliance Notes

For a specific driver to comply with the IviDigitizerWidthArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

23. IviDigitizerWindowArm Extension Group

23.1 IviDigitizerWindowArm Overview

In addition to the fundamental capabilities, the IviDigitizerWindowArm extension group defines extensions for digitizers that can arm on user-specified voltage level windows. The device arms in response to a signals entering or leaving voltage ranges in the same way that it triggers in response to voltage range windows. See Section33for details on pulse width triggering.

23.2 IviDigitizerWindowArm Attributes

The IviDigitizerWindowArm extension group defines the following attributes:

- Window Arm Condition
- Window Arm High Threshold
- Window Arm Low Threshold

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

23.2.1 Window Arm Condition

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	ArmSource	None	Configure Window Arm Source

.NET Property Name

`Arm.Sources[].Window.Condition`

.NET Enumeration Name

`WindowCondition`

COM Property Name

`Arm.Sources.Item().Window.Condition`

COM Enumeration Name

`IviDigitizerWindowConditionEnum`

C Constant Name

`IVIDIGITIZER_ATTR_WINDOW_ARM_CONDITION`

Description

Specifies whether a signal that is entering or leaving the voltage window defined by the high and low thresholds arms the digitizer. The end-user specifies the high and low thresholds with the Window High Threshold and Window Low Threshold attributes.

Defined Values

Name	***Description***		
		Language	***Identifier***
Entering	Configures the digitizer to arm on signals when they enter the given arming window. The end-user specifies the high and low thresholds with the Window Arm High Threshold and Window Arm Low Threshold properties.		
		.NET	`WindowCondition.Entering`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`
		COM	`IviDigitizerWindowConditionEntering`
Leaving	Configures the digitizer to arm on signals when they leave the given arming window. The end-user specifies the high and low thresholds with the Window Arm High Threshold and Window Arm Low Threshold properties.		
		.NET	`WindowCondition.Leaving`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`
		COM	`IviDigitizerWindowConditionLeaving`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

23.2.2 Window Arm High Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	ArmSource	None	Configure Window Arm Source

.NET Property Name

`Arm.Sources[].Window.ThresholdHigh`

COM Property Name

`Arm.Sources.Item().Window.ThresholdHigh`

C Constant Name

`IVIDIGITIZER_ATTR_WINDOW_ARM_HIGH_THRESHOLD`

Description

Specifies the high window threshold voltage in Volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

23.2.3 Window Arm Low Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	ArmSource	None	Configure Window Arm Source

.NET Property Name

`Arm.Sources[].Window.ThresholdLow`

COM Property Name

`Arm.Sources.Item().Window.ThresholdLow`

C Constant Name

`IVIDIGITIZER_ATTR_WINDOW_ARM_LOW_THRESHOLD`

Description

Specifies the low window threshold voltage in Volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

23.3 IviDigitizerWindowArm Functions

The IviDigitizerWindowArm extension group defines the following function:

- Configure Window Arm Source

This section describes the behavior and requirements of this function.

23.3.1 Configure Window Arm Source

Description

Configures the window Arm Source, ThresholdLow, ThresholdHigh, and Condition. A window arm occurs when a signal that enters or leaves a given voltage range is detected.

.NET Method Prototype

```
void Arm.Sources[].Window.Configure (Double thresholdLow,
                                     Double thresholdHigh,
                                     WindowCondition condition);
```

COM Method Prototype

```
HRESULT Arm.Sources.Item().Window.Configure ([in] double ThresholdLow,
                               [in] double ThresholdHigh,
                               [in] IviDigitizerWindowConditionEnum Condition);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureWindowArmSource (ViSession Vi,
                                                ViConstString Source,
                                                ViReal64 ThresholdLow,
                                                ViReal64 ThresholdHigh,
                                                ViInt32 Condition);
```

Parameters

Inputs	**Description**	**Base Type**
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the arm source that is to be configured.	`ViString`
`ThresholdLow`	Sets the window arming low threshold in Volts. This value sets the Window Arm Low Threshold attribute.	`ViReal64`
`ThresholdHigh`	Sets the window arming high threshold in Volts. This value sets the Window Arm High Threshold attribute.	`ViReal64`
`Condition`	Specifies whether a signal that is entering or leaving the voltage window defined by the high and low thresholds arms the digitizer. This value sets the Window Arm Condition attribute.	`ViInt32`

Defined Values for the Condition Parameter

Name	***Description***		
		Language	***Identifier***
Entering	Configures the digitizer to arm on signals when they enter the given arming window. The end-user specifies the high and low thresholds with the Window Arm High Threshold and Window Arm Low Threshold properties.		
		.NET	`WindowCondition.Entering`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`
		COM	`IviDigitizerWindowConditionEntering`

Leaving	Configures the digitizer to arm on signals when they leave the given arming window. The end-user specifies the high and low thresholds with the Window Arm High Threshold and Window Arm Low Threshold properties.		
		.NET	`WindowCondition.Leaving`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`
		COM	`IviDigitizerWindowConditionLeaving`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

23.4 IviDigitizerWindowArm Behavior Model

The IviDigitizerWindowArm extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

23.5 IviDigitizerWindowArm Compliance Notes

For a specific driver to comply with the IviDigitizerWindowArm extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

24. IviDigitizerTriggerModifier Extension Group

24.1 IviDigitizerTriggerModifier Overview

The IviDigitizerTriggerModifier extension group provides support for digitizers that can specify the behavior of the triggering subsystem in the absence of the configured trigger.

24.2 IviDigitizerTriggerModifier Attributes

The IviDigitizerTriggerModifier capability group defines the following attribute:

- Trigger Modifier

This section describes the behavior and requirements of this attribute.

24.2.1 Trigger Modifier

Data Type	Access	Applies to	Coercion	High Level Functions
`ViInt32`	`R/W`	N/A	None	Configure Trigger Modifier (IVI-C only)

.NET Property Name

`Trigger.Modifier`

.NET Enumeration Name

`TriggerModifier`

COM Property Name

`Trigger.Modifier`

COM Enumeration Name

`IviDigitizerTriggerModifierEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_MODIFIER`

Description

Specifies the trigger modifier. The trigger modifier determines the digitizer’s behavior in the absence of the configured trigger.

Defined Values

Name	***Description***		
		Language	***Identifier***
No Trigger Modifier	The digitizer waits until the trigger the end-user specifies occurs.		
		.NET	`TriggerModifier.None`
		C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_NONE`
		COM	`IviDigitizerTriggerModifierNone`
Auto	The digitizer automatically triggers if the configured trigger does not occur within the digitizer timeout period.		
		.NET	`TriggerModifier.Auto`
		C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO`
		COM	`IviDigitizerTriggerModifierAuto`
Auto Level	The digitizer adjusts the trigger level if the trigger the end-user specifies does not occur.		
		.NET	`TriggerModifier.AutoLevel`
		C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO_LEVEL`
		COM	`IviDigitizerTriggerModifierAutoLevel`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

Compliance Notes

1. Instrument driver shall support the value No Trigger Modifier and at least one of the following values:
    - Auto
    - Auto Level
2. If an IVI-C class driver defines additional values for this attribute, the actual values shall be greater than or equal to `IVIDIGITIZER_VAL_TRIGGER_MOD_CLASS_EXT_BASE` and less than `IVIDIGITIZER_VAL_TRIGGER_MOD_SPECIFIC_EXT_BASE`.
3. If an IVI-C specific driver defines additional values for this attribute, the actual values shall be greater than or equal to `IVIDIGITIZER_VAL_TRIGGER_MOD_SPECIFIC_EXT_BASE`.

24.3 IviDigitizerTriggerModifier Functions

The IviDigitizerTriggerModifier capability group defines the following function:

- Configure Trigger Modifier (IVI-C only)

This section describes the behavior and requirements of this function.

24.3.1 Configure Trigger Modifier (IVI-C Only)

Description

This function configures the digitizer's trigger modifier.

.NET Method Prototype

```
N/A
```

(use the `Trigger.Modifier` property)

COM Method Prototype

```
N/A
```

(use the `Trigger.Modifier` property)

C Prototype

```
ViStatus IviDigitizer_ConfigureTriggerModifier (ViSession Vi,
                                                ViInt32 TriggerModifier);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle	`ViSession`
`TriggerModifier`	Specifies the method the digitizer uses in the absence of trigger conditions. The driver sets the Trigger Modifier attribute to this value. See the attribute description for more information and defined values.	`ViInt32`

Defined Values for the TriggerModifier Parameter

Name	*Description*		
		Language	*Identifier*
No Trigger Modifier	The digitizer waits until the trigger the end-user specifies occurs.		
		C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_NONE`
		COM	`IviDigitizerTriggerModifierNone`
Auto	The digitizer automatically triggers if the configured trigger does not occur within the digitizer timeout period.		
		C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO`
		COM	`IviDigitizerTriggerModifierAuto`
Auto Level	The digitizer adjusts the trigger level if the trigger the end-user specifies does not occur.		
		C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO_LEVEL`
		COM	`IviDigitizerTriggerModifierAutoLevel`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

24.4 IviDigitizerTriggerModifier Behavior Model

The IviDigitizerTriggerModifier group uses the behavior model defined by the IviDigitizerBase Capabilities.

24.5 IviDigitizerTriggerModifier Compliance Notes

For a specific driver to comply with the IviDigitizerTriggerModifier extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

25. IviDigitizerMultiTrigger Extension Group

25.1 IviDigitizerMultiTrigger Overview

The IviDigitizerMultiTrigger extension group supports digitizers with the ability to trigger on combinations of trigger sources and their associated trigger conditions. The user has the ability to logically OR trigger sources together or to logically AND them together and specify the result as the overall trigger source.

25.2 IviDigitizerMultiTrigger Attributes

The IviDigitizerMultiTrigger extension group defines the following attributes:

- Trigger Source List
- Trigger Source Operator

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

25.2.1 Trigger Source List

Data Type	Access	Applies To	Coercion	High Level Functions
`ViString`	R/W	N/A	None	Configure Multi Trigger

.NET Property Name

`Trigger.MultiTrigger.SourceList`

COM Property Name

`Trigger.MultiTrigger.SourceList`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_SOURCE_LIST`

Description

A comma separated list of source names to be used in a multi trigger scenario. This attribute only affects instrument behavior when the Trigger Source Operator attribute is set to AND or OR. When Trigger Source Operator is set to AND, the trigger conditions associated with each source in this list must be simultaneously satisfied in order to trigger the digitizer. When Trigger Source Operator is set to OR, the first trigger source in the list that satisfies its trigger conditions will trigger the digitizer. Any valid name used for the Trigger Source attribute may be used in this list. A trigger source may appear only once in the list. If a name in the list is not recognized, the driver returns the Unknown Channel Name error. See IVI-3.2 for the definition of this error.

If an IVI driver supports a trigger source and the trigger source is listed in IVI-3.3 *Cross Class Capabilities Specification*, Section 3, then the IVI driver shall accept the standard string for that trigger source. This attribute is case insensitive, but case preserving. That is, the setting is case insensitive but when reading it back the programmed case is returned. IVI specific drivers may define new trigger source strings for trigger sources that are not defined by IVI-3.3 *Cross Class Capabilities Specification* if needed.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

25.2.2 Trigger Source Operator

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	N/A	None	Configure Multi Trigger

.NET Property Name

`Trigger.MultiTrigger.SourceOperator`

.NET Enumeration Name

`TriggerSourceOperator`

COM Property Name

`Trigger.MultiTrigger.SourceOperator`

COM Enumeration Name

`IviDigitizerTriggerSourceOperatorEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_SOURCE_OPERATOR`

Description

Specifies the boolean operation to apply to the trigger sources specified by the Trigger Source List attribute. See the definition of that attribute for details.

Defined Values

Name	***Description***		
		Language	***Identifier***
AND	Trigger sources are AND'd together. The digitizer triggers when all configured trigger source conditions are satisfied.		
		.NET	`TriggerSourceOperator.And`
		C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_AND`
		COM	`IviDigitizerTriggerSourceOperatorAND`
OR	Trigger sources are OR'd together. The digitizer triggers when the first configured trigger source condition is satisfied.		
		.NET	`TriggerSourceOperator.Or`
		C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_OR`
		COM	`IviDigitizerTriggerSourceOperatorOR`
None	No operator is applied to the configured list of trigger sources. The trigger source list is ignored, and the digitizer triggers when the active trigger source, given by the Active Trigger Source attribute, and its associated trigger conditions are satisfied.		
		.NET	`TriggerSourceOperator.None`
		C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_NONE`

		COM	`IviDigitizerTriggerSourceOperatorNone`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

25.3 IviDigitizerMultiTrigger Functions

The IviDigitizerMultiTrigger extension group defines the following function:

- Configure Multi Trigger

This section describes the behavior and requirements of this function.

25.3.1 Configure Multi Trigger

Description

Configures the digitizer to trigger based on multiple trigger sources. The digitizer can be instructed to trigger when any one of multiple trigger source conditions are met or when all specified trigger source conditions are met.

.NET Method Prototype

```
void Trigger.MultiTrigger.Configure (String sourceList,
                                     TriggerSourceOperator operator);
```

COM Method Prototype

```
HRESULT Trigger.MultiTrigger.Configure (
                        [in] BSTR SourceList,
                        [in] IviDigitizerTriggerSourceOperatorEnum Operator);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureMultiTrigger (ViSession Vi,
                                             ViConstString SourceList,
                                             ViInt32 Operator);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`SourceList`	A comma separated list of source names to be used in a multi trigger scenario. The driver uses this value to set the Trigger Source List attribute. See the attribute description for more details.	`ViConstString`
`Operator`	Specifies the boolean operation to apply to the trigger sources specified by the SourceList parameter. The driver uses this value to set the Trigger Source Operator attribute. See the attribute description for more details.	`ViInt32`

Defined Values for the Operator Parameter

Name	***Description***		
		Language	***Identifier***
AND	Trigger sources are AND'd together. The digitizer triggers when all configured trigger source conditions are satisfied.		
		.NET	`TriggerSourceOperator.And`
		C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_AND`
		COM	`IviDigitizerTriggerSourceOperatorAND`
OR	Trigger sources are OR'd together. The digitizer triggers when the first configured trigger source condition is satisfied.		
		.NET	`TriggerSourceOperator.Or`
		C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_OR`

		COM	`IviDigitizerTriggerSourceOperatorOR`
None	No operator is applied to the configured list of trigger sources. The trigger source list is ignored, and the digitizer trigger when the active trigger source, given by the Active Trigger Source attribute, and its associated trigger conditions are satisfied.		
		.NET	`TriggerSourceOperator.None`
		C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_NONE`
		COM	`IviDigitizerTriggerSourceOperatorNone`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

25.4 IviDigitizerMultiTrigger Behavior Model

The IviDigitizerMultiTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

25.5 IviDigitizerMultiTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerMultiTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

26. IviDigitizerPretriggerSamples Extension Group

26.1 IviDigitizerPretriggerSamples Overview

The IviDigitizerPretriggerSamples extension group supports digitizers with the ability to specify pretrigger samples, which is defined as the number of samples needed to fill up the data buffer with pretrigger data. This is often used to capture as much pretrigger data as possible without losing important events.

26.2 IviDigitizerPretriggerSamples Attributes

The IviDigitizerPretriggerSamples extension group defines the following attributes:

- Pretrigger Samples

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

26.2.1 Pretrigger Samples

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt64`	R/W	N/A	None	Configure Pretrigger Samples (IVI-C only)

.NET Property Name

`Trigger.PretriggerSamples`

COM Property Name

`Trigger.PretriggerSamples`

C Constant Name

`IVIDIGITIZER_ATTR_PRETRIGGER_SAMPLES`

Description

Specifies the number of samples that must be collected before a trigger event will be recognized. The Pretrigger Samples attribute affects instrument operation only when the digitizer requires multiple acquisitions to build a complete waveform. If Trigger Holdoff and Pretrigger Samples are are both non-zero, then both conditions must be satisfied before the digitizer will accept a trigger.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

26.3 IviDigitizerPretriggerSamples Functions

The IviDigitizerPretriggerSamples extension group defines the following function:

- Configure Pretrigger Samples (IVI-C only)

This section describes the behavior and requirements of this function.

26.3.1 Configure Pretrigger Samples (IVI-C Only)

Description

This function configures the digitizer's pretrigger samples.

.NET Method Prototype

```
N/A
(use the Trigger.PretriggerSamples property)
```

COM Method Prototype

```
N/A
(use the Trigger.PretriggerSamples property)
```

C Prototype

```
ViStatus IviDigitizer_ConfigurePretriggerSamples (ViSession Vi,
                                                   ViInt64 PretriggerSamples);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`PretriggerSamples`	Specifies the number of samples that must be collected before a trigger event will be recognized. See the attribute description for more information. The driver sets the Pretrigger Samples attribute to this value.	`ViInt64`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

26.4 IviDigitizerPretriggerSamples Behavior Model

The IviDigitizerPretriggerSamples extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

26.5 IviDigitizerPretriggerSamples Compliance Notes

For a specific driver to comply with the IviDigitizerPretriggerSamples extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

27. IviDigitizerTriggerHoldoff Extension Group

27.1 IviDigitizerTriggerHoldoff Overview

The IviDigitizerTriggerHoldoff extension group supports digitizers with the ability to specify a trigger holdoff, which is defined as the length of time the digitizer waits after it detects a trigger until the digitizer enables the trigger subsystem to detect another trigger. Trigger holdoff is often used avoid triggering in the middle of a long signal. This allows the digitizer to skip over uninteresting portions of the signal.

27.2 IviDigitizerTriggerHoldoff Attributes

The IviDigitizerTriggerHoldoff extension group defines the following attributes:

- Trigger Holdoff

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

27.2.1 Trigger Holdoff

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	N/A	Down	Configure Trigger Holdoff (IVI-C only)

.NET Property Name

`Trigger.Holdoff`

COM Property Name

`Trigger.Holdoff`

C Constant Name

`IVIDIGITIZER_ATTR_TRIGGER_HOLDOFF`

Description

Specifies the length of time the digitizer waits after it detects a trigger until the digitizer enables the trigger subsystem to detect another trigger. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan. The Trigger Holdoff attribute affects instrument operation only when the digitizer requires multiple acquisitions to build a complete waveform. If Trigger Holdoff and PretriggerSamples are both non-zero, then both conditions must be satisfied before the digitizer will accept a trigger. Note: Many digitizers have a small, non-zero value as the minimum value for this attribute. To configure the instrument to use the shortest trigger hold-off, the user can specify a value of zero for this attribute. Therefore, the IVI Class-Compliant specific driver shall coerce any value between zero and the minimum value to the minimum value. No other coercion is allowed on this attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

27.3 IviDigitizerTriggerHoldoff Functions

The IviDigitizerTriggerHoldoff extension group defines the following function:

- Configure Trigger Holdoff (IVI-C only)

This section describes the behavior and requirements of this function.

27.3.1 Configure Trigger Holdoff (IVI-C Only)

Description

This function configures the digitizer’s trigger holdoff.

.NET Method Prototype

```
N/A
(use the Trigger.Holdoff property)
```

COM Method Prototype

```
N/A
(use the Trigger.Holdoff property)
```

C Prototype

```
ViStatus IviDigitizer_ConfigureTriggerHoldoff (ViSession Vi,
                                               ViReal64 TriggerHoldoff);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`TriggerHoldoff`	Specifies the length of time the digitizer waits after it detects a trigger until the digitizer enables the trigger subsystem to detect another trigger. The units are seconds. The driver sets the Trigger Holdoff attribute to this value. See the attribute description for more information.	`ViReal64`

Return Values (C)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

27.4 IviDigitizerTriggerHoldoff Behavior Model

The IviDigitizerTriggerHoldoff extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

27.5 IviDigitizerTriggerHoldoff Compliance Notes

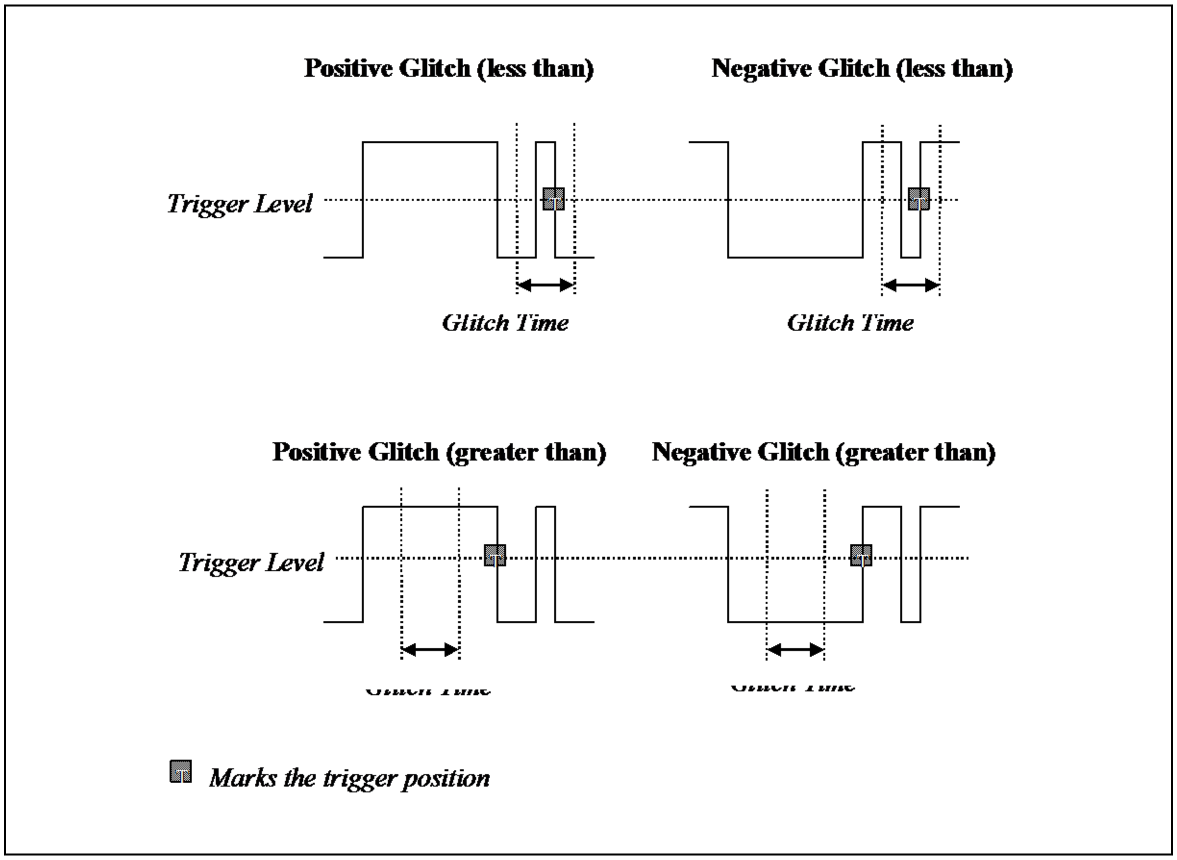
For a specific driver to comply with the IviDigitizerTriggerHoldoff extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

28. IviDigitizerGlitchTrigger Extension Group

28.1 IviDigitizerGlitchTrigger Overview

In addition to the fundamental capabilities, the IviDigitizerGlitchTrigger extension group defines extensions for digitizers that can trigger on a "glitch" pulses.

A glitch occurs when the digitizer detects a pulse width that is less than or a greater than a specified glitch duration. The figure below shows both positive and negative glitches for the "less than" condition as well as the positive "greater than" glitch.



Positive Glitch (less than)
Negative Glitch (less than)
Trigger Level
Glitch Time
Glitch Time
Positive Glitch (greater than)
Negative Glitch (greater than)
Trigger Level
Marks the trigger position

Figure 28-1 Glitch Triggers

With the IviDigitizerGlitchTrigger extension group the end-user can select whether a positive glitch, negative glitch, or either triggers the acquisition.

28.2 IviDigitizerGlitchTrigger Attributes

The IviDigitizerGlitchTrigger extension group defines the following attributes:

- Glitch Trigger Condition
- Glitch Trigger Polarity
- Glitch Trigger Width

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

28.2.1 Glitch Trigger Condition

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure Glitch Trigger Source

.NET Property Name

`Trigger.Sources[].Glitch.Condition`

.NET Enumeration Name

`GlitchCondition`

COM Property Name

`Trigger.Sources.Item().Glitch.Condition`

COM Enumeration Name

`IviDigitizerGlitchConditionEnum`

C Constant Name

`IVIDIGITIZER_ATTR_GLITCH_TRIGGER_CONDITION`

Description

Specifies the glitch condition. This attribute determines whether the glitch trigger happens when the digitizer detects a pulse with a width less than or greater than the width value.

Defined Values

Name	***Description***		
		Language	***Identifier***
Less Than	The digitizer triggers when the pulse width is less than the value you specify with the Glitch Trigger Width attribute.		
		.NET	`GlitchCondition.LessThan`
		C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`
		COM	`IviDigitizerGlitchConditionLessThan`
Greater Than	The digitizer triggers when the pulse width is greater than the value you specify with the Glitch Trigger Width attribute.		
		.NET	`GlitchCondition.GreaterThan`
		C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`
		COM	`IviDigitizerGlitchConditionGreaterThan`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

28.2.2 Glitch Trigger Polarity

Data Type	Access	Applies to	Coercion	High Level Functions
`ViInt32`	`R/W`	TriggerSource	None	Configure Glitch Trigger Source

.NET Property Name

`Trigger.Sources[].Glitch.Polarity`

.NET Enumeration Name

`GlitchPolarity`

COM Property Name

`Trigger.Sources.Item().Glitch.Polarity`

COM Enumeration Name

`IviDigitizerGlitchPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_GLITCH_TRIGGER_POLARITY`

Description

Specifies the polarity of the glitch that triggers the digitizer.

Defined Values

Name	***Description***		
		Language	***Identifier***
Glitch Positive	The digitizer triggers on a positive glitch.		
		.NET	`GlitchPolarity.Positive`
		C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`
		COM	`IviDigitizerGlitchPolarityPositive`
Glitch Negative	The digitizer triggers on a negative glitch.		
		.NET	`GlitchPolarity.Negative`
		C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`
		COM	`IviDigitizerGlitchPolarityNegative`
Glitch Either	The digitizer triggers on either a positive or negative glitch.		
		.NET	`GlitchPolarity.Either`
		C	`IVIDIGITIZER_VAL_GLITCH_EITHER`
		COM	`IviDigitizerGlitchPolarityEither`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

28.2.3 Glitch Trigger Width

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	TriggerSource	None	Configure Glitch Trigger Source

.NET Property Name

`Trigger.Sources[].Glitch.Width`

COM Property Name

`Trigger.Sources.Item().Glitch.Width`

C Constant Name

`IVIDIGITIZER_ATTR_GLITCH_TRIGGER_WIDTH`

Description

Specifies the glitch width. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan. The digitizer triggers when it detects a pulse with a width less than or greater than this value, depending on the Glitch Condition attribute.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

28.3 IviDigitizerGlitchTrigger Functions

The IviDigitizerGlitchTrigger extension group defines the following function:

- Configure Glitch Trigger Source

This section describes the behavior and requirements of this function.

28.3.1 Configure Glitch Trigger Source

Description

This function configures the glitch trigger. A glitch trigger occurs when the trigger signal has a pulse with a width that is less than or greater than the glitch width. The end user specifies which comparison criterion to use with the Glitch Condition parameter. The end-user specifies the glitch width with the Width parameter. The end-user specifies the polarity of the pulse with the Polarity parameter. The trigger does not actually occur until the edge of a pulse that corresponds to the Width and Polarity crosses the threshold the end-user specifies in the Level parameter. This function affects instrument behavior only if the trigger type is Glitch Trigger. Set the trigger type and trigger coupling before calling this function.

.NET Method Prototype

```
void Trigger.Sources[].Glitch.Configure (Double level,
                                         PrecisionTimeSpan width,
                                         GlitchPolarity polarity,
                                         GlitchCondition condition);
```

COM Method Prototype

```
HRESULT Trigger.Sources.Item().Glitch.Configure ([in] double Level,
                               [in] double Width,
                               [in] IviDigitizerGlitchPolarityEnum Polarity,
                               [in] IviDigitizerGlitchConditionEnum Condition);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureGlitchTriggerSource (ViSession Vi,
                                                    ViConstString Source,
                                                    ViReal64 Level,
                                                    ViReal64 Width,
                                                    ViInt32 Polarity,
                                                    ViInt32 Condition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the trigger source that is to be configured.	`ViString`
`Level`	Specifies the trigger level. This value sets the Trigger Level attribute.	`ViReal64`
`Width`	Specifies the glitch triggering glitch width. This value sets the Glitch Trigger Width attribute.	`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`
`Polarity`	Specifies the glitch polarity. This value sets the Glitch Trigger Polarity attribute.	`ViInt32`
`Condition`	Specifies the glitch condition. This value sets the Glitch Trigger Condition attribute.	`ViInt32`

Defined Values for the Polarity Parameter

Name	***Description***		
		Language	***Identifier***
Positive	The digitizer triggers on a positive glitch.		
		.NET	`GlitchPolarity.Positive`
		C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`
		COM	`IviDigitizerGlitchPolarityPositive`
Negative	The digitizer triggers on a negative glitch.		
		.NET	`GlitchPolarity.Negative`
		C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`
		COM	`IviDigitizerGlitchPolarityNegative`
Either	The digitizer triggers on either a positive or negative glitch.		
		.NET	`GlitchPolarity.Either`
		C	`IVIDIGITIZER_VAL_GLITCH_EITHER`
		COM	`IviDigitizerGlitchPolarityEither`

Defined Values for the Condition Parameter

Name	***Description***		
		Language	***Identifier***
Less Than	The digitizer triggers when the pulse width is less than the value you specify with the Glitch Trigger Width attribute.		
		.NET	`GlitchCondition.LessThan`
		C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`
		COM	`IviDigitizerGlitchConditionLessThan`
Greater Than	The digitizer triggers when the pulse width is greater than the value you specify with the Glitch Trigger Width attribute.		
		.NET	`GlitchCondition.GreaterThan`
		C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`
		COM	`IviDigitizerGlitchConditionGreaterThan`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

28.4 IviDigitizerGlitchTrigger Behavior Model

The IviDigitizerGlitchTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

28.5 IviDigitizerGlitchTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerGlitchTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

29. IviDigitizerRuntTrigger Extension Group

29.1 IviDigitizerRuntTrigger Overview

In addition to the fundamental capabilities, the IviDigitizerRuntTrigger extension group defines extensions for digitizers with the capability to trigger on “runt” pulses.

A runt condition occurs when the digitizer detects a positive or negative going pulse that crosses one voltage threshold but fails to cross a second threshold before re-crossing the first. The figure below shows both positive and negative runt polarities.

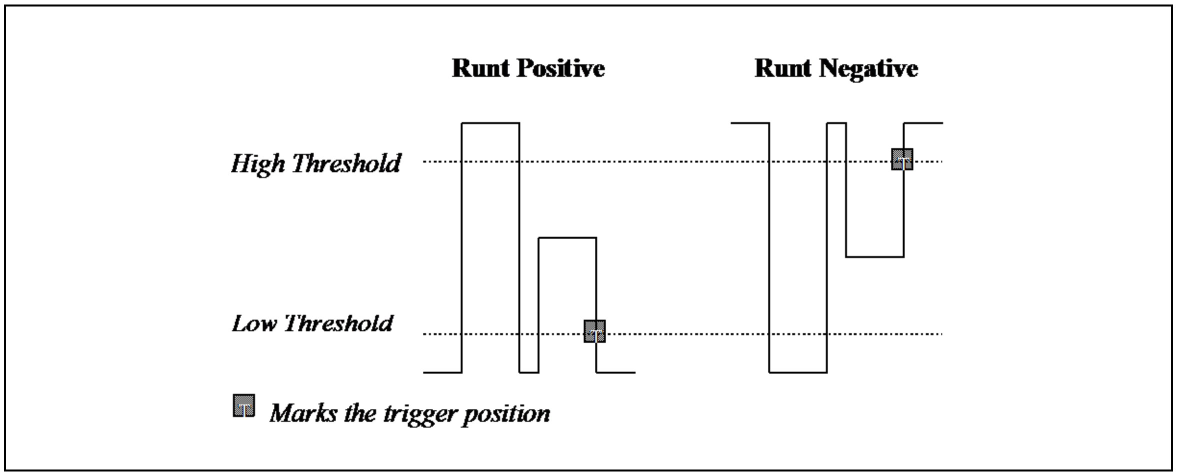


Figure 29-1 Runt Triggers

With the IviDigitizerRuntTrigger extension group the end-user can select whether a positive runt, negative runt, or either triggers the acquisition.

29.2 IviDigitizerRuntTrigger Attributes

The IviDigitizerRuntTrigger extension group defines the following attributes:

- Runt Trigger High Threshold
- Runt Trigger Low Threshold
- Runt Trigger Polarity

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

29.2.1 Runt Trigger High Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	TriggerSource	None	Configure Runt Trigger Source

COM Property Name

`Trigger.Sources[].Runt.ThresholdHigh`

COM Property Name

`Trigger.Sources.Item().Runt.ThresholdHigh`

C Constant Name

`IVIDIGITIZER_ATTR_RUNT_TRIGGER_HIGH_THRESHOLD`

Description

The high threshold the digitizer uses for runt triggering. The units are volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

29.2.2 Runt Trigger Low Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	TriggerSource	None	Configure Runt Trigger Source

COM Property Name

`Trigger.Sources[].Runt.ThresholdLow`

COM Property Name

`Trigger.Sources.Item().Runt.ThresholdLow`

C Constant Name

`IVIDIGITIZER_ATTR_RUNT_TRIGGER_LOW_THRESHOLD`

Description

The low threshold the digitizer uses for runt triggering. The units are volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

29.2.3 Runt Trigger Polarity

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure Runt Trigger Source

.NET Property Name

`Trigger.Sources[].Runt.Polarity`

.NET Enumeration Name

`RuntPolarity`

COM Property Name

`Trigger.Sources.Item().Runt.Polarity`

COM Enumeration Name

`IviDigitizerRuntPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_RUNT_TRIGGER_POLARITY`

Description

Specifies the polarity of the runt that triggers the digitizer.

Defined Values

Name	***Description***		
		Language	***Identifier***
Positive	The digitizer triggers on a positive runt. A positive runt occurs when a rising edge crosses the low runt threshold and does not cross the high runt threshold before re-crossing the low runt threshold.		
		.NET	`RuntPolarity.Positive`
		C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`
		COM	`IviDigitizerRuntPolarityPositive`
Negative	The digitizer triggers on a negative runt. A negative runt occurs when a falling edge crosses the high runt threshold and does not cross the low runt threshold before re-crossing the high runt threshold.		
		.NET	`RuntPolarity.Negative`
		C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`
		COM	`IviDigitizerRuntPolarityNegative`
Either	The digitizer triggers on either a positive or negative runt.		
		.NET	`RuntPolarity.Either`
		C	`IVIDIGITIZER_VAL_RUNT_EITHER`
		COM	`IviDigitizerRuntPolarityEither`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

29.3 IviDigitizerRuntTrigger Functions

The IviDigitizerRuntTrigger extension group defines the following function:

- Configure Runt Trigger Source

This section describes the behavior and requirements of this function.

29.3.1 Configure Runt Trigger Source

Description

This function configures the runt trigger. A runt trigger occurs when the trigger signal crosses one of the runt thresholds twice without crossing the other runt threshold. The end-user specifies the runt thresholds with the RuntLowThreshold and RuntHighThreshold parameters. The end-user specifies the polarity of the runt with the RuntPolarity parameter. This function affects instrument behavior only if the trigger type is Runt Trigger. Set the trigger type and trigger coupling before calling this function.

.NET Method Prototype

```
void Trigger.Sources[].Runt.Configure (Double thresholdLow,
                                       Double thresholdHigh,
                                       RuntPolarity polarity);
```

COM Method Prototype

```
HRESULT Trigger.Sources.Item().Runt.Configure ([in] double ThresholdLow,
                                   [in] double ThresholdHigh,
                                   [in] IviDigitizerRuntPolarityEnum Polarity);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureRuntTriggerSource (ViSession Vi,
                                                  ViConstString Source,
                                                  ViReal64 ThresholdLow,
                                                  ViReal64 ThresholdHigh,
                                                  ViInt32 Polarity);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the trigger source that is to be configured.	`ViString`
`ThresholdLow`	Sets the runt triggering low threshold in volts. This value sets the Runt Trigger Low Threshold attribute.	`ViReal64`
`ThresholdHigh`	Sets the runt triggering high threshold in volts. This value sets the Runt Trigger High Threshold attribute.	`ViReal64`
`Polarity`	Sets the runt polarity. This value sets the Runt Trigger Polarity attribute.	`ViInt32`

Defined Values for the Polarity Parameter

Name	***Description***		
		Language	***Identifier***
Positive	The digitizer triggers on a positive runt. A positive runt occurs when a rising edge crosses the low runt threshold and does not cross the high runt threshold before re-crossing the low runt threshold.		
		.NET	`RuntPolarity.Positive`
		C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`
		COM	`IviDigitizerRuntPolarityPositive`
Negative	The digitizer triggers on a negative runt. A negative runt occurs when a falling edge crosses the high runt threshold and does not cross the low runt threshold before re-crossing the high runt threshold.		
		.NET	`RuntPolarity.Negative`
		C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`
		COM	`IviDigitizerRuntPolarityNegative`
Either	The digitizer triggers on either a positive or negative runt.		
		.NET	`RuntPolarity.Either`
		C	`IVIDIGITIZER_VAL_RUNT_EITHER`
		COM	`IviDigitizerRuntPolarityEither`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

29.4 IviDigitizerRuntTrigger Behavior Model

The IviDigitizerRuntTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

29.5 IviDigitizerRuntTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerRuntTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

30. IviDigitizerSoftwareTrigger Extension Group

30.1 IviDigitizerSoftwareTrigger Overview

The IviDigitizerSoftwareTrigger extension group supports digitizers with the ability to trigger an acquisition.

30.2 IviDigitizerSoftwareTrigger Functions

The IviDigitizerSoftwareTrigger extension group defines the following function:

- Send Software Trigger

This section describes the behavior and requirements of this function.

30.2.1 Send Software Trigger

Description

Refer to IVI-3.3: Standard Cross-Class Capabilities Specification for the prototype and complete description of this function.

.NET Method Prototype

```
void Trigger.SendSoftwareTrigger ();
```

COM Method Prototype

```
HRESULT Trigger.SendSoftwareTrigger ();
```

C Prototype

```
ViStatus IviDigitizer_SendSoftwareTrigger (ViSession Vi);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return. This function can also return this additional standard cross-class status code:

- Trigger Not Software

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method. This method can also return this additional standard cross-class exception:

- Trigger Not Software

30.3 IviDigitizerSoftwareTrigger Behavior Model

The IviDigitizerSoftwareTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

30.4 IviDigitizerSoftwareTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerSoftwareTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

31. IviDigitizerTVTrigger Extension Group

31.1 IviDigitizerTVTrigger Overview

The IviDigitizerTVTrigger extension group defines extensions for digitizers capable of triggering on TV signals.

31.2 IviDigitizerTVTrigger Attributes

The IviDigitizerTVTrigger extension group defines the following attributes:

- TV Trigger Event
- TV Trigger Line Number
- TV Trigger Polarity
- TV Trigger Signal Format

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

31.2.1 TV Trigger Event

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure TV Trigger Source

.NET Property Name

`Trigger.Sources[].TV.Event`

.NET Enumeration Name

`TVTriggerEvent`

COM Property Name

`Trigger.Sources.Item().TV.Event`

COM Enumeration Name

`IviDigitizerTVTriggerEventEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TV_TRIGGER_EVENT`

Description

Specifies the event on which the digitizer triggers.

Defined Values

Name	***Description***		
		Language	***Identifier***
Field1	Sets the digitizer to trigger on field 1 of the video signal.		
		.NET	`TVTriggerEvent.Field1`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`
		COM	`IviDigitizerTVTriggerEventField1`
Field2	Sets the digitizer to trigger on field 2 of the video signal.		
		.NET	`TVTriggerEvent.Field2`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`
		COM	`IviDigitizerTVTriggerEventField2`
Any Field	Sets the digitizer to trigger on any field.		
		.NET	`TVTriggerEvent.AnyField`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`
		COM	`IviDigitizerTVTriggerEventAnyField`
Any Line	Sets the digitizer to trigger on any line.		
		.NET	`TVTriggerEvent.AnyLine`

		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`
		COM	`IviDigitizerTVTriggerEventAnyLine`
Line Number	Sets the digitizer to trigger on a specific line number you specify with the TV Trigger Line Number attribute.		
		.NET	`TVTriggerEvent.LineNumber`
		C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`
		COM	`IviDigitizerTVTriggerEventLineNumber`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

31.2.2 TV Trigger Line Number

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	N/A

.NET Property Name

`Trigger.Sources[].TV.LineNumber`

COM Property Name

`Trigger.Sources.Item().TV.LineNumber`

C Constant Name

`IVIDIGITIZER_ATTR_TV_TRIGGER_LINE_NUMBER`

Description

Specifies the line on which the digitizer triggers. The driver uses this attribute when the TV Trigger Event is set to TV Event Line Number. The line number setting is independent of the field. For example, to trigger on the first line of the second field, the user must configure the line number to the value of 263 (if we presume that field one had 262 lines).

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

31.2.3 TV Trigger Polarity

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure TV Trigger Source

.NET Property Name

`Trigger.Sources[].TV.Polarity`

.NET Enumeration Name

`TVTriggerPolarity`

COM Property Name

`Trigger.Sources.Item().TV.Polarity`

COM Enumeration Name

`IviDigitizerTVTriggerPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TV_TRIGGER_POLARITY`

Description

Specifies the polarity of the TV signal.

Defined Values

Name	*Description*		
		Language	*Identifier*
Positive	Configures the digitizer to trigger on a positive video sync pulse.		
		.NET	`TVTriggerPolarity.Positive`
		C	`IVIDIGITIZER_VAL_TV_POSITIVE`
		COM	`IviDigitizerTVTriggerPolarityPositive`
Negative	Configures the digitizer to trigger on a negative video sync pulse.		
		.NET	`TVTriggerPolarity.Negative`
		C	`IVIDIGITIZER_VAL_TV_NEGATIVE`
		COM	`IviDigitizerTVTriggerPolarityNegative`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

31.2.4 TV Trigger Signal Format

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure TV Trigger Source

.NET Property Name

`Trigger.Sources[].TV.SignalFormat`

.NET Enumeration Name

`TVSignalFormat`

COM Property Name

`Trigger.Sources.Item().TV.SignalFormat`

COM Enumeration Name

`IviDigitizerTVSignalFormatEnum`

C Constant Name

`IVIDIGITIZER_ATTR_TV_TRIGGER_SIGNAL_FORMAT`

Description

Specifies the format of TV signal on which the digitizer triggers.

Defined Values

Name	*Description*		
		Language	*Identifier*
NTSC	Configures the digitizer to trigger on the NTSC signal format.		
		.NET	`TVSignalFormat.Ntsc`
		C	`IVIDIGITIZER_VAL_NTSC`
		COM	`IviDigitizerTVSignalFormatNTSC`
PAL	Configures the digitizer to trigger on the PAL signal format		
		.NET	`TVSignalFormat.Pal`
		C	`IVIDIGITIZER_VAL_PAL`
		COM	`IviDigitizerTVSignalFormatPAL`
SECAM	Configures the digitizer to trigger on the SECAM signal format		
		.NET	`TVSignalFormat.Secam`
		C	`IVIDIGITIZER_VAL_SECAM`
		COM	`IviDigitizerTVSignalFormatSECAM`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

31.3 IviDigitizerTVTrigger Functions

The IviDigitizerTVTrigger extension group defines the following function:

- ConfigureTV Trigger Source

This section describes the behavior and requirements of this function.

31.3.1 Configure TV Trigger Source

Description

This function configures the digitizer for TV triggering. It configures the TV signal format, the event and the signal polarity. This function affects instrument behavior only if the trigger type is TV Trigger. Set the Trigger Type and Trigger Coupling before calling this function.

.NET Method Prototype

```
void Trigger.Sources[].TV.Configure (TVSignalFormat signalFormat,
                                     TVTriggerEvent event,
                                     TVTriggerPolarity polarity);
```

COM Method Prototype

```
HRESULT Trigger.Sources.Item().TV.Configure (
                              [in] IviDigitizerTVSignalFormatEnum SignalFormat,
                              [in] IviDigitizerTVTriggerEventEnum Event,
                              [in] IviDigitizerTVTriggerPolarityEnum Polarity);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureTVTriggerSource (ViSession Vi,
                                                ViConstString Source,
                                                ViInt32 SignalFormat,
                                                ViInt32 Event,
                                                ViInt32 Polarity);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the trigger source that is to be configured.	`ViString`
`SignalFormat`	Specifies the TV trigger signal format. This value sets the TV Trigger Signal Format attribute.	`ViInt32`
`Event`	Specifies the TV trigger event. This value sets the TV Trigger Event attribute.	`ViInt32`
`Polarity`	Specifies the polarity of the TV trigger. This value sets the TV Trigger Polarity attribute.	`ViInt32`

Defined Values for the SignalFormat Parameter

Name	***Description***		
		Language	***Identifier***
NTSC	Configures the digitizer to trigger on the NTSC signal format.		
		.NET	`TVSignalFormat.Ntsc`
		C	`IVIDIGITIZER_VAL_NTSC`
		COM	`IviDigitizerTVSignalFormatNTSC`
PAL	Configures the digitizer to trigger on the PAL signal format		
		.NET	`TVSignalFormat.Pal`
		C	`IVIDIGITIZER_VAL_PAL`

		COM	`IviDigitizerTVSignalFormatPAL`
SECAM	Configures the digitizer to trigger on the SECAM signal format		
		.NET	`TVSignalFormat.Secam`
		C	`IVIDIGITIZER_VAL_SECAM`
		COM	`IviDigitizerTVSignalFormatSECAM`

Defined Values for the Event Parameter

Name	***Description***		
		Language	***Identifier***
Field1	Sets the digitizer to trigger on field 1 of the video signal.		
		.NET	`TVTriggerEvent.Field1`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`
		COM	`IviDigitizerTVTriggerEventField1`
Field2	Sets the digitizer to trigger on field 2 of the video signal.		
		.NET	`TVTriggerEvent.Field2`
		C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`
		COM	`IviDigitizerTVTriggerEventField2`
Any Field	Sets the digitizer to trigger on any field.		
		.NET	`TVTriggerEvent.AnyField`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`
		COM	`IviDigitizerTVTriggerEventAnyField`
Any Line	Sets the digitizer to trigger on any line.		
		.NET	`TVTriggerEvent.AnyLine`
		C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`
		COM	`IviDigitizerTVTriggerEventAnyLine`
Line Number	Sets the digitizer to trigger on a specific line number you specify with the TV Trigger Line Number attribute.		
		.NET	`TVTriggerEvent.LineNumber`
		C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`
		COM	`IviDigitizerTVTriggerEventLineNumber`

Defined Values for the Polarity Parameter

Name	***Description***		
		Language	***Identifier***
Positive	Configures the digitizer to trigger on a positive video sync pulse.		
		.NET	`TVTriggerPolarity.Positive`
		C	`IVIDIGITIZER_VAL_TV_POSITIVE`
		COM	`IviDigitizerTVTriggerPolarityPositive`
Negative	Configures the digitizer to trigger on a negative video sync pulse.		

		.NET	`TVTriggerPolarity.Negative`
		C	`IVIDIGITIZER_VAL_TV_NEGATIVE`
		COM	`IviDigitizerTVTriggerPolarityNegative`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

31.4 IviDigitizerTVTrigger Behavior Model

The IviDigitizerTVTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

31.5 IviDigitizerTVTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerTVTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

32. IviDigitizerWidthTrigger Extension Group

32.1 IviDigitizerWidthTrigger Overview

In addition to the fundamental capabilities, the IviDigitizerWidthTrigger extension group defines extensions for digitizers capable of triggering on user-specified pulse widths.

Width triggering occurs when the digitizer detects a positive or negative pulse with a width between, or optionally outside, the user-specified thresholds. The figure below shows positive and negative pulses that fall within the user-specified thresholds.

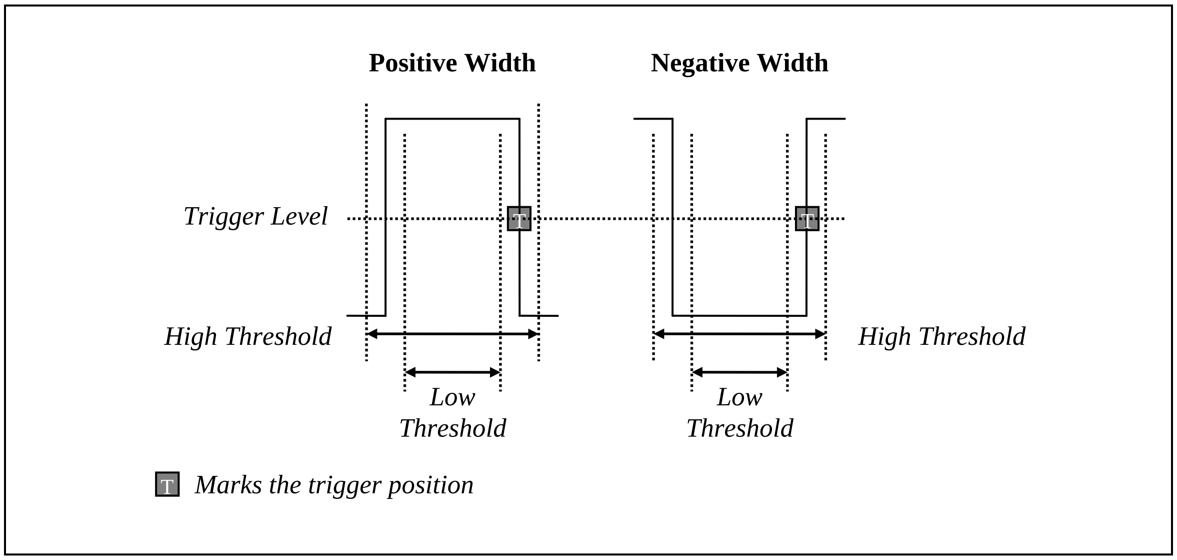


Figure 32-1 Width Triggers Within the Thresholds

The figure below shows positive and negative pulses that are not inside the user-specified thresholds.

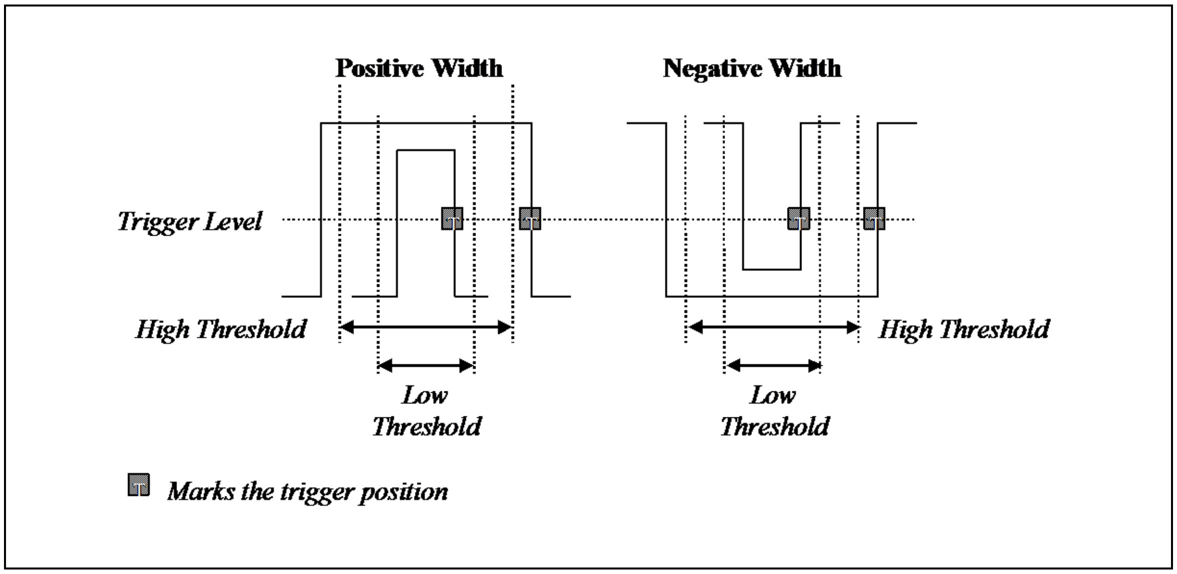


Figure 32-2 Width Triggers Outside the Thresholds

32.2 IviDigitizerWidthTrigger Attributes

The IviDigitizerWidthTrigger extension group defines the following attributes:

- Width Trigger Condition
- Width Trigger High Threshold
- Width Trigger Low Threshold
- Width Trigger Polarity

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

32.2.1 Width Trigger Condition

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure Width Trigger Source

.NET Property Name

`Trigger.Sources[].Width.Condition`

.NET Enumeration Name

`WidthCondition`

COM Property Name

`Trigger.Sources.Item().Width.Condition`

COM Enumeration Name

`IviDigitizerWidthConditionEnum`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_CONDITION`

Description

Specifies whether a pulse that is within or outside the high and low thresholds triggers the digitizer. The end-user specifies the high and low thresholds with the Width High Threshold and Width Low Threshold attributes.

Defined Values

Name	***Description***		
		Language	***Identifier***
Within	Configures the digitizer to trigger on pulses that have a width that is less than the high threshold and greater than the low threshold. The end-user specifies the high and low thresholds with the Width Trigger High Threshold and Width Trigger Low Threshold properties.		
		NET	`WidthCondition.Within`
		C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`
		COM	`IviDigitizerWidthConditionWithin`
Outside	Configures the digitizer to trigger on pulses that have a width that is either greater than the high threshold or less than a low threshold. The end-user specifies the high and low thresholds with the Width Trigger High Threshold and Width Trigger Low Threshold properties.		
		.NET	`WidthCondition.Outside`
		C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`
		COM	`IviDigitizerWidthConditionOutside`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

32.2.2 Width Trigger High Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	TriggerSource	None	Configure Width Trigger Source

.NET Property Name

```
Trigger.Sources[].Width.ThresholdHigh
```

COM Property Name

```
Trigger.Sources.Item().Width.ThresholdHigh
```

C Constant Name

```
IVIDIGITIZER_ATTR_WIDTH_TRIGGER_HIGH_THRESHOLD
```

Description

Specifies the high width threshold time. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

32.2.3 Width Trigger Low Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64 (C/COM)` `PrecisionTimeSpan (.NET)`	R/W	TriggerSource	None	Configure Width Trigger Source

.NET Property Name

`Trigger.Sources[].Width.ThresholdLow`

COM Property Name

`Trigger.Sources.Item().Width.ThresholdLow`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_LOW_THRESHOLD`

Description

Specifies the low width threshold time. For C and COM, the units are seconds. For .NET, the units are implicit in the definition of PrecisionTimeSpan.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

32.2.4 Width Trigger Polarity

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure Width Trigger Source

.NET Property Name

`Trigger.Sources[].Width.Polarity`

.NET Enumeration Name

`WidthPolarity`

COM Property Name

`Trigger.Sources.Item().Width.Polarity`

COM Enumeration Name

`IviDigitizerWidthPolarityEnum`

C Constant Name

`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_POLARITY`

Description

Specifies the polarity of the pulse that triggers the digitizer.

Defined Values

Name	*Description*		
		Language	*Identifier*
Positive	Configures the digitizer to trigger on positive pulses that have a width that meets the condition the user specifies with the Width Trigger Condition attribute.		
		.NET	`WidthPolarity.Positive`
		C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`
		COM	`IviDigitizerWidthPolarityPositive`
Negative	Configures the digitizer to trigger on negative pulses that have a width that meets the condition the user specifies with the Width Trigger Condition attribute.		
		.NET	`WidthPolarity.Negative`
		C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`
		COM	`IviDigitizerWidthPolarityNegative`
Either	Configures the digitizer to trigger on either positive or negative pulses that have a width that meets the condition the user specifies with the Width Trigger Condition attribute.		
		.NET	`WidthPolarity.Either`

		C	IVIDIGITIZER_VAL_WIDTH_EITHER
		COM	IviDigitizerWidthPolarityEither

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

32.3 IviDigitizerWidthTrigger Functions

The IviDigitizerWidthTrigger extension group defines the following function:

- Configure Width Trigger Source

This section describes the behavior and requirements of this function.

32.3.1 Configure Width Trigger Source

Description

Configures the width trigger Source, Level, ThresholdLow, ThresholdHigh, Polarity, and Condition. A width trigger occurs when a pulse, that passes through Level, with a width between or outside, the width thresholds is detected.

.NET Method Prototype

```
void Trigger.Sources[].Width.Configure (Double level,
                                        PrecisionTimeSpan thresholdLow,
                                        PrecisionTimeSpan thresholdHigh,
                                        WidthPolarity polarity,
                                        WidthCondition condition);
```

COM Method Prototype

```
HRESULT Trigger.Sources.Item().Width.Configure ([in] double Level,
                                [in] double ThresholdLow,
                                [in] double ThresholdHigh,
                                [in] IviDigitizerWidthPolarityEnum Polarity,
                                [in] IviDigitizerWidthConditionEnum Condition);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureWidthTriggerSource (ViSession Vi,
                                                   ViConstString Source,
                                                   ViReal64 Level,
                                                   ViReal64 ThresholdLow,
                                                   ViReal64 ThresholdHigh,
                                                   ViInt32 Polarity,
                                                   ViInt32 Condition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the trigger source that is to be configured.	`ViString`
`Level`	Trigger Level. This value sets the Trigger Level attribute.	`ViReal64`
`ThresholdLow`	Sets the width triggering low threshold. This value sets the Width Trigger Low Threshold attribute.	`ViReal64(C/COM)` `PrecisionTimeSpan (.NET)`
`ThresholdHigh`	Sets the width triggering high threshold. This value sets the Width Trigger High Threshold attribute.	`ViReal64(C/COM)` `PrecisionTimeSpan (.NET)`
`Polarity`	Sets the width polarity. This value sets the Width Trigger Polarity attribute.	`ViInt32`
`Condition`	Specifies whether a pulse that is within or outside the user-specified thresholds trigger waveform acquisition. This value sets the Width Trigger Condition attribute.	`ViInt32`

Defined Values for the Polarity Parameter

Name	*Description*		
		Language	*Identifier*
Positive	Configures the digitizer to trigger on positive pulses that have a width that meets the condition the user specifies with the Width Trigger Condition attribute.		
		.NET	`WidthPolarity.Positive`
		C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`
		COM	`IviDigitizerWidthPolarityPositive`
Negative	Configures the digitizer to trigger on negative pulses that have a width that meets the condition the user specifies with the Width Trigger Condition attribute.		
		.NET	`WidthPolarity.Negative`
		C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`
		COM	`IviDigitizerWidthPolarityNegative`
Either	Configures the digitizer to trigger on either positive or negative pulses that have a width that meets the condition the user specifies with the Width Trigger Condition attribute.		
		.NET	`WidthPolarity.Either`
		C	`IVIDIGITIZER_VAL_WIDTH_EITHER`
		COM	`IviDigitizerWidthPolarityEither`

Defined Values for the Condition Parameter

Name	*Description*		
		Language	*Identifier*
Within	Configures the digitizer to trigger on pulses that have a width that is less than the high threshold and greater than the low threshold. The end-user specifies the high and low thresholds with the Width Trigger High Threshold and Width Trigger Low Threshold properties.		
		.NET	`WidthCondition.Within`
		C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`
		COM	`IviDigitizerWidthConditionWithin`
Outside	Configures the digitizer to trigger on pulses that have a width that is either greater than the high threshold or less than a low threshold. The end-user specifies the high and low thresholds with the Width Trigger High Threshold and Width Trigger Low Threshold properties.		
		.NET	`WidthCondition.Outside`
		C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`
		COM	`IviDigitizerWidthConditionOutside`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

32.4 IviDigitizerWidthTrigger Behavior Model

The IviDigitizerWidthTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

32.5 IviDigitizerWidthTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerWidthTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

33. IviDigitizerWindowTrigger Extension Group

33.1 IviDigitizerWindowTrigger Overview

In addition to the fundamental capabilities, the IviDigitizerWindowTrigger extension group defines extensions for digitizers capable of triggering on user-specified voltage ranges.

Window triggering occurs when the digitizer detects a signal entering or leaving a user-specified set of thresholds. The figure below shows how the device triggers on signals entering the voltage range.

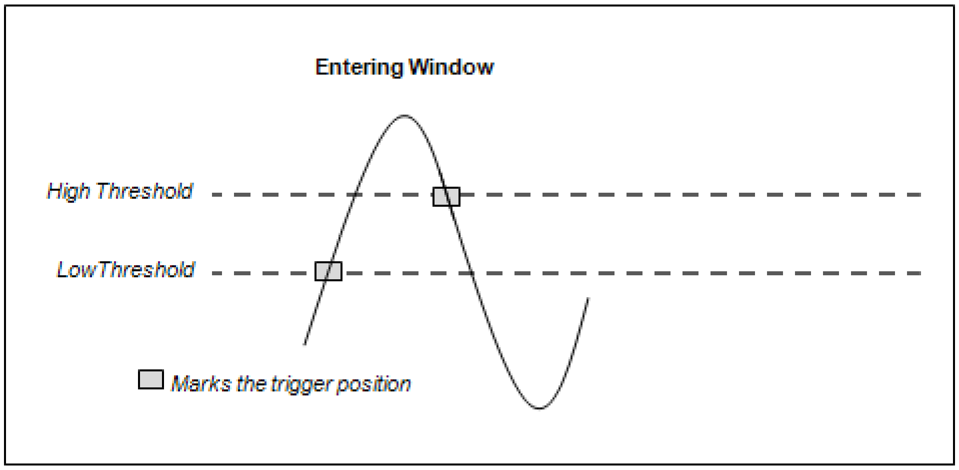


Figure 33-1 Window Triggers Entering the Thresholds

The figure below shows how the device triggers on signals leaving the voltage range.

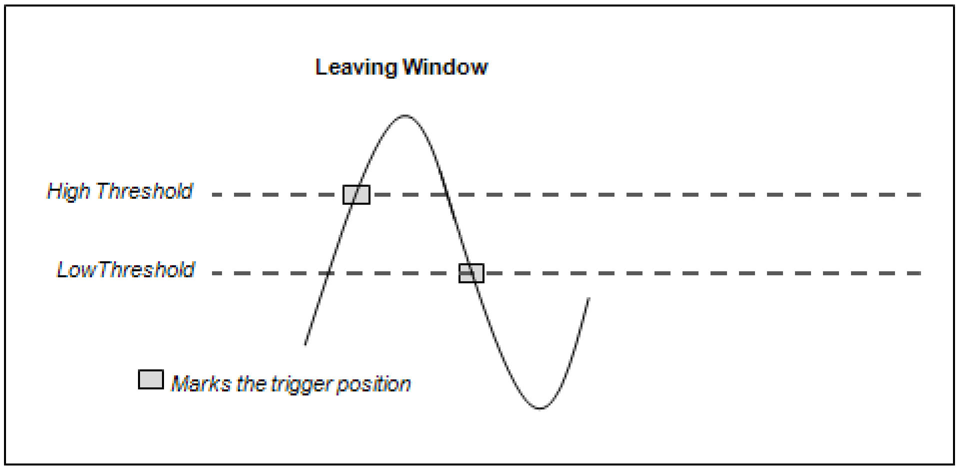


Figure 33-2 Window Triggers Leaving the Thresholds

33.2 IviDigitizerWindowTrigger Attributes

The IviDigitizerWindowTrigger extension group defines the following attributes:

- Window Trigger Condition
- Window Trigger High Threshold
- Window Trigger Low Threshold

This section describes the behavior and requirements of each attribute. The actual value for each attribute ID is defined in Section 31, *IviDigitizer Attribute ID Definitions.*

33.2.1 Window Trigger Condition

Data Type	Access	Applies To	Coercion	High Level Functions
`ViInt32`	R/W	TriggerSource	None	Configure Window Trigger Source

.NET Property Name

`Trigger.Sources[].Window.Condition`

.NET Enumeration Name

`WindowCondition`

COM Property Name

`Trigger.Sources.Item().Window.Condition`

COM Enumeration Name

`IviDigitizerWindowConditionEnum`

C Constant Name

`IVIDIGITIZER_ATTR_WINDOW_TRIGGER_CONDITION`

Description

Specifies whether a signal that is entering or leaving the voltage window defined by the high and low thresholds triggers the digitizer. The end-user specifies the high and low thresholds with the Window High Threshold and Window Low Threshold attributes.

Defined Values

Name	*Description*		
		Language	*Identifier*
Entering	Configures the digitizer to trigger on signals when they enter the given triggering window. The end-user specifies the high and low thresholds with the Window Trigger High Threshold and Window Trigger Low Threshold properties.		
		.NET	`WindowCondition.Entering`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`
		COM	`IviDigitizerWindowConditionEntering`
Leaving	Configures the digitizer to trigger on signals when they leave the given triggering window. The end-user specifies the high and low thresholds with the Window Trigger High Threshold and Window Trigger Low Threshold properties.		
		.NET	`WindowCondition.Leaving`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`
		COM	`IviDigitizerWindowConditionLeaving`

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

33.2.2 Window Trigger High Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	TriggerSource	None	Configure Window Trigger Source

.NET Property Name

```
Trigger.Sources[].Window.ThresholdHigh
```

COM Property Name

```
Trigger.Sources.Item().Window.ThresholdHigh
```

C Constant Name

```
IVIDIGITIZER_ATTR_WINDOW_TRIGGER_HIGH_THRESHOLD
```

Description

Specifies the high window threshold voltage in Volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

33.2.3 Window Trigger Low Threshold

Data Type	Access	Applies To	Coercion	High Level Functions
`ViReal64`	R/W	TriggerSource	None	Configure Window Trigger Source

.NET Property Name

`Trigger.Sources[].Window.ThresholdLow`

COM Property Name

`Trigger.Sources.Item().Window.ThresholdLow`

C Constant Name

`IVIDIGITIZER_ATTR_WINDOW_TRIGGER_LOW_THRESHOLD`

Description

Specifies the low window threshold voltage in Volts.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this property.

33.3 IviDigitizerWindowTrigger Functions

The IviDigitizerWindowTrigger extension group defines the following function:

- Configure Window Trigger Source

This section describes the behavior and requirements of this function.

33.3.1 Configure Window Trigger Source

Description

Configures the window trigger Source, ThresholdLow, ThresholdHigh, and Condition. A window trigger occurs when a signal that enters or leaves a given voltage range is detected.

.NET Method Prototype

```
void Trigger.Sources[].Window.Configure (Double thresholdLow,
                                         Double thresholdHigh,
                                         WindowCondition condition);
```

COM Method Prototype

```
HRESULT Trigger.Sources.Item().Window.Configure ([in] double ThresholdLow,
                                [in] double ThresholdHigh,
                                [in] IviDigitizerWindowConditionEnum
Condition);
```

C Prototype

```
ViStatus IviDigitizer_ConfigureWindowTriggerSource (ViSession Vi,
                                                    ViConstString Source,
                                                    ViReal64 ThresholdLow,
                                                    ViReal64 ThresholdHigh,
                                                    ViInt32 Condition);
```

Parameters

Inputs	Description	Base Type
`Vi`	Instrument handle.	`ViSession`
`Source`	Specifies the trigger source that is to be configured.	`ViString`
`ThresholdLow`	Sets the window triggering low threshold in Volts. This value sets the Window Trigger Low Threshold attribute.	`ViReal64`
`ThresholdHigh`	Sets the window triggering high threshold in Volts. This value sets the Window Trigger High Threshold attribute.	`ViReal64`
`Condition`	Specifies whether a signal that is entering or leaving the voltage window defined by the high and low thresholds triggers the digitizer. This value sets the Window Trigger Condition attribute.	`ViInt32`

Defined Values for the Condition Parameter

Name	*Description*		
		Language	*Identifier*
Entering	Configures the digitizer to trigger on signals when they enter the given triggering window. The end-user specifies the high and low thresholds with the Window Trigger High Threshold and Window Trigger Low Threshold properties.		
		.NET	`WindowCondition.Entering`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`
		COM	`IviDigitizerWindowConditionEntering`

Leaving	Configures the digitizer to trigger on signals when they leave the given triggering window. The end-user specifies the high and low thresholds with the Width High Threshold and Width Low Threshold properties.		
		.NET	`WindowCondition.Leaving`
		C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`
		COM	`IviDigitizerWindowConditionLeaving`

Return Values (C/COM)

The *IVI-3.2: Inherent Capabilities Specification* defines general status codes that this function can return.

.NET Exceptions

The IVI-3.2: *Inherent Capabilities Specification* defines general exceptions that may be thrown, and warning events that may be raised, by this method.

33.4 IviDigitizerWindowTrigger Behavior Model

The IviDigitizerWindowTrigger extension group follows the same behavior model as the IviDigitizerBase capability group described in Section 4.4, *IviDigitizerBase Behavior Model*.

33.5 IviDigitizerWindowTrigger Compliance Notes

For a specific driver to comply with the IviDigitizerWindowTrigger extension, it shall be compliant with the IviDigitizerBase capability group and it shall implement all of the attributes and functions listed in this section.

34. IviDigitizer Attribute ID Definitions

The following table defines the ID value for all IviDigitizer class attributes.

Table 34-1. IviDigitizer Attributes ID Values

Attribute Name	ID Definition
Channel Count	`IVI_INHERENT_ATTR_BASE + 203`
Active Trigger Source	`IVI_CLASS_ATTR_BASE + 1`
Channel Enabled	`IVI_CLASS_ATTR_BASE + 2`
Input Connector Selection	`IVI_CLASS_ATTR_BASE + 3`
Input Impedance	`IVI_CLASS_ATTR_BASE + 4`
Is Idle	`IVI_CLASS_ATTR_BASE + 5`
Is Measuring	`IVI_CLASS_ATTR_BASE + 6`
Is Waiting For Arm	`IVI_CLASS_ATTR_BASE + 7`
Is Waiting For Trigger	`IVI_CLASS_ATTR_BASE + 8`
Max First Valid Point Value	`IVI_CLASS_ATTR_BASE + 9`
Max Samples Per Channel	`IVI_CLASS_ATTR_BASE + 10`
Min Record Size	`IVI_CLASS_ATTR_BASE + 11`
Num Acquired Records	`IVI_CLASS_ATTR_BASE + 12`
Num Records To Acquire	`IVI_CLASS_ATTR_BASE + 13`
Record Size	`IVI_CLASS_ATTR_BASE + 14`
Sample Rate	`IVI_CLASS_ATTR_BASE + 15`
Trigger Coupling	`IVI_CLASS_ATTR_BASE + 16`
Trigger Delay	`IVI_CLASS_ATTR_BASE + 17`
Trigger Hysteresis	`IVI_CLASS_ATTR_BASE + 18`
Trigger Level	`IVI_CLASS_ATTR_BASE + 19`
Trigger Output Enabled	`IVI_CLASS_ATTR_BASE + 20`
Trigger Slope	`IVI_CLASS_ATTR_BASE + 21`
Trigger Source Count	`IVI_CLASS_ATTR_BASE + 22`
Trigger Type	`IVI_CLASS_ATTR_BASE + 23`
Vertical Coupling	`IVI_CLASS_ATTR_BASE + 24`
Vertical Offset	`IVI_CLASS_ATTR_BASE + 25`
Vertical Range	`IVI_CLASS_ATTR_BASE + 26`
Board Temperature	`IVI_CLASS_ATTR_BASE + 100`
Temperature Units	`IVI_CLASS_ATTR_BASE + 101`
Input Filter Bypass	`IVI_CLASS_ATTR_BASE + 200`
Input Filter Max Frequency	`IVI_CLASS_ATTR_BASE + 201`
Input Filter Min Frequency	`IVI_CLASS_ATTR_BASE + 202`

Attribute Name	ID Definition
Channel Temperature	`IVI_CLASS_ATTR_BASE + 300`
Time Interleaved Channel List	`IVI_CLASS_ATTR_BASE + 400`
Time Interleaved Channel List Auto	`IVI_CLASS_ATTR_BASE + 401`
Data Interleaved Channel List	`IVI_CLASS_ATTR_BASE + 500`
Reference Oscillator External Frequency	`IVI_CLASS_ATTR_BASE + 600`
Reference Oscillator Output Enabled	`IVI_CLASS_ATTR_BASE + 601`
Reference Oscillator Source	`IVI_CLASS_ATTR_BASE + 602`
Sample Clock Divider	`IVI_CLASS_ATTR_BASE + 700`
Sample Clock External Frequency	`IVI_CLASS_ATTR_BASE + 701`
Sample Clock Output Enabled	`IVI_CLASS_ATTR_BASE + 702`
Sample Clock Source	`IVI_CLASS_ATTR_BASE + 703`
Sample Mode	`IVI_CLASS_ATTR_BASE + 800`
Downconversion Center Frequency	`IVI_CLASS_ATTR_BASE + 900`
Downconversion Enabled	`IVI_CLASS_ATTR_BASE + 901`
Downconversion IQ Interleaved	`IVI_CLASS_ATTR_BASE + 902`
Active Arm Source	`IVI_CLASS_ATTR_BASE + 1000`
Arm Count	`IVI_CLASS_ATTR_BASE + 1001`
Arm Coupling	`IVI_CLASS_ATTR_BASE + 1002`
Arm Delay	`IVI_CLASS_ATTR_BASE + 1003`
Arm Hysteresis	`IVI_CLASS_ATTR_BASE + 1004`
Arm Level	`IVI_CLASS_ATTR_BASE + 1005`
Arm Output Enabled	`IVI_CLASS_ATTR_BASE + 1006`
Arm Slope	`IVI_CLASS_ATTR_BASE + 1007`
Arm Source Count	`IVI_CLASS_ATTR_BASE + 1008`
Arm Type	`IVI_CLASS_ATTR_BASE + 1009`
Arm Source List	`IVI_CLASS_ATTR_BASE + 1100`
Arm Source Operator	`IVI_CLASS_ATTR_BASE + 1101`
Glitch Arm Condition	`IVI_CLASS_ATTR_BASE + 1200`
Glitch Arm Polarity	`IVI_CLASS_ATTR_BASE + 1201`

Attribute Name	ID Definition
Glitch Arm Width	`IVI_CLASS_ATTR_BASE + 1202`
Runt Arm High Threshold	`IVI_CLASS_ATTR_BASE + 1300`
Runt Arm Low Threshold	`IVI_CLASS_ATTR_BASE + 1301`
Runt Arm Polarity	`IVI_CLASS_ATTR_BASE + 1302`
TV Arm Event	`IVI_CLASS_ATTR_BASE + 1400`
TV Arm Line Number	`IVI_CLASS_ATTR_BASE + 1401`
TV Arm Polarity	`IVI_CLASS_ATTR_BASE + 1402`
TV Arm Signal Format	`IVI_CLASS_ATTR_BASE + 1403`
Width Arm Condition	`IVI_CLASS_ATTR_BASE + 1500`
Width Arm High Threshold	`IVI_CLASS_ATTR_BASE + 1501`
Width Arm Low Threshold	`IVI_CLASS_ATTR_BASE + 1502`
Width Arm Polarity	`IVI_CLASS_ATTR_BASE + 1503`
Window Arm Condition	`IVI_CLASS_ATTR_BASE + 1600`
Window Arm High Threshold	`IVI_CLASS_ATTR_BASE + 1601`
Window Arm Low Threshold	`IVI_CLASS_ATTR_BASE + 1602`
Trigger Modifier	`IVI_CLASS_ATTR_BASE + 1700`
Trigger Source List	`IVI_CLASS_ATTR_BASE + 1800`
Trigger Source Operator	`IVI_CLASS_ATTR_BASE + 1801`
Pretrigger Samples	`IVI_CLASS_ATTR_BASE + 1900`
Trigger Holdoff	`IVI_CLASS_ATTR_BASE + 2000`
Glitch Trigger Condition	`IVI_CLASS_ATTR_BASE + 2100`
Glitch Trigger Polarity	`IVI_CLASS_ATTR_BASE + 2101`
Glitch Trigger Width	`IVI_CLASS_ATTR_BASE + 2102`
Runt Trigger High Threshold	`IVI_CLASS_ATTR_BASE + 2200`
Runt Trigger Low Threshold	`IVI_CLASS_ATTR_BASE + 2201`
Runt Trigger Polarity	`IVI_CLASS_ATTR_BASE + 2202`
TV Trigger Event	`IVI_CLASS_ATTR_BASE + 2300`

Attribute Name	ID Definition
TV Trigger Line Number	`IVI_CLASS_ATTR_BASE + 2301`
TV Trigger Polarity	`IVI_CLASS_ATTR_BASE + 2302`
TV Trigger Signal Format	`IVI_CLASS_ATTR_BASE + 2303`
Width Trigger Condition	`IVI_CLASS_ATTR_BASE + 2400`
Width Trigger High Threshold	`IVI_CLASS_ATTR_BASE + 2401`
Width Trigger Low Threshold	`IVI_CLASS_ATTR_BASE + 2402`
Width Trigger Polarity	`IVI_CLASS_ATTR_BASE + 2403`
Window Trigger Condition	`IVI_CLASS_ATTR_BASE + 2500`
Window Trigger High Threshold	`IVI_CLASS_ATTR_BASE + 2501`
Window Trigger Low Threshold	`IVI_CLASS_ATTR_BASE + 2502`

35. IviDigitizer Attribute Value Definitions

This section specifies the actual value for each defined attribute value.

Arm Coupling

Value Name	*Language*	*Identifier*	*Actual Value*
AC	.NET	`TriggerCoupling.AC`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_AC`	0
	COM	`IviDigitizerTriggerCouplingAC`	0
DC	.NET	`TriggerCoupling.DC`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_DC`	1
	COM	`IviDigitizerTriggerCouplingDC`	1
HF Reject	.NET	`TriggerCoupling.HFReject`	2
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_HF_REJECT`	2
	COM	`IviDigitizerTriggerCouplingHFReject`	2
LF Reject	.NET	`TriggerCoupling.LFReject`	3
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_LF_REJECT`	3
	COM	`IviDigitizerTriggerCouplingLFReject`	3
Noise Reject	.NET	`TriggerCoupling.NoiseReject`	4
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_NOISE_REJECT`	4
	COM	`IviDigitizerTriggerCouplingNoiseReject`	4

Arm Slope

Value Name	*Language*	*Identifier*	*Actual Value*
Negative	.NET	`Slope.Negative`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`	0
	COM	`IviDigitizerTriggerSlopeNegative`	0
Positive	.NET	`Slope.Positive`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`	1
	COM	`IviDigitizerTriggerSlopePositive`	1

Arm Source Operator

Value Name	*Language*	*Identifier*	*Actual Value*
AND	.NET	`ArmSourceOperator.And`	0
	C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_AND`	0
	COM	`IviDigitizerArmSourceOperatorAND`	0
OR	.NET	`ArmSourceOperator.Or`	1

Value Name	*Language*	*Identifier*	*Actual Value*
	C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_OR`	1
	COM	`IviDigitizerArmSourceOperatorOR`	1
None	.NET	`ArmSourceOperator.None`	2
	C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_NONE`	2
	COM	`IviDigitizerArmSourceOperatorNone`	2

Arm Type

Value Name	*Language*	*Identifier*	*Actual Value*
Arm Edge	.NET	`ArmType.Edge`	0
	C	`IVIDIGITIZER_VAL_EDGE_ARM`	1
	COM	`IviDigitizerArmEdge`	1
Arm Width	.NET	`ArmType.Width`	1
	C	`IVIDIGITIZER_VAL_WIDTH_ARM`	2
	COM	`IviDigitizerArmWidth`	2
Arm Runt	.NET	`ArmType.Runt`	2
	C	`IVIDIGITIZER_VAL_RUNT_ARM`	3
	COM	`IviDigitizerArmRunt`	3
Arm Glitch	.NET	`ArmType.Glitch`	3
	C	`IVIDIGITIZER_VAL_GLITCH_ARM`	4
	COM	`IviDigitizerArmGlitch`	4
Arm TV	.NET	`ArmType.TV`	4
	C	`IVIDIGITIZER_VAL_TV_ARM`	5
	COM	`IviDigitizerArmTV`	5
Arm Window	.NET	`ArmType.Window`	5
	C	`IVIDIGITIZER_VAL_WINDOW_ARM`	6
	COM	`IviDigitizerArmWindow`	6

Glitch Arm Condition

Value Name	*Language*	*Identifier*	*Actual Value*
Less Than	.NET	`GlitchCondition.LessThan`	0
	C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`	1
	COM	`IviDigitizerGlitchConditionLessThan`	1
Greater Than	.NET	`GlitchCondition.GreaterThan`	1
	C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`	2
	COM	`IviDigitizerGlitchConditionGreaterThan`	2

Glitch Arm Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`GlitchPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`	1
	COM	`IviDigitizerGlitchPolarityPositive`	1
Negative	.NET	`GlitchPolarity.Negative`	1

Value Name	***Language***	***Identifier***	***Actual Value***
	C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`	2
	COM	`IviDigitizerGlitchPolarityNegative`	2
Either	.NET	`GlitchPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_GLITCH_EITHER`	3
	COM	`IviDigitizerGlitchPolarityEither`	3

Glitch Trigger Condition

Value Name	*Language*	*Identifier*	*Actual Value*
Less Than	.NET	`GlitchCondition.LessThan`	0
	C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`	1
	COM	`IviDigitizerGlitchConditionLessThan`	1
Greater Than	.NET	`GlitchCondition.GreaterThan`	1
	C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`	2
	COM	`IviDigitizerGlitchConditionGreaterThan`	2

Glitch Trigger Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`GlitchPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`	1
	COM	`IviDigitizerGlitchPolarityPositive`	1
Negative	.NET	`GlitchPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`	2
	COM	`IviDigitizerGlitchPolarityNegative`	2
Either	.NET	`GlitchPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_GLITCH_EITHER`	3
	COM	`IviDigitizerGlitchPolarityEither`	3

Is Idle, Is Measuring, Is Waiting For Arm, Is Waiting For Trigger

Value Name	*Language*	*Identifier*	*Actual Value*
True	.NET	`AcqusitionStatusResult.True`	0
	C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`	1
	COM	`IviDigitizerAcqusitionStatusResultTrue`	1
False	.NET	`AcqusitionStatusResult.False`	1
	C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`	2
	COM	`IviDigitizerAcqusitionStatusResultFalse`	2
Unknown	.NET	`AcqusitionStatusResult.Unknown`	2
	C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`	3
	COM	`IviDigitizerAcqusitionStatusResultUnknown`	3

Reference Oscillator Source

Value Name	*Language*	*Identifier*	*Actual Value*
Internal	.NET	`ReferenceOscillatorSource.Internal`	0
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_INTERNAL`	0
	COM	`IviDigitizerReferenceOscillatorSourceInternal`	0
External	.NET	`ReferenceOscillatorSource.External`	1
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_EXTERNAL`	1
	COM	`IviDigitizerReferenceOscillatorSourceExternal`	1
PXIClk10	.NET	`ReferenceOscillatorSource.PxiClk10`	2
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXI_CLK10`	2
	COM	`IviDigitizerReferenceOscillatorSourcePXIClk10`	2
PXIeClk100	.NET	`ReferenceOscillatorSource.PxiExpressClk100`	3
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXIE_CLK100`	3
	COM	`IviDigitizerReferenceOscillatorSourcePXIeClk100`	3

Runt Arm Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`RuntPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`	1
	COM	`IviDigitizerRuntPolarityPositive`	1
Negative	.NET	`RuntPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`	2
	COM	`IviDigitizerRuntPolarityNegative`	2
Either	.NET	`RuntPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_RUNT_EITHER`	3
	COM	`IviDigitizerRuntPolarityEither`	3

Runt Trigger Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`RuntPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`	1
	COM	`IviDigitizerRuntPolarityPositive`	1
Negative	.NET	`RuntPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`	2
	COM	`IviDigitizerRuntPolarityNegative`	2

Value Name	*Language*	*Identifier*	*Actual Value*
Either	.NET	`RuntPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_RUNT_EITHER`	3
	COM	`IviDigitizerRuntPolarityEither`	3

Sample Clock Source

Value Name	*Language*	*Identifier*	*Actual Value*
Internal	.NET	`SampleClockSource.Internal`	0
	C	`IVIDIGITIZER_VAL_SAMPLE_CLOCK_SOURCE_INTERNAL`	0
	COM	`IviDigitizerSampleClockSourceInternal`	0
External	.NET	`SampleClockSource.External`	1
	C	`IVIDIGITIZER_VAL_SAMPLE_CLOCK_SOURCE_EXTERNAL`	1
	COM	`IviDigitizerSampleClockSourceExternal`	1

Sample Mode

Value Name	*Language*	*Identifier*	*Actual Value*
Real Time	.NET	`SampleMode.RealTime`	0
	C	`IVIDIGITIZER_VAL_SAMPLE_MODE_REAL_TIME`	0
	COM	`IviDigitizerSampleModeRealTime`	0
Equivalent Time	.NET	`SampleMode.EquivalentTime`	1
	C	`IVIDIGITIZER_VAL_SAMPLE_MODE_EQUIVALENT_TIME`	1
	COM	`IviDigitizerSampleModeEquivalentTime`	1

Temperature Units

Value Name	*Language*	*Identifier*	*Actual Value*
Celsius	.NET	`TemperatureUnits.Celsius`	0
	C	`IVIDIGITIZER_VAL_CELSIUS`	0
	COM	`IviDigitizerTemperatureUnitsCelsius`	0
Fahrenheit	.NET	`TemperatureUnits.Fahrenheit`	1
	C	`IVIDIGITIZER_VAL_FAHRENHEIT`	1
	COM	`IviDigitizerTemperatureUnitsFahrenheit`	1
Kelvin	.NET	`TemperatureUnits.Kelvin`	2
	C	`IVIDIGITIZER_VAL_KELVIN`	2
	COM	`IviDigitizerTemperatureUnitsKelvin`	2

Trigger Coupling

Value Name	*Language*	*Identifier*	*Actual Value*
AC	.NET	`TriggerCoupling.AC`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_AC`	0
	COM	`IviDigitizerTriggerCouplingAC`	0
DC	.NET	`TriggerCoupling.DC`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_DC`	1
	COM	`IviDigitizerTriggerCouplingDC`	1
HF Reject	.NET	`TriggerCoupling.HFReject`	2
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_HF_REJECT`	2
	COM	`IviDigitizerTriggerCouplingHFReject`	2
LF Reject	.NET	`TriggerCoupling.LFReject`	3
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_LF_REJECT`	3
	COM	`IviDigitizerTriggerCouplingLFReject`	3
Noise Reject	.NET	`TriggerCoupling.NoiseReject`	4
	C	`IVIDIGITIZER_VAL_TRIGGER_COUPLING_NOISE_REJECT`	4
	COM	`IviDigitizerTriggerCouplingNoiseReject`	4

Trigger Modifier

Value Name	*Language*	*Identifier*	*Actual Value*
No Trigger Modifier	.NET	`TriggerModifier.None`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_NONE`	1
	COM	`IvidigitizerTriggerModifierNone`	1
Auto	.NET	`TriggerModifier.Auto`	1

	C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO`	2
	COM	`IviDigitizerTriggerModifierAuto`	2
Auto Level	.NET	`TriggerModifier.AutoLevel`	2
	C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO_LEVEL`	3
	COM	`IviDigitizerTriggerModifierAutoLevel`	3
Trigger Modifier Class Ext Base	C	`IVIDIGITIZER_VAL_TRIGGER_MOD_ CLASS_EXT_BASE`	100
Trigger Modifier Specific Ext Base	C	`IVIDIGITIZER_VAL_TRIGGER_MOD_ SPECIFIC_EXT_BASE`	1000

Trigger Slope

Value Name	*Language*	*Identifier*	*Actual Value*
Negative	.NET	`Slope.Negative`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`	0
	COM	`IviDigitizerTriggerSlopeNegative`	0
Positive	.NET	`Slope.Positive`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`	1
	COM	`IviDigitizerTriggerSlopePositive`	1

Trigger Source Operator

Value Name	*Language*	*Identifier*	*Actual Value*
AND	.NET	`TriggerSourceOperator.And`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_AND`	0
	COM	`IviDigitizerTriggerSourceOperatorAND`	0
OR	.NET	`TriggerSourceOperator.Or`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_OR`	1
	COM	`IviDigitizerTriggerSourceOperatorOR`	1
None	.NET	`TriggerSourceOperator.None`	2
	C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_NONE`	2
	COM	`IviDigitizerTriggerSourceOperatorNone`	2

Trigger Type

Value Name	*Language*	*Identifier*	*Actual Value*
Trigger Edge	.NET	`Trigger.Edge`	0
	C	`IVIDIGITIZER_VAL_EDGE_TRIGGER`	1
	COM	`IviDigitizerTriggerEdge`	1
Trigger Width	.NET	`Trigger.Width`	1
	C	`IVIDIGITIZER_VAL_WIDTH_TRIGGER`	2

Value Name	*Language*	*Identifier*	*Actual Value*
	COM	`IviDigitizerTriggerWidth`	2
Trigger Runt	.NET	`Trigger.Runt`	2
	C	`IVIDIGITIZER_VAL_RUNT_TRIGGER`	3
	COM	`IviDigitizerTriggerRunt`	3
Trigger Glitch	.NET	`Trigger.Glitch`	3
	C	`IVIDIGITIZER_VAL_GLITCH_TRIGGER`	4
	COM	`IviDigitizerTriggerGlitch`	4
TriggerTV	.NET	`Trigger.TV`	4
	C	`IVIDIGITIZER_VAL_TV_TRIGGER`	5
	COM	`IviDigitizerTriggerTV`	5
Trigger Window	.NET	`Trigger.Window`	5
	C	`IVIDIGITIZER_VAL_WINDOW_TRIGGER`	6
	COM	`IviDigitizerTriggerWindow`	6

TV Arm Event

Value Name	*Language*	*Identifier*	*Actual Value*
Field1	.NET	`TVTriggerEvent.Field1`	0
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`	1
	COM	`IviDigitizerTVTriggerEventField1`	1
Field2	.NET	`TVTriggerEvent.Field2`	1
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`	2
	COM	`IviDigitizerTVTriggerEventField2`	2
Any Field	.NET	`TVTriggerEvent.AnyField`	2
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`	3
	COM	`IviDigitizerTVTriggerEventAnyField`	3
Any Line	.NET	`TVTriggerEvent.AnyLine`	3
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`	4
	COM	`IviDigitizerTVTriggerEventAnyLine`	4
Line Number	.NET	`TVTriggerEvent.LineNumber`	4
	C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`	5
	COM	`IviDigitizerTVTriggerEventLineNumber`	5

TV Arm Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`TVTriggerPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_TV_POSITIVE`	1
	COM	`IviDigitizerTVTriggerPolarityPositive`	1
Negative	.NET	`TVTriggerPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_TV_NEGATIVE`	2
	COM	`IviDigitizerTVTriggerPolarityNegative`	2

TV Arm Signal Format

Value Name	*Language*	*Identifier*	*Actual Value*
NTSC	.NET	`TVSignalFormat.Ntsc`	0
	C	`IVIDIGITIZER_VAL_NTSC`	1
	COM	`IviDigitizerTVSignalFormatNTSC`	1
PAL	.NET	`TVSignalFormat.Pal`	1
	C	`IVIDIGITIZER_VAL_PAL`	2
	COM	`IviDigitizerTVSignalFormatPAL`	2
SECAM	.NET	`TVSignalFormat.Secam`	2

Value Name	*Language*	*Identifier*	*Actual Value*
	C	`IVIDIGITIZER_VAL_SECAM`	3
	COM	`IviDigitizerTVSignalFormatSECAM`	3

TV Trigger Event

Value Name	*Language*	*Identifier*	*Actual Value*
Field1	.NET	`TVTriggerEvent.Field1`	0
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`	1
	COM	`IviDigitizerTVTriggerEventField1`	1
Field2	.NET	`TVTriggerEvent.Field2`	1
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`	2
	COM	`IviDigitizerTVTriggerEventField2`	2
Any Field	.NET	`TVTriggerEvent.AnyField`	2
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`	3
	COM	`IviDigitizerTVTriggerEventAnyField`	3
Any Line	.NET	`TVTriggerEvent.AnyLine`	3
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`	4
	COM	`IviDigitizerTVTriggerEventAnyLine`	4
Line Number	.NET	`TVTriggerEvent.LineNumber`	4
	C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`	5
	COM	`IviDigitizerTVTriggerEventLineNumber`	5

TV Trigger Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`TVTriggerPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_TV_POSITIVE`	1
	COM	`IviDigitizerTVTriggerPolarityPositive`	1
Negative	.NET	`TVTriggerPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_TV_NEGATIVE`	2
	COM	`IviDigitizerTVTriggerPolarityNegative`	2

TV Trigger Signal Format

Value Name	*Language*	*Identifier*	*Actual Value*
NTSC	.NET	`TVSignalFormat.Ntsc`	0
	C	`IVIDIGITIZER_VAL_NTSC`	1
	COM	`IviDigitizerTVSignalFormatNTSC`	1
PAL	.NET	`TVSignalFormat.Pal`	1
	C	`IVIDIGITIZER_VAL_PAL`	2
	COM	`IviDigitizerTVSignalFormatPAL`	2
SECAM	.NET	`TVSignalFormat.Secam`	2

Value Name	*Language*	*Identifier*	*Actual Value*
	C	`IVIDIGITIZER_VAL_SECAM`	3
	COM	`IviDigitizerTVSignalFormatSECAM`	3

Vertical Coupling

Value Name	*Language*	*Identifier*	*Actual Value*
AC	.NET	`VerticalCoupling.AC`	0
	C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_AC`	0
	COM	`IviDigitizerVerticalCouplingAC`	0
DC	.NET	`VerticalCoupling.DC`	1
	C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_DC`	1
	COM	`IviDigitizerVerticalCouplingDC`	1
Gnd	.NET	`VerticalCoupling.Ground`	2
	C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_GND`	2
	COM	`IviDigitizerVerticalCouplingGnd`	2

Width Arm Condition

Value Name	*Language*	*Identifier*	*Actual Value*
Within	.NET	`WidthCondition.Within`	0
	C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`	1
	COM	`IviDigitizerWidthConditionWithin`	1
Outside	.NET	`WidthCondition.Outside`	1
	C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`	2
	COM	`IviDigitizerWidthConditionOutside`	2

Width Arm Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`WidthPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`	1
	COM	`IviDigitizerWidthPolarityPositive`	1
Negative	.NET	`WidthPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`	2
	COM	`IviDigitizerWidthPolarityNegative`	2
Either	.NET	`WidthPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_WIDTH_EITHER`	3
	COM	`IviDigitizerWidthPolarityEither`	3

Width Trigger Condition

Value Name	*Language*	*Identifier*	*Actual Value*
Within	.NET	`WidthCondition.Within`	0
	C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`	1
	COM	`IviDigitizerWidthConditionWithin`	1
Outside	.NET	`WidthCondition.Outside`	1
	C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`	2
	COM	`IviDigitizerWidthConditionOutside`	2

Width Trigger Polarity

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`WidthPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`	1
	COM	`IviDigitizerWidthPolarityPositive`	1
Negative	.NET	`WidthPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`	2
	COM	`IviDigitizerWidthPolarityNegative`	2
Either	.NET	`WidthPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_WIDTH_EITHER`	3
	COM	`IviDigitizerWidthPolarityEither`	3

Window Arm Condition

Value Name	*Language*	*Identifier*	*Actual Value*
Entering	.NET	`WindowCondition.Entering`	0
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`	1
	COM	`IviDigitizerWindowConditionEntering`	1
Leaving	.NET	`WindowCondition.Leaving`	1
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`	2
	COM	`IviDigitizerWindowConditionLeaving`	2

Window Trigger Condition

Value Name	*Language*	*Identifier*	*Actual Value*
Entering	.NET	`WindowCondition.Entering`	0
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`	1
	COM	`IviDigitizerWindowConditionEntering`	1
Leaving	.NET	`WindowCondition.Leaving`	1

Value Name	*Language*	*Identifier*	*Actual Value*
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`	2
	COM	`IviDigitizerWindowConditionLeaving`	2

36. IviDigitizer Function Parameter Value Definitions

This section specifies the actual values for each function parameter that defines values.

Configure Channel

Parameter: `Coupling`

Value Name	*Language*	*Identifier*	*Actual Value*
AC	.NET	`VerticalCoupling.AC`	0
	C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_AC`	0
	COM	`IviDigitizerVerticalCouplingAC`	0
DC	.NET	`VerticalCoupling.DC`	1
	C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_DC`	1
	COM	`IviDigitizerVerticalCouplingDC`	1
Gnd	.NET	`VerticalCoupling.Ground`	2
	C	`IVIDIGITIZER_VAL_VERTICAL_COUPLING_GND`	2
	COM	`IviDigitizerVerticalCouplingGnd`	2

Configure Edge Arm Source

Parameter: `Slope`

Value Name	*Language*	*Identifier*	*Actual Value*
Negative	.NET	`Slope.Negative`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`	0
	COM	`IviDigitizerTriggerSlopeNegative`	0
Positive	.NET	`Slope.Positive`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`	1
	COM	`IviDigitizerTriggerSlopePositive`	1

Configure Edge Trigger Source

Parameter: `Slope`

Value Name	*Language*	*Identifier*	*Actual Value*
Negative	.NET	`Slope.Negative`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_NEGATIVE`	0
	COM	`IviDigitizerTriggerSlopeNegative`	0
Positive	.NET	`Slope.Positive`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_SLOPE_POSITIVE`	1

Parameter: `Slope`

Value Name	*Language*	*Identifier*	*Actual Value*
	COM	`IviDigitizerTriggerSlopePositive`	1

Configure Glitch Arm Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`GlitchPolarity.Positive`	
	C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`	1
	COM	`IviDigitizerGlitchPolarityPositive`	1
Negative	.NET	`GlitchPolarity.Negative`	
	C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`	2
	COM	`IviDigitizerGlitchPolarityNegative`	2
Either	.NET	`GlitchPolarity.Either`	
	C	`IVIDIGITIZER_VAL_GLITCH_EITHER`	3
	COM	`IviDigitizerGlitchPolarityEither`	3

Configure Glitch Arm Source

Parameter: `Condition`

Value Name	*Language*	*Identifier*	*Actual Value*
Less Than	.NET	`GlitchCondition.LessThan`	0
	C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`	1
	COM	`IviDigitizerGlitchConditionLessThan`	1
Greater Than	.NET	`GlitchCondition.GreaterThan`	1
	C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`	2
	COM	`IviDigitizerGlitchConditionGreaterThan`	2

Configure Glitch Trigger Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`GlitchPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_GLITCH_POSITIVE`	1
	COM	`IviDigitizerGlitchPolarityPositive`	1

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Negative	.NET	`GlitchPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_GLITCH_NEGATIVE`	2
	COM	`IviDigitizerGlitchPolarityNegative`	2
Either	.NET	`GlitchPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_GLITCH_EITHER`	3
	COM	`IviDigitizerGlitchPolarityEither`	3

Configure Glitch Trigger Source

Parameter: `Condition`

Value Name	*Language*	*Identifier*	*Actual Value*
Less Than	.NET	`GlitchCondition.LessThan`	0
	C	`IVIDIGITIZER_VAL_GLITCH_LESS_THAN`	1
	COM	`IviDigitizerGlitchConditionLessThan`	1
Greater Than	.NET	`GlitchCondition.GreaterThan`	1
	C	`IVIDIGITIZER_VAL_GLITCH_GREATER_THAN`	2
	COM	`IviDigitizerGlitchConditionGreaterThan`	2

Configure Multi Arm

Parameter: `Operator`

Value Name	*Language*	*Identifier*	*Actual Value*
AND	.NET	`ArmSourceOperator.And`	0
	C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_AND`	0
	COM	`IviDigitizerArmSourceOperatorAND`	0
OR	.NET	`ArmSourceOperator.Or`	1
	C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_OR`	1
	COM	`IviDigitizerArmSourceOperatorOR`	1
None	.NET	`ArmSourceOperator.None`	2
	C	`IVIDIGITIZER_VAL_ARM_SOURCE_OPERATOR_NONE`	2
	COM	`IviDigitizerArmSourceOperatorNone`	2

Configure Multi Trigger

Parameter: `Operator`

Value Name	*Language*	*Identifier*	*Actual Value*
AND	.NET	`TriggerSourceOperator.And`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_AND`	0
	COM	`IviDigitizerTriggerSourceOperatorAND`	0
OR	.NET	`TriggerSourceOperator.Or`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_OR`	1
	COM	`IviDigitizerTriggerSourceOperatorOR`	1
None	.NET	`TriggerSourceOperator.None`	2
	C	`IVIDIGITIZER_VAL_TRIGGER_SOURCE_OPERATOR_NONE`	2

Parameter: `Operator`

Value Name	*Language*	*Identifier*	*Actual Value*
	COM	`IviDigitizerTriggerSourceOperatorNone`	2

Configure Reference Oscillator

Parameter: `Source`

Value Name	*Language*	*Identifier*	*Actual Value*
Internal	.NET	`ReferenceOscillatorSource.Internal`	0
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_INTERNAL`	0
	COM	`IviDigitizerReferenceOscillatorSourceInternal`	0
External	.NET	`ReferenceOscillatorSource.External`	1
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_EXTERNAL`	1
	COM	`IviDigitizerReferenceOscillatorSourceExternal`	1
PXIClk10	.NET	`ReferenceOscillatorSource.PxiClk10`	2
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXI_CLK10`	2
	COM	`IviDigitizerReferenceOscillatorSourcePXIClk10`	2
PXIeClk100	.NET	`ReferenceOscillatorSource.PxiExpressClk100`	3
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_PXIE_CLK100`	3
	COM	`IviDigitizerReferenceOscillatorSourcePXIeClk100`	3

Configure Runt Arm Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`RuntPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`	1
	COM	`IviDigitizerRuntPolarityPositive`	1
Negative	.NET	`RuntPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`	2
	COM	`IviDigitizerRuntPolarityNegative`	2
Either	.NET	`RuntPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_RUNT_EITHER`	3
	COM	`IviDigitizerRuntPolarityEither`	3

Configure Runt Trigger Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`RuntPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_RUNT_POSITIVE`	1
	COM	`IviDigitizerRuntPolarityPositive`	1
Negative	.NET	`RuntPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_RUNT_NEGATIVE`	2
	COM	`IviDigitizerRuntPolarityNegative`	2
Either	.NET	`RuntPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_RUNT_EITHER`	3
	COM	`IviDigitizerRuntPolarityEither`	3

Configure Sample Clock

Parameter: `Source`

Value Name	*Language*	*Identifier*	*Actual Value*
Internal	.NET	`ReferenceOscillatorSource.Internal`	0
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_INTERNAL`	0
	COM	`IviDigitizerReferenceOscillatorSourceInternal`	0
External	.NET	`ReferenceOscillatorSource.External`	1
	C	`IVIDIGITIZER_VAL_REFERENCE_OSCILLATOR_SOURCE_EXTERNAL`	1
	COM	`IviDigitizerReferenceOscillatorSourceExternal`	1

Configure Sample Mode

Parameter: `SampleMode`

Value Name	*Language*	*Identifier*	*Actual Value*
Real Time	.NET	`SampleMode.RealTime`	0
	C	`IVIDIGITIZER_VAL_SAMPLE_MODE_REAL_TIME`	0
	COM	`IviDigitizerSampleModeRealTime`	0
Equivalent Time	.NET	`SampleMode.EquivalentTime`	1
	C	`IVIDIGITIZER_VAL_SAMPLE_MODE_EQUIVALENT_TIME`	1
	COM	`IviDigitizerSampleModeEquivalentTime`	1

Configure Temperature Units

Parameter: `Units`

Value Name	*Language*	*Identifier*	*Actual Value*
Celsius	.NET	`TemperatureUnits.Celsius`	0
	C	`IVIDIGITIZER_VAL_CELSIUS`	0
	COM	`IviDigitizerTemperatureUnitsCelsius`	0
Fahrenheit	.NET	`TemperatureUnits.Fahrenheit`	1
	C	`IVIDIGITIZER_VAL_FAHRENHEIT`	1
	COM	`IviDigitizerTemperatureUnitsFahrenheit`	1
Kelvin	.NET	`TemperatureUnits.Kelvin`	2
	C	`IVIDIGITIZER_VAL_KELVIN`	2
	COM	`IviDigitizerTemperatureUnitsKelvin`	2

Configure Trigger Modifier

Parameter: `TriggerModifier`

Value Name	*Language*	*Identifier*	*Actual Value*
No Trigger Modifier	.NET	`TriggerModifier.None`	0
	C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_NONE`	1
	COM	`IvidigitizerTriggerModifierNone`	1
Auto	.NET	`TriggerModifier.Auto`	1
	C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO`	2
	COM	`IviDigitizerTriggerModifierAuto`	2
Auto Level	.NET	`TriggerModifier.AutoLevel`	2
	C	`IVIDIGITIZER_VAL_TRIGGER_MODIFIER_AUTO_LEVEL`	3
	COM	`IviDigitizerTriggerModifierAutoLevel`	3
Trigger Modifier Class Ext Base	C	`IVIDIGITIZER_VAL_TRIGGER_MOD_CLASS_EXT_BASE`	100
Trigger Modifier Specific Ext Base	C	`IVIDIGITIZER_VAL_TRIGGER_MOD_SPECIFIC_EXT_BASE`	1000

Configure TV Arm Source

Parameter: `SignalFormat`

Value Name	*Language*	*Identifier*	*Actual Value*
NTSC	.NET	`TVSignalFormat.Ntsc`	0
	C	`IVIDIGITIZER_VAL_NTSC`	1
	COM	`IviDigitizerTVSignalFormatNTSC`	1
PAL	.NET	`TVSignalFormat.Pal`	1
	C	`IVIDIGITIZER_VAL_PAL`	2
	COM	`IviDigitizerTVSignalFormatPAL`	2
SECAM	.NET	`TVSignalFormat.Secam`	2
	C	`IVIDIGITIZER_VAL_SECAM`	3
	COM	`IviDigitizerTVSignalFormatSECAM`	3

Configure TV Arm Source

Parameter: `Event`

Value Name	*Language*	*Identifier*	*Actual Value*
Field1	.NET	`TVTriggerEvent.Field1`	0
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`	1
	COM	`IviDigitizerTVTriggerEventField1`	1
Field2	.NET	`TVTriggerEvent.Field2`	1
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`	2
	COM	`IviDigitizerTVTriggerEventField2`	2
Any Field	.NET	`TVTriggerEvent.AnyField`	2
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`	3
	COM	`IviDigitizerTVTriggerEventAnyField`	3
Any Line	.NET	`TVTriggerEvent.AnyLine`	3
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`	4
	COM	`IviDigitizerTVTriggerEventAnyLine`	4
Line Number	.NET	`TVTriggerEvent.LineNumber`	4
	C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`	5
	COM	`IviDigitizerTVTriggerEventLineNumber`	5

Configure TV Arm Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`TVTriggerPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_TV_POSITIVE`	1
	COM	`IviDigitizerTVTriggerPolarityPositive`	1
Negative	.NET	`TVTriggerPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_TV_NEGATIVE`	2
	COM	`IviDigitizerTVTriggerPolarityNegative`	2

Configure TV Trigger Source

Parameter: `SignalFormat`

Value Name	*Language*	*Identifier*	*Actual Value*
NTSC	.NET	`TVSignalFormat.Ntsc`	0
	C	`IVIDIGITIZER_VAL_NTSC`	1
	COM	`IviDigitizerTVSignalFormatNTSC`	1
PAL	.NET	`TVSignalFormat.Pal`	1
	C	`IVIDIGITIZER_VAL_PAL`	2
	COM	`IviDigitizerTVSignalFormatPAL`	2
SECAM	.NET	`TVSignalFormat.Secam`	2
	C	`IVIDIGITIZER_VAL_SECAM`	3
	COM	`IviDigitizerTVSignalFormatSECAM`	3

Configure TV Trigger Source

Parameter: `Event`

Value Name	*Language*	*Identifier*	*Actual Value*
Field1	.NET	`TVTriggerEvent.Field1`	0
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD1`	1
	COM	`IviDigitizerTVTriggerEventField1`	1
Field2	.NET	`TVTriggerEvent.Field2`	1
	C	`IVIDIGITIZER_VAL_TV_EVENT_FIELD2`	2
	COM	`IviDigitizerTVTriggerEventField2`	2
Any Field	.NET	`TVTriggerEvent.AnyField`	2
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_FIELD`	3

Parameter: `Event`

Value Name	*Language*	*Identifier*	*Actual Value*
	COM	`IviDigitizerTVTriggerEventAnyField`	3
Any Line	.NET	`TVTriggerEvent.AnyLine`	3
	C	`IVIDIGITIZER_VAL_TV_EVENT_ANY_LINE`	4
	COM	`IviDigitizerTVTriggerEventAnyLine`	4
Line Number	.NET	`TVTriggerEvent.LineNumber`	4
	C	`IVIDIGITIZER_VAL_TV_EVENT_LINE_NUMBER`	5
	COM	`IviDigitizerTVTriggerEventLineNumber`	5

Configure TV Trigger Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`TVTriggerPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_TV_POSITIVE`	1
	COM	`IviDigitizerTVTriggerPolarityPositive`	1
Negative	.NET	`TVTriggerPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_TV_NEGATIVE`	2
	COM	`IviDigitizerTVTriggerPolarityNegative`	2

Configure Width Arm Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`WidthPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`	1
	COM	`IviDigitizerWidthPolarityPositive`	1
Negative	.NET	`WidthPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`	2
	COM	`IviDigitizerWidthPolarityNegative`	2
Either	.NET	`WidthPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_WIDTH_EITHER`	3
	COM	`IviDigitizerWidthPolarityEither`	3

Configure Width Arm Source

Parameter: `Condition`

Value Name	*Language*	*Identifier*	*Actual Value*
Within	.NET	`WidthCondition.Within`	0
	C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`	1
	COM	`IviDigitizerWidthConditionWithin`	1
Outside	.NET	`WidthCondition.Outside`	1
	C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`	2
	COM	`IviDigitizerWidthConditionOutside`	2

Configure Width Trigger Source

Parameter: `Polarity`

Value Name	*Language*	*Identifier*	*Actual Value*
Positive	.NET	`WidthPolarity.Positive`	0
	C	`IVIDIGITIZER_VAL_WIDTH_POSITIVE`	1
	COM	`IviDigitizerWidthPolarityPositive`	1
Negative	.NET	`WidthPolarity.Negative`	1
	C	`IVIDIGITIZER_VAL_WIDTH_NEGATIVE`	2
	COM	`IviDigitizerWidthPolarityNegative`	2
Either	.NET	`WidthPolarity.Either`	2
	C	`IVIDIGITIZER_VAL_WIDTH_EITHER`	3
	COM	`IviDigitizerWidthPolarityEither`	3

Configure Width Trigger Source

Parameter: `Condition`

Value Name	*Language*	*Identifier*	*Actual Value*
Within	.NET	`WidthCondition.Within`	`0`
	C	`IVIDIGITIZER_VAL_WIDTH_WITHIN`	`1`
	COM	`IviDigitizerWidthConditionWithin`	`1`
Outside	.NET	`WidthCondition.Outside`	`1`
	C	`IVIDIGITIZER_VAL_WIDTH_OUTSIDE`	`2`
	COM	`IviDigitizerWidthConditionOutside`	`2`

Configure Window Arm Source

Parameter: `Condition`

Value Name	*Language*	*Identifier*	*Actual Value*
Entering	.NET	`WindowCondition.Entering`	`0`
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`	`1`
	COM	`IviDigitizerWindowConditionEntering`	`1`
Leaving	.NET	`WindowCondition.Leaving`	`1`
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`	`2`
	COM	`IviDigitizerWindowConditionLeaving`	`2`

Configure Window Trigger Source

Parameter: `Condition`

Value Name	*Language*	*Identifier*	*Actual Value*
Entering	.NET	`WindowConditionEntering`	`0`
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_ENTERING`	`1`
	COM	`IviDigitizerWindowConditionEntering`	`1`
Leaving	.NET	`WindowConditionLeaving`	`1`
	C	`IVIDIGITIZER_VAL_WINDOW_CONDITION_LEAVING`	`2`
	COM	`IviDigitizerWindowConditionLeaving`	`2`

Is Idle, Is Measuring, Is Waiting For Arm, Is Waiting For Trigger (C Only)

Parameter: `Status`

Value Name	*Language*	*Identifier*	*Actual Value*
True	C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_TRUE`	1
False	C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_FALSE`	2
Unknown	C	`IVIDIGITIZER_VAL_ACQUISITION_STATUS_RESULT_UNKNOWN`	3

Read Waveform Int16, Read Waveform Int32, Read Waveform Int8, Read Waveform Real64

Parameter: `MaxTimeMilliseconds (C/COM), maximumTime (.NET)`

Value Name	*Language*	*Identifier*	*Actual Value*
Immediate	.NET	`PrecisionTimeSpan.Zero`	N/A
	C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`	0
	COM	`IviDigitizerTimeOutImmediate`	0
Infinite	.NET	`PrecisionTimeSpan.MaxValue`	N/A
	C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`	-1
	COM	`IviDigitizerTimeOutInfinite`	-1

Wait For Acquisition Complete

Parameter: `MaxTimeMilliseconds (C/COM), maximumTime (.NET)`

Value Name	*Language*	*Identifier*	*Actual Value*
Immediate	.NET	`PrecisionTimeSpan.Zero`	N/A
	C	`IVIDIGITIZER_VAL_TIMEOUT_IMMEDIATE`	0
	COM	`IviDigitizerTimeOutImmediate`	0
Infinite	.NET	`PrecisionTimeSpan.MaxValue`	N/A
	C	`IVIDIGITIZER_VAL_TIMEOUT_INFINITE`	-1
	COM	`IviDigitizerTimeOutInfinite`	-1

37. IviDigitizer Error and Completion Code Value Definitions

The table below specifies the actual value for each status code that the IviDigitizer class specification defines.

Table 37-1. IviDigitizer Error and Completion Codes

Error Name	***Description***			
		Language	***Identifier***	***Value(hex)***
Channel Not Enabled	Specified channel is not enabled.			
		.NET	`ChannelNotEnabledException`	
		C	`ERROR_CHANNEL_NOT_ENABLED`	`0xBFFA2001`
		COM	`E_IVIDIGITIZER_CHANNEL_NOT_ENABLED`	`0x80042001`
Max Time Exceeded	Maximum time exceeded before the operation completed.			
		.NET	`Ivi.Driver.MaxTimeExceededException`	`IVI Defined Exception (See IVI-3.2)`
		C	`ERROR_MAX_TIME_EXCEEDED`	`0xBFFA2002`
		COM	`E_IVIDIGITIZER_MAX_TIME_EXCEEDED`	`0x80042002`
Arm Not Software	Arm source is not set to software arm.			
		.NET	`ArmNotSoftwareException`	
		C	`ERROR_ARM_NOT_SOFTWARE`	`0xBFFA2003`
		COM	`E_IVIDIGITIZER_ARM_NOT_SOFTWARE`	`0x80042003`
Trigger Not Software	Trigger source is not set to software trigger.			
		.NET	`Ivi.Driver.TriggerNotSoftwareException`	`IVI Defined Exception (See IVI-3.2)`
		C	`ERROR_TRIGGER_NOT_SOFTWARE`	`0xBFFA1001`
		COM	`E_IVIDIGITIZER_TRIGGER_NOT_SOFTWARE`	`0x80041001`
Incompatible Fetch	The multi-record acquisition fetch functions must be used if the number of records to acquire is greater than 1.			
		.NET	`IncompatibleFetchException`	
		C	`ERROR_INCOMPATIBLE_FETCH`	`0xBFFA2004`
		COM	`E_IVIDIGITIZER_INCOMPATIBLE_FETCH`	`0x80042004`

Table 37-2. IviDigitizer Error Message Strings defines the recommended format of the message string associated with the errors. In C, these strings are returned by the Get Error function. In COM, these strings are the description contained in the ErrorInfo object.

Note: In the description string table entries listed below, **{0}** is always used to represent the component name.

Table 37-2. IviDigitizer Error Message Strings

Name	**Message String**
Channel Not Enabled	"**{0}**: Channel **{1}** is not enabled."
Max Time Exceeded	"**{0}**: Maximum time exceeded before the operation completed."

Name	Message String
Arm Not Software	"**{0}**: Arm source is not set to software arm."
Trigger Not Software	"**{0}**: Trigger source is not set to software trigger."
Incompatible Fetch	"**{0}**: Records to acquire >1, multi-record acquisition fetch must be used."

37.1 IVI.NET IviDigitizer Exceptions and Warnings

This section defines the list of IVI.NET exceptions and warnings that are specific to the IviLxiSync class. For general information on IVI.NET exceptions and warnings, refer to *IVI-3.1: Driver Architecture Specification* and section 12, *Common IVI.NET Exceptions and Warnings*, of *IVI-3.2: Inherent Capabilities Specification*.

The IVI.NET exceptions defined in this specification are declared in the Ivi.LxiSync namespace.

- ArmNotSoftwareException
- ChannelNotEnabledException
- IncompatibleFetchException

37.1.1 ArmNotSoftwareException

Description

A Send Arm Trigger method could not send an arm trigger.

Exception

```
Ivi.Digitizer.ArmNotSoftwareException(String message,
                                      String armSource)
```

Constructors

```
Ivi.Digitizer.ArmNotSoftwareException();

Ivi.Digitizer.ArmNotSoftwareException(String message);

Ivi.Digitizer.ArmNotSoftwareException(String message,
                                      System.Exception innerException);
```

Default Message String

```
The arm source is not set to software arm.
Actual arm source: <armSource>
```

Parameters

Inputs	Description	Base Type
`armSource`	The actual arm source.	`String`

Usage

This exception should only be thrown by the SendArmTrigger() method.

If driver developers specify the message string, they are responsible for message string localization.

37.1.2 ChannelNotEnabledException

Description

This exception is used when the driver finds that a channel is not enabled for measurement.

Constructors

```
Ivi.Digitizer.ChannelNotEnabledException(String message
                                         String channelName);

Ivi.Digitizer.ChannelNotEnabledException();

Ivi.Digitizer.ChannelNotEnabledException(String message);

Ivi.Digitizer.ChannelNotEnabledException(String message,
                                         System.Exception innerException);
```

Message String

```
The channel is not enabled for measurement.
Channel name: <channelName>
```

Parameters

Inputs	Description	Base Type
`ChannelName`	The name of the channel that is not enabled.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

37.1.3 IncompatibleFetchException

Description

This exception is thrown when the user attempts to fetch a single record when the digitizer is set to acquire multiple records.

Constructors

```
Ivi.Digitizer.IncompatibleFetchException(String alarmName,
                                         String recordsToAquire);

Ivi.Digitizer.IncompatibleFetchException();

Ivi.Digitizer.IncompatibleFetchException(String message);

Ivi.Digitizer.IncompatibleFetchException(String message,
                                         System.Exception innerException);
```

Message String

```
Records to acquire >1, multi-record acquisition fetch must be used.
Records to aquire: <recordsToAquire>.
```

Parameters

Inputs	Description	Base Type
`recordsToAquire`	The number of records that the digitizer is set to acquire.	`String`

Usage

If driver developers use constructors that take a message string, they are responsible for message string localization.

38. IviDigitizer Hierarchies

38.1 IviDigitizer .NET Hierarchy

The full IviDigitizer .NET Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.1, *.NET Inherent Capabilities* of *IVI-3.2: Inherent Capabilities Specification*. To avoid redundancy, it is omitted here.

Table 38-1. IviDigitizer .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
`Acquisition`		
`Abort`	Abort	M
`ConfigureAcquisition`	Configure Acquisition Record	M
`Initiate`	Initiate Acquisition	M
`QueryMinWaveformMemory`	Query Minimum Waveform Memory	M
`WaitForAcquisitionComplete`	Wait For Acquisition Complete	M
`MaxFirstValidPointValue`	Max First Valid Point Value	P
`MaxSamplesPerChannel`	Max Samples Per Channel	P
`MinRecordSize`	Min Record Size	P
`NumberOfAcquiredRecords`	Num Acquired Records	P
`NumberOfRecordsToAcquire`	Num Records To Acquire	P
`RecordSize`	Record Size	P
`SampleMode`	Sample Mode	P
`SampleRate`	Sample Rate	P
`TimeInterleavedChannelListAuto`	Time Interleaved Channel List Auto	P
`Status`		
`IsIdle`	Is Idle	P
`IsMeasuring`	Is Measuring	P
`IsWaitingForArm`	Is Waiting For Arm	P
`IsWaitingForTrigger`	Is Waiting For Trigger	P
`Arm`		
`SendSoftwareArm`	Send Software Arm	M
`ActiveSource`	Active Arm Source	P
`Count`	Arm Count	P
`Delay`	Arm Delay	P
`OutputEnabled`	Arm Output Enabled	P
`MultiArm`		
`Configure`	Configure Multi Arm	M
`SourceList`	Arm Source List	P
`SourceOperator`	Arm Source Operator	P
`Sources`		
`Count`	Arm Source Count	P

Table 38-1. IviDigitizer .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
`Name`	Arm Source Name	P
`Sources[]`		
`Coupling`	Arm Source Coupling	P
`Hysteresis`	Arm Source Hysteresis	P
`Level`	Arm Source Level	P
`Type`	Arm Source Type	P
`Edge`		
`Configure`	Configure Edge Arm Source	M
`Slope`	Arm Slope	P
`Glitch`		
`Configure`	Configure Glitch Arm Source	M
`Condition`	Glitch Arm Condition	P
`Polarity`	Glitch Arm Polarity	P
`Width`	Glitch Arm Width	P
`Runt`		
`Configure`	Configure Runt Arm Source	M
`Polarity`	Runt Arm Polarity	P
`ThresholdHigh`	Runt Arm High Threshold	P
`ThresholdLow`	Runt Arm Low Threshold	P
`TV`		
`Configure`	ConfigureTV Arm Source	M
`TriggerEvent`	TV Arm Event	P
`LineNumber`	TV Arm Line Number	P
`Polarity`	TV Arm Polarity	P
`SignalFormat`	TV Arm Signal Format	P
`Width`		
`Configure`	Configure Width Arm Source	M
`Condition`	Width Arm Condition	P
`Polarity`	Width Arm Polarity	P
`ThresholdHigh`	Width Arm High Threshold	P
`ThresholdLow`	Width Arm Lowthreshold	P
`Window`		
`Configure`	Configure Window Arm Source	M
`Condition`	Window Arm Condition	P
`ThresholdHigh`	Window Arm High Threshold	P
`ThresholdLow`	Window Arm Low Threshold	P
`Calibration`		
`SelfCalibrate`	Self Calibrate	M

Table 38-1. IviDigitizer .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
`Channels`		
`Count`	Channel Count	P
`Name`	Channel Name	P
`Channels[]`		
`Configure`	Configure Channel	M
`Coupling`	Vertical Coupling	P
`DataInterleavedChannelList`	Data Interleaved Channel List	P
`Enabled`	Channel Enabled	P
`InputConnectorSelection`	Input Connector Selection	P
`InputImpedance`	Input Impedance	P
`Offset`	Vertical Offset	P
`Range`	Vertical Range	P
`Temperature`	Channel Temperature	P
`TimeInterleavedChannelList`	Time Interleaved Channel List	P
`Downconversion`		
`Configure`	Configure Downconversion	M
`CenterFrequency`	Downconversion Center Frequency	P
`Enabled`	Downconversion Enabled	P
`IQInterleaved`	Downconversion IQ Interleaved	P
`Filter`		
`Configure`	Configure Input Filter	M
`Bypass`	Input Filter Bypass	P
`MaxFrequency`	Input Filter Max Frequency	P
`MinFrequency`	Input Filter Min Frequency	P
`Measurement`		
`FetchWaveform`	Fetch Waveform Int16 Fetch Waveform Int32 Fetch Waveform Int8 Fetch Waveform Real64	M
`ReadWaveform`	Read Waveform Int16 Read Waveform Int32 Read Waveform Int8 Read Waveform Real64	M
`MultiRecordMeasurement`		
`FetchMultiRecordWaveform`	Fetch Multi-Record Waveform Int16 Fetch Multi-Record Waveform Int32 Fetch Multi-Record Waveform Int8 Fetch Multi-Record Waveform Real64	M

Table 38-1. IviDigitizer .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
`ReferenceOscillator`		
`Configure`	Configure Reference Oscillator	M
`ExternalFrequency`	Reference Oscillator External Frequency	P
`OutputEnabled`	Reference Oscillator Output Enabled	P
`Source`	Reference Oscillator Source	P
`SampleClock`		
`Configure`	Configure Sample Clock	M
`ExternalDivider`	Sample Clock Divider	P
`ExternalFrequency`	Sample Clock External Frequency	P
`OutputEnabled`	Sample Clock Output Enabled	P
`Source`	Sample Clock Source	P
`Temperature`		
`BoardTemperature`	Board Temperature	P
`Units`	Temperature Units	P
`Trigger`		
`SendSoftwareTrigger`	Send Software Trigger	M
`ActiveSource`	Active Trigger Source	P
`Delay`	Trigger Delay	P
`Holdoff`	Trigger Holdoff	P
`Modifier`	Trigger Modifier	P
`OutputEnabled`	Trigger Output Enabled	P
`PretriggerSamples`	Pretrigger Samples	P
`MultiTrigger`		
`Configure`	Configure Multi Trigger	M
`SourceList`	Trigger Source List	P
`SourceOperator`	Trigger Source Operator	P
`Sources`		
`Count`	Trigger Source Count	P
`Name`	Trigger Source Name	P
`Sources[]`		
`Coupling`	Trigger Source Coupling	P
`Hysteresis`	Trigger Source Hysteresis	P
`Level`	Trigger Source Level	P
`Type`	Trigger Source Type	P
`Edge`		
`Configure`	Configure Edge Trigger Source	M
`Slope`	Trigger Slope	P
`Glitch`		

Table 38-1. IviDigitizer .NET Hierarchy

.NET Interface Hierarchy	Generic Name	Type
`Configure`	Configure Glitch Trigger Source	M
`Condition`	Glitch Trigger Condition	P
`Polarity`	Glitch Trigger Polarity	P
`Width`	Glitch Trigger Width	P
`Runt`		
`Configure`	Configure Runt Trigger Source	M
`Polarity`	Runt Trigger Polarity	P
`ThresholdHigh`	Runt Trigger High Threshold	P
`ThresholdLow`	Runt Trigger Low Threshold	P
`TV`		
`Configure`	ConfigureTV Trigger Source	M
`Event`	TV Trigger Event	P
`LineNumber`	TV Trigger Line Number	P
`Polarity`	TV Trigger Polarity	P
`SignalFormat`	TV Trigger Signal Format	P
`Width`		
`Configure`	Configure Width Trigger Source	M
`Condition`	Width Trigger Condition	P
`Polarity`	Width Trigger Polarity	P
`ThresholdHigh`	Width Trigger High Threshold	P
`ThresholdLow`	Width Trigger Low Threshold	P
`Window`		
`Configure`	Configure Window Trigger Source	M
`Condition`	Window Trigger Condition	P
`ThresholdHigh`	Window Trigger High Threshold	P
`ThresholdLow`	Window Trigger Low Threshold	P

38.1.1 IviDigitizer .NET Interfaces

In addition to implementing IVI inherent capabilities interfaces, IviDigitizer interfaces contain interface reference properties for accessing the following IviDigitizer interfaces:

- IIviDigitizerAcquisition
- IIviDigitizerAcquisitionStatus
- IIviDigitizerArm
- IIviDigitizerCalibration
- IIviDigitizerChannels
- IIviDigitizerReferenceOscillator
- IIviDigitizerSampleClock
- IIviDigitizerTemperature
- IIviDigitizerTrigger

The IIviDigitizerArm interface contains interface reference properties for accessing the following additional IviDigitizer arm interface(s):

- IIviDigitizerArmSources
- IIviDigitizerMultiArm

The IIviDigitizerArmSources interface contains methods and properties for accessing a collection of objects that implement the IIviDigitizerArmSource interface.

The IIviDigitizerArmSource interface contains interface reference properties for accessing the following additional IviDigitizer arm source interface(s):

- IIviDigitizerArmEdge
- IIviDigitizerArmGlitch
- IIviDigitizerArmRunt
- IIviDigitizerArmTV
- IIviDigitizerArmWidth
- IIviDigitizerArmWindow

The IIviDigitizerChannels interface contains methods and properties for accessing a collection of objects that implement the IIviDigitizerChannel interface.

The IIviDigitizerChannel interface contains interface reference properties for accessing the following additional IviDigitizer channel interface(s):

- IIviDigitizerChannelDownconversion
- IIviDigitizerChannelFilter
- IIviDigitizerChannelMeasurement
- IIviDigitizerChannelMultiRecordMeasurement

The IIviDigitizerTrigger interface contains interface reference properties for accessing the following additional IviDigitizer trigger interface(s):

- IIviDigitizerMultiTrigger
- IIviDigitizerTriggerSources

The IIviDigitizerTriggerSources interface contains methods and properties for accessing a collection of objects that implement the IIviDigitizerTriggerSource interface.

The IIviDigitizerTriggerSource interface contains interface reference properties for accessing the following additional IviDigitizer trigger source interface(s):

- IIviDigitizerTriggerEdge
- IIviDigitizerTriggerGlitch
- IIviDigitizerTriggerRunt
- IIviDigitizerTriggerTV
- IIviDigitizerTriggerWidth
- IIviDigitizerTriggerWindow

38.1.2 .NET Interface Reference Properties

Interface reference properties are used to navigate the IviDigitizer .NET hierarchy. This section describes the interface reference properties that the IviDigitizer interfaces define. All interface reference properties are read-only.

Table 38-2. IviDigitizer .NET Interface Reference Properties

Interface	Interface Reference Property
IIviDigitizerAcquisition	Acquisition
IIviDigitizerAcquisitionStatus	Acquisition.Status
IIviDigitizerArm	Arm
IIviDigitizerArmEdge	ArmSources[].Edge
IIviDigitizerArmGlitch	ArmSources[].Glitch
IIviDigitizerArmRunt	ArmSources[].Runt
IIviDigitizerArmSource	ArmSources[]
IIviDigitizerArmSourceCollection	ArmSources
IIviDigitizerArmTV	ArmSources[].TV
IIviDigitizerArmWidth	ArmSources[].Width
IIviDigitizerArmWindow	ArmSources[].Window
IIviDigitizerCalibration	Calibration
IIviDigitizerChannel	Channels[]
IIviDigitizerChannelDownconversion	Channels[].Downconversion
IIviDigitizerChannelFilter	Channels[].Filter
IIviDigitizerChannelMeasurement	Channels[].Measurement
IIviDigitizerChannelMultiRecordMeasurement	Channels[].MultiRecordMeasurement
IIviDigitizerChannelCollection	Channels
IIviDigitizerMultiArm	Arm.MultiArm
IIviDigitizerMultiTrigger	Trigger.MultiTrigger
IIviDigitizerReferenceOscillator	ReferenceOscillator
IIviDigitizerSampleClock	SampleClock
IIviDigitizerTemperature	Temperature
IIviDigitizerTrigger	Trigger
IIviDigitizerTriggerEdge	Trigger.Sources[].Edge
IIviDigitizerTriggerGlitch	Trigger.Sources[].Glitch
IIviDigitizerTriggerRunt	Trigger.Sources[].Runt
IIviDigitizerTriggerSource	Trigger.Sources[]
IIviDigitizerTriggerSourceCollection	Trigger.Sources
IIviDigitizerTriggerTV	Trigger.Sources[].TV
IIviDigitizerTriggerWidth	Trigger.Sources[].Width
IIviDigitizerTriggerWindow	Trigger.Sources[].Window

38.2 IviDigitizer COM Hierarchy

The full IviDigitizer COM Hierarchy includes the Inherent Capabilities Hierarchy as defined in Section 4.2, *COM Inherent Capabilities* of *IVI-3.2: Inherent Capabilities Specification*. To avoid redundancy, it is omitted here.

Table 38-3. IviDigitizer COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
Acquisition		
Abort	Abort	M
ConfigureAcquisition	Configure Acquisition Record	M
Initiate	Initiate Acquisition	M
QueryMinWaveformMemory	Query Minimum Waveform Memory	M
WaitForAcquisitionComplete	Wait For Acquisition Complete	M
MaxFirstValidPointValue	Max First Valid Point Value	P
MaxSamplesPerChannel	Max Samples Per Channel	P
MinRecordSize	Min Record Size	P
NumAcquiredRecords	Num Acquired Records	P
NumRecordsToAcquire	Num Records To Acquire	P
RecordSize	Record Size	P
SampleMode	Sample Mode	P
SampleRate	Sample Rate	P
TimeInterleavedChannelListAuto	Time Interleaved Channel List Auto	P
Status		
IsIdle	Is Idle	P
IsMeasuring	Is Measuring	P
IsWaitingForArm	Is Waiting For Arm	P
IsWaitingForTrigger	Is Waiting For Trigger	P
Arm		
SendSoftwareArm	Send Software Arm	M
ActiveSource	Active Arm Source	P
Count	Arm Count	P
Delay	Arm Delay	P
OutputEnabled	Arm Output Enabled	P
MultiArm		
Configure	Configure Multi Arm	M
SourceList	Arm Source List	P
SourceOperator	Arm Source Operator	P
Sources		
Count	Arm Source Count	P
Name	Arm Source Name	P
Item		

Table 38-3. IviDigitizer COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
Coupling	Arm Source Coupling	P
Hysteresis	Arm Source Hysteresis	P
Level	Arm Source Level	P
Type	Arm Source Type	P
Edge		
Configure	Configure Edge Arm Source	M
Slope	Arm Slope	P
Glitch		
Configure	Configure Glitch Arm Source	M
Condition	Glitch Arm Condition	P
Polarity	Glitch Arm Polarity	P
Width	Glitch Arm Width	P
Runt		
Configure	Configure Runt Arm Source	M
Polarity	Runt Arm Polarity	P
ThresholdHigh	Runt Arm High Threshold	P
ThresholdLow	Runt Arm Low Threshold	P
TV		
Configure	ConfigureTV Arm Source	M
Event	TV Arm Event	P
LineNumber	TV Arm Line Number	P
Polarity	TV Arm Polarity	P
SignalFormat	TV Arm Signal Format	P
Width		
Configure	Configure Width Arm Source	M
Condition	Width Arm Condition	P
Polarity	Width Arm Polarity	P
ThresholdHigh	Width Arm High Threshold	P
ThresholdLow	Width Arm Lowthreshold	P
Window		
Configure	Configure Window Arm Source	M
Condition	Window Arm Condition	P
ThresholdHigh	Window Arm High Threshold	P
ThresholdLow	Window Arm Low Threshold	P
Calibration		
SelfCalibrate	Self Calibrate	M
Channels		
Count	Channel Count	P

Table 38-3. IviDigitizer COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
`Name`	Channel Name	P
`Item`		
`Configure`	Configure Channel	M
`Coupling`	Vertical Coupling	P
`DataInterleavedChannelList`	Data Interleaved Channel List	P
`Enabled`	Channel Enabled	P
`InputConnectorSelection`	Input Connector Selection	P
`InputImpedance`	Input Impedance	P
`Offset`	Vertical Offset	P
`Range`	Vertical Range	P
`Temperature`	Channel Temperature	P
`TimeInterleavedChannelList`	Time Interleaved Channel List	P
`Downconversion`		
`Configure`	Configure Downconversion	M
`CenterFrequency`	Downconversion Center Frequency	P
`Enabled`	Downconversion Enabled	P
`IQInterleaved`	Downconversion IQ Interleaved	P
`Filter`		
`Configure`	Configure Input Filter	M
`Bypass`	Input Filter Bypass	P
`MaxFrequency`	Input Filter Max Frequency	P
`MinFrequency`	Input Filter Min Frequency	P
`Measurement`		
`FetchWaveformInt16`	Fetch Waveform Int16	M
`FetchWaveformInt32`	Fetch Waveform Int32	M
`FetchWaveformInt8`	Fetch Waveform Int8	M
`FetchWaveformReal64`	Fetch Waveform Real64	M
`ReadWaveformInt16`	Read Waveform Int16	M
`ReadWaveformInt32`	Read Waveform Int32	M
`ReadWaveformInt8`	Read Waveform Int8	M
`ReadWaveformReal64`	Read Waveform Real64	M
`MultiRecordMeasurement`		
`FetchMultiRecordWaveformInt16`	Fetch Multi-Record Waveform Int16	M
`FetchMultiRecordWaveformInt32`	Fetch Multi-Record Waveform Int32	M
`FetchMultiRecordWaveformInt8`	Fetch Multi-Record Waveform Int8	M
`FetchMultiRecordWaveformReal64`	Fetch Multi-Record Waveform Real64	M
`ReferenceOscillator`		
`Configure`	Configure Reference Oscillator	M

Table 38-3. IviDigitizer COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
ExternalFrequency	Reference Oscillator External Frequency	P
OutputEnabled	Reference Oscillator Output Enabled	P
Source	Reference Oscillator Source	P
SampleClock		
Configure	Configure Sample Clock	M
ExternalDivider	Sample Clock Divider	P
ExternalFrequency	Sample Clock External Frequency	P
OutputEnabled	Sample Clock Output Enabled	P
Source	Sample Clock Source	P
Temperature		
BoardTemperature	Board Temperature	P
Units	Temperature Units	P
Trigger		
SendSoftwareTrigger	Send Software Trigger	M
ActiveSource	Active Trigger Source	P
Delay	Trigger Delay	P
Holdoff	Trigger Holdoff	P
Modifier	Trigger Modifier	P
OutputEnabled	Trigger Output Enabled	P
PretriggerSamples	Pretrigger Samples	P
MultiTrigger		
Configure	Configure Multi Trigger	M
SourceList	Trigger Source List	P
SourceOperator	Trigger Source Operator	P
Sources		
Count	Trigger Source Count	P
Name	Trigger Source Name	P
Item		
Coupling	Trigger Source Coupling	P
Hysteresis	Trigger Source Hysteresis	P
Level	Trigger Source Level	P
Type	Trigger Source Type	P
Edge		
Configure	Configure Edge Trigger Source	M
Slope	Trigger Slope	P
Glitch		
Configure	Configure Glitch Trigger Source	M
Condition	Glitch Trigger Condition	P

Table 38-3. IviDigitizer COM Hierarchy

COM Interface Hierarchy	Generic Name	Type
`Polarity`	Glitch Trigger Polarity	P
`Width`	Glitch Trigger Width	P
`Runt`		
`Configure`	Configure Runt Trigger Source	M
`Polarity`	Runt Trigger Polarity	P
`ThresholdHigh`	Runt Trigger High Threshold	P
`ThresholdLow`	Runt Trigger Low Threshold	P
`TV`		
`Configure`	ConfigureTV Trigger Source	M
`Event`	TV Trigger Event	P
`LineNumber`	TV Trigger Line Number	P
`Polarity`	TV Trigger Polarity	P
`SignalFormat`	TV Trigger Signal Format	P
`Width`		
`Configure`	Configure Width Trigger Source	M
`Condition`	Width Trigger Condition	P
`Polarity`	Width Trigger Polarity	P
`ThresholdHigh`	Width Trigger High Threshold	P
`ThresholdLow`	Width Trigger Low Threshold	P
`Window`		
`Configure`	Configure Window Trigger Source	M
`Condition`	Window Trigger Condition	P
`ThresholdHigh`	Window Trigger High Threshold	P
`ThresholdLow`	Window Trigger Low Threshold	P

38.2.1 IviDigitizer COM Interfaces

In addition to implementing IVI inherent capabilities interfaces, IviDigitizer interfaces contain interface reference properties for accessing the following IviDigitizer interfaces:

- IIviDigitizerAcquisition
- IIviDigitizerAcquisitionStatus
- IIviDigitizerArm
- IIviDigitizerCalibration
- IIviDigitizerChannels
- IIviDigitizerReferenceOscillator
- IIviDigitizerSampleClock
- IIviDigitizerTemperature
- IIviDigitizerTrigger

The IIviDigitizerArm interface contains interface reference properties for accessing the following additional IviDigitizer arm interface(s):

- IIviDigitizerArmSources
- IIviDigitizerMultiArm

The IIviDigitizerArmSources interface contains methods and properties for accessing a collection of objects that implement the IIviDigitizerArmSource interface.

The IIviDigitizerArmSource interface contains interface reference properties for accessing the following additional IviDigitizer arm source interface(s):

- IIviDigitizerArmEdge
- IIviDigitizerArmGlitch
- IIviDigitizerArmRunt
- IIviDigitizerArmTV
- IIviDigitizerArmWidth
- IIviDigitizerArmWindow

The IIviDigitizerChannels interface contains methods and properties for accessing a collection of objects that implement the IIviDigitizerChannel interface.

The IIviDigitizerChannel interface contains interface reference properties for accessing the following additional IviDigitizer channel interface(s):

- IIviDigitizerChannelDownconversion
- IIviDigitizerChannelFilter
- IIviDigitizerChannelMeasurement
- IIviDigitizerChannelMultiRecordMeasurement

The IIviDigitizerTrigger interface contains interface reference properties for accessing the following additional IviDigitizer trigger interface(s):

- IIviDigitizerMultiTrigger
- IIviDigitizerTriggerSources

The IIviDigitizerTriggerSources interface contains methods and properties for accessing a collection of objects that implement the IIviDigitizerTriggerSource interface.

The IIviDigitizerTriggerSource interface contains interface reference properties for accessing the following additional IviDigitizer trigger source interface(s):

- IIviDigitizerTriggerEdge
- IIviDigitizerTriggerGlitch
- IIviDigitizerTriggerRunt
- IIviDigitizerTriggerTV
- IIviDigitizerTriggerWidth
- IIviDigitizerTriggerWindow

Table 38-4. IviDigitizer Interface GUIDs lists the interfaces that this specification defines and their GUIDs.

Table 38-4. IviDigitizer Interface GUIDs

Interface	GUID
`IIviDigitizer`	`{47ed540e-a398-11d4-ba58-000064657374}`
`IIviDigitizerAcquisition`	`{47ed540f-a398-11d4-ba58-000064657374}`
`IIviDigitizerAcquisitionStatus`	`{47ed542e-a398-11d4-ba58-000064657374}`
`IIviDigitizerArm`	`{47ed5410-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmEdge`	`{47ed5411-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmGlitch`	`{47ed5412-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmRunt`	`{47ed5413-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmSource`	`{47ed5414-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmSources`	`{47ed5415-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmTV`	`{47ed5416-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmWidth`	`{47ed5417-a398-11d4-ba58-000064657374}`
`IIviDigitizerArmWindow`	`{47ed5418-a398-11d4-ba58-000064657374}`
`IIviDigitizerCalibration`	`{47ed5419-a398-11d4-ba58-000064657374}`
`IIviDigitizerChannel`	`{47ed541a-a398-11d4-ba58-000064657374}`
`IIviDigitizerChannelDownconversion`	`{47ed542c-a398-11d4-ba58-000064657374}`
`IIviDigitizerChannelFilter`	`{47ed541b-a398-11d4-ba58-000064657374}`
`IIviDigitizerChannelMeasurement`	`{47ed541c-a398-11d4-ba58-000064657374}`
`IIviDigitizerChannelMultiRecordMeasurement`	`{47ed542d-a398-11d4-ba58-000064657374}`
`IIviDigitizerChannels`	`{47ed541d-a398-11d4-ba58-000064657374}`
`IIviDigitizerMultiArm`	`{47ed541e-a398-11d4-ba58-000064657374}`
`IIviDigitizerMultiTrigger`	`{47ed541f-a398-11d4-ba58-000064657374}`
`IIviDigitizerReferenceOscillator`	`{47ed5420-a398-11d4-ba58-000064657374}`
`IIviDigitizerSampleClock`	`{47ed5421-a398-11d4-ba58-000064657374}`
`IIviDigitizerTemperature`	`{47ed5422-a398-11d4-ba58-000064657374}`
`IIviDigitizerTrigger`	`{47ed5423-a398-11d4-ba58-000064657374}`
`IIviDigitizerTriggerEdge`	`{47ed5424-a398-11d4-ba58-000064657374}`
`IIviDigitizerTriggerGlitch`	`{47ed5425-a398-11d4-ba58-000064657374}`
`IIviDigitizerTriggerRunt`	`{47ed5426-a398-11d4-ba58-000064657374}`

Table 38-4. IviDigitizer Interface GUIDs

Interface	GUID
IIviDigitizerTriggerSource	{47ed5427-a398-11d4-ba58-000064657374}
IIviDigitizerTriggerSources	{47ed5428-a398-11d4-ba58-000064657374}
IIviDigitizerTriggerTV	{47ed5429-a398-11d4-ba58-000064657374}
IIviDigitizerTriggerWidth	{47ed542a-a398-11d4-ba58-000064657374}
IIviDigitizerTriggerWindow	{47ed542b-a398-11d4-ba58-000064657374}

38.2.1 COM Interface Reference Properties

Interface reference properties are used to navigate the IviDigitizer COM hierarchy. This section describes the interface reference properties that the IviDigitizer interfaces define. All interface reference properties are read-only.

Table 38-5. IviDigitizer COM Interface Reference Properties

Interface	Interface Reference Property
IIviDigitizerAcquisition	Acquisition
IIviDigitizerAcquisitionStatus	Acquisition.Status
IIviDigitizerArm	Arm
IIviDigitizerArmEdge	ArmSource.Edge
IIviDigitizerArmGlitch	ArmSource.Glitch
IIviDigitizerArmRunt	ArmSource.Runt
IIviDigitizerArmSource	ArmSource
IIviDigitizerArmSources	ArmSources
IIviDigitizerArmTV	ArmSource.TV
IIviDigitizerArmWidth	ArmSource.Width
IIviDigitizerArmWindow	ArmSource.Window
IIviDigitizerCalibration	Calibration
IIviDigitizerChannel	Channel
IIviDigitizerChannelDownconversion	Channel.Downconversion
IIviDigitizerChannelFilter	Channel.Filter
IIviDigitizerChannelMeasurement	Channel.Measurement
IIviDigitizerChannelMultiRecordMeasurement	Channel.MultiRecordMeasurement
IIviDigitizerChannels	Channels
IIviDigitizerMultiArm	Arm.MultiArm
IIviDigitizerMultiTrigger	Trigger.MultiTrigger
IIviDigitizerReferenceOscillator	ReferenceOscillator
IIviDigitizerSampleClock	SampleClock
IIviDigitizerTemperature	Temperature
IIviDigitizerTrigger	Trigger
IIviDigitizerTriggerEdge	Trigger.Source.Edge
IIviDigitizerTriggerGlitch	Trigger.Source.Glitch
IIviDigitizerTriggerRunt	Trigger.Source.Runt
IIviDigitizerTriggerSource	Trigger.Source

Table 38-5. IviDigitizer COM Interface Reference Properties

Interface	Interface Reference Property
`IIviDigitizerTriggerSources`	`Trigger.Sources`
`IIviDigitizerTriggerTV`	`Trigger.Source.TV`
`IIviDigitizerTriggerWidth`	`Trigger.Source.Width`
`IIviDigitizerTriggerWindow`	`Trigger.Source.Window`

38.2.2 IviDigitizer COM Category

The IviDigitizer class COM Category shall be "IviDigitizer", and the Category ID (CATID) shall be:
{ 47ed5160-a398-11d4-ba58-000064657374}.

38.3 IviDigitizer C Function Hierarchy

The IviDigitizer class function hierarchy is shown in the following table. The full IviDigitizer C Function Hierarchy includes the C Inherent Functions as defined in Section 4.3, *C Inherent Capabilities* of *IVI-3.2: Inherent Capabilities Specification*. To avoid redundancy, the Inherent Capabilities are omitted here.

Table 38-6. IviDigitizer C Function Hierarchy

Name or Class	Function Name
Calibration…	
Self Calibrate	`IviDigitizer_SelfCalibrate`
Configuration…	
Acquisition…	
Configure Acquisition Record	`IviDigitizer_ConfigureAcquisition`
Configure Sample Mode	`IviDigitizer_ConfigureSampleMode`
Downconversion…	
Configure Downconversion	`IviDigitizer_ConfigureDownconversion`
Arm…	
Configure Edge Arm Source	`IviDigitizer_ConfigureEdgeArmSource`
Configure Glitch Arm Source	`IviDigitizer_ConfigureGlitchArmSource`
Configure Multi Arm	`IviDigitizer_ConfigureMultiArm`
Configure Runt Arm Source	`IviDigitizer_ConfigureRuntArmSource`
Configure TV Arm Source	`IviDigitizer_ConfigureTVArmSource`
Configure Width Arm Source	`IviDigitizer_ConfigureWidthArmSource`
Configure Window Arm Source	`IviDigitizer_ConfigureWindowArmSource`
Get Arm Source Name	`IviDigitizer_GetArmSourceName`
Channel…	
Configure Channel	`IviDigitizer_ConfigureChannel`
Configure Input Filter	`IviDigitizer_ConfigureInputFilter`
Get Channel Name	`IviDigitizer_GetChannelName`
Interleaved Data…	
Configure Data Interleaved Channel List	`IviDigitizer_ ConfigureDataInterleavedChannelList`
Configure Time Interleaved Channel List	`IviDigitizer_ ConfigureTimeInterleavedChannelList`
Reference Oscillator…	
Configure Reference Oscillator	`IviDigitizer_ ConfigureReferenceOscillator`

Name or Class	Function Name
Configure Reference Oscillator Output Enabled	`IviDigitizer_ ConfigureReferenceOscillatorOutputEnabled`
Sample Clock…	
Configure Sample Clock	`IviDigitizer_ ConfigureSampleClock`
Configure Sample Clock Output Enabled	`IviDigitizer_ ConfigureSampleClockOutputEnabled`
Trigger…	
Configure Edge Trigger Source	`IviDigitizer_ConfigureEdgeTriggerSource`
Configure Glitch Trigger Source	`IviDigitizer_ConfigureGlitchTriggerSource`
Configure Multi Trigger	`IviDigitizer_ConfigureMultiTrigger`
Configure Pretrigger Samples	`IviDigitizer_ConfigurePretriggerSamples`
Configure Runt Trigger Source	`IviDigitizer_ConfigureRuntTriggerSource`
Configure Trigger Holdoff	`IviDigitizer_ConfigureTriggerHoldoff`
Configure Trigger Modifier	`IviDigitizer_ConfigureTriggerModifier`
Configure TV Trigger Source	`IviDigitizer_ConfigureTVTriggerSource`
Configure Width Trigger Source	`IviDigitizer_ConfigureWidthTriggerSource`
Configure Window Trigger Source	`IviDigitizer_ConfigureWindowTriggerSource`
Get Trigger Source Name	`IviDigitizer_GetTriggerSourceName`
Utility…	
Temperature…	
Configure Temperature Units	`IviDigitizer_ConfigureTemperatureUnits`
Query Board Temperature	`IviDigitizer_QueryBoardTemperature`
Query Channel Temperature	`IviDigitizer_QueryChannelTemperature`
Waveform Acquisition…	
Read Waveform Int16	`IviDigitizer_ReadWaveformInt16`
Read Waveform Int32	`IviDigitizer_ReadWaveformInt32`
Read Waveform Int8	`IviDigitizer_ReadWaveformInt8`
Read Waveform Real64	`IviDigitizer_ReadWaveformReal64`
Low-Level Acquisition…	
Abort	`IviDigitizer_Abort`
Fetch Waveform Int16	`IviDigitizer_FetchWaveformInt16`
Fetch Waveform Int32	`IviDigitizer_FetchWaveformInt32`
Fetch Waveform Int8	`IviDigitizer_FetchWaveformInt8`
Fetch Waveform Real64	`IviDigitizer_FetchWaveformReal64`
Initiate Acquisition	`IviDigitizer_InitiateAcquisition`
Is Idle	`IviDigitizer_IsIdle`
Is Measuring	`IviDigitizer_IsMeasuring`
Is Waiting For Arm	`IviDigitizer_IsWaitingForArm`

Name or Class	Function Name
Is Waiting For Trigger	`IviDigitizer_IsWaitingForTrigger`
Query Minimum Waveform Memory	`IviDigitizer_QueryMinWaveformMemory`
Send Software Arm	`IviDigitizer_SendSoftwareArm`
Send Software Trigger	`IviDigitizer_SendSoftwareTrigger`
Wait For Acquisition Complete	`IviDigitizer_WaitForAcquisitionComplete`
Multi-Record Acquisition…	
Fetch Multi-Record Waveform Int16	`IviDigitizer_FetchMultiRecordWaveformInt16`
Fetch Multi-Record Waveform Int32	`IviDigitizer_FetchMultiRecordWaveformInt32`
Fetch Multi-Record Waveform Int8	`IviDigitizer_FetchMultiRecordWaveformInt8`
Fetch Multi-Record Waveform Real64	`IviDigitizer_FetchMultiRecordWaveformReal64`

38.4 IviDigitizer C Attribute Hierarchy

The IviDigitizer class C attribute hierarchy is shown in the following table. The full IviDigitizer C Attribute Hierarchy includes the C Inherent Attributes as defined in Section 4.3, *C Inherent Capabilities* of *IVI-3.2: Inherent Capabilities Specification*. To avoid redundancy, the C Inherent Attributes are omitted here.

Table 38-7. IviDigitizer C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Arm	
Active Arm Source	`IVIDIGITIZER_ATTR_ACTIVE_ARM_SOURCE`
Arm Count	`IVIDIGITIZER_ATTR_ARM_COUNT`
Arm Source Count	`IVIDIGITIZER_ATTR_ARM_SOURCE_COUNT`
Arm Coupling	`IVIDIGITIZER_ATTR_ARM_COUPLING`
Arm Delay	`IVIDIGITIZER_ATTR_ARM_DELAY`
Arm Hysteresis	`IVIDIGITIZER_ATTR_ARM_HYSTERESIS`
Arm Level	`IVIDIGITIZER_ATTR_ARM_LEVEL`
Arm Output Enabled	`IVIDIGITIZER_ATTR_ARM_OUTPUT_ENABLED`
Arm Type	`IVIDIGITIZER_ATTR_ARM_TYPE`
Edge Arming	
Arm Slope	`IVIDIGITIZER_ATTR_ARM_SLOPE`
Glitch Arming	
Glitch Arm Condition	`IVIDIGITIZER_ATTR_GLITCH_ARM_CONDITION`
Glitch Arm Polarity	`IVIDIGITIZER_ATTR_GLITCH_ARM_POLARITY`
Glitch Arm Width	`IVIDIGITIZER_ATTR_GLITCH_ARM_WIDTH`
Multi Arm	
Arm Source List	`IVIDIGITIZER_ATTR_ARM_SOURCE_LIST`
Arm Source Operator	`IVIDIGITIZER_ATTR_ARM_SOURCE_OPERATOR`
Runt Arming	
Runt Arm High Threshold	`IVIDIGITIZER_ATTR_RUNT_ARM_HIGH_THRESHOLD`
Runt Arm Low Threshold	`IVIDIGITIZER_ATTR_RUNT_ARM_LOW_THRESHOLD`
Runt Arm Polarity	`IVIDIGITIZER_ATTR_RUNT_ARM_POLARITY`
TV Arming	
TV Arm Event	`IVIDIGITIZER_ATTR_TV_ARM_EVENT`
TV Arm Line Number	`IVIDIGITIZER_ATTR_TV_ARM_LINE_NUMBER`
TV Arm Polarity	`IVIDIGITIZER_ATTR_TV_ARM_POLARITY`
TV Arm Signal Format	`IVIDIGITIZER_ATTR_TV_ARM_SIGNAL_FORMAT`
Width Arming	
Width Arm Condition	`IVIDIGITIZER_ATTR_WIDTH_ARM_CONDITION`
Width Arm High Threshold	`IVIDIGITIZER_ATTR_WIDTH_ARM_HIGH_THRESHOLD`

Table 38-7. IviDigitizer C Attributes Hierarchy

Category or Generic Attribute Name	C Defined Constant
Width Arm Low Threshold	`IVIDIGITIZER_ATTR_WIDTH_ARM_LOWTHRESHOLD`
Width Arm Polarity	`IVIDIGITIZER_ATTR_WIDTH_ARM_POLARITY`
Window Arming	
Window Arm Condition	`IVIDIGITIZER_ATTR_WINDOW_ARM_CONDITION`
Window Arm High Threshold	`IVIDIGITIZER_ATTR_WINDOW_ARM_HIGH_THRESHOLD`
Window Arm Low Threshold	`IVIDIGITIZER_ATTR_WINDOW_ARM_LOW_THRESHOLD`
Channel	
Channel Count	`IVIDIGITIZER_ATTR_CHANNEL_COUNT`
Channel Enabled	`IVIDIGITIZER_ATTR_CHANNEL_ENABLED`
Data Interleaved Channel List	`IVIDIGITIZER_ATTR_DATA_INTERLEAVED_CHANNEL_LIST`
Downconversion	
Downconversion Enabled	`IVIDIGITIZER_ATTR_DOWNCONVERSION_ENABLED`
Downconversion Center Frequency	`IVIDIGITIZER_ATTR_DOWNCONVERSION_CENTER_FREQUENCY`
Downconversion IQ Interleaved	`IVIDIGITIZER_ATTR_DOWNCONVERSION_IQ_INTERLEAVED`
Filter	
Input Filter Bypass	`IVIDIGITIZER_ATTR_INPUT_FILTER_BYPASS`
Input Filter Max Frequency	`IVIDIGITIZER_ATTR_INPUT_FILTER_MAX_FREQUENCY`
Input Filter Min Frequency	`IVIDIGITIZER_ATTR_INPUT_FILTER_MIN_FREQUENCY`
Input Connector Selection	`IVIDIGITIZER_ATTR_INPUT_CONNECTOR_SELECTION`
Input Impedance	`IVIDIGITIZER_ATTR_INPUT_IMPEDANCE`
Time Interleaved Channel List	`IVIDIGITIZER_ATTR_TIME_INTERLEAVED_CHANNEL_LIST`
Vertical Coupling	`IVIDIGITIZER_ATTR_VERTICAL_COUPLING`
Vertical Offset	`IVIDIGITIZER_ATTR_VERTICAL_OFFSET`
Vertical Range	`IVIDIGITIZER_ATTR_VERTICAL_RANGE`
Reference Oscillator	
Reference Oscillator External Frequency	`IVIDIGITIZER_ATTR_REFERENCE_OSCILLATOR_EXTERNAL_FREQUENCY`
Reference Oscillator Output Enabled	`IVIDIGITIZER_ATTR_REFERENCE_OSCILLATOR_OUTPUT_ENABLED`
Reference Oscillator Source	`IVIDIGITIZER_ATTR_REFERENCE_OSCILLATOR_SOURCE`
Sample Clock	
Sample Clock External Divider	`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_EXTERNAL_DIVIDER`
Sample Clock External Frequency	`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_EXTERNAL_FREQUENCY`
Sample Clock Output Enabled	`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_OUTPUT_ENABLED`
Sample Clock Source	`IVIDIGITIZER_ATTR_SAMPLE_CLOCK_SOURCE`
Temperature	

Table 38-7. IviDigitizer C Attributes Hierarchy

Category or Generic Attribute Name	**C Defined Constant**
Board Temperature	`IVIDIGITIZER_ATTR_BOARD_TEMPERATURE`
Channel Temperature	`IVIDIGITIZER_ATTR_CHANNEL_TEMPERATURE`
Temperature Units	`IVIDIGITIZER_ATTR_TEMPERATURE_UNITS`
Trigger	
Active Trigger Source	`IVIDIGITIZER_ATTR_ACTIVE_TRIGGER_SOURCE`
Pretrigger Samples	`IVIDIGITIZER_ATTR_PRETRIGGER_SAMPLES`
Trigger Source Count	`IVIDIGITIZER_ATTR_TRIGGER_SOURCE_COUNT`
Trigger Coupling	`IVIDIGITIZER_ATTR_TRIGGER_COUPLING`
Trigger Modifier	`IVIDIGITIZER_ATTR_TRIGGER_MODIFIER`
Trigger Delay	`IVIDIGITIZER_ATTR_TRIGGER_DELAY`
Trigger Holdoff	`IVIDIGITIZER_ATTR_TRIGGER_HOLDOFF`
Trigger Hysteresis	`IVIDIGITIZER_ATTR_TRIGGER_HYSTERESIS`
Trigger Level	`IVIDIGITIZER_ATTR_TRIGGER_LEVEL`
Trigger Output Enabled	`IVIDIGITIZER_ATTR_TRIGGER_OUTPUT_ENABLED`
Trigger Type	`IVIDIGITIZER_ATTR_TRIGGER_TYPE`
Edge Triggering	
Trigger Slope	`IVIDIGITIZER_ATTR_TRIGGER_SLOPE`
Glitch Triggering	
Glitch Trigger Condition	`IVIDIGITIZER_ATTR_GLITCH_TRIGGER_CONDITION`
Glitch Trigger Polarity	`IVIDIGITIZER_ATTR_GLITCH_TRIGGER_POLARITY`
Glitch Trigger Width	`IVIDIGITIZER_ATTR_GLITCH_TRIGGER_WIDTH`
MultiTrigger	
Trigger Source List	`IVIDIGITIZER_ATTR_TRIGGER_SOURCE_LIST`
Trigger Source Operator	`IVIDIGITIZER_ATTR_TRIGGER_SOURCE_OPERATOR`
Runt Triggering	
Runt Trigger High Threshold	`IVIDIGITIZER_ATTR_RUNT_TRIGGER_HIGH_THRESHOLD`
Runt Trigger Low Threshold	`IVIDIGITIZER_ATTR_RUNT_TRIGGER_LOW_THRESHOLD`
Runt Trigger Polarity	`IVIDIGITIZER_ATTR_RUNT_TRIGGER_POLARITY`
TV Triggering	
TV Trigger Event	`IVIDIGITIZER_ATTR_TV_TRIGGER_EVENT`
TV Trigger Line Number	`IVIDIGITIZER_ATTR_TV_TRIGGER_LINE_NUMBER`
TV Trigger Polarity	`IVIDIGITIZER_ATTR_TV_TRIGGER_POLARITY`
TV Trigger Signal Format	`IVIDIGITIZER_ATTR_TV_TRIGGER_SIGNAL_FORMAT`
Width Triggering	
Width Trigger Condition	`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_CONDITION`

Table 38-7. IviDigitizer C Attributes Hierarchy

Category or Generic Attribute Name	**C Defined Constant**
Width Trigger High Threshold	`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_HIGH_THRESHOLD`
Width Trigger Low Threshold	`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_LOW_THRESHOLD`
Width Trigger Polarity	`IVIDIGITIZER_ATTR_WIDTH_TRIGGER_POLARITY`
Window Triggering	
Window Trigger Condition	`IVIDIGITIZER_ATTR_WINDOW_TRIGGER_CONDITION`
Window Trigger High Threshold	`IVIDIGITIZER_ATTR_WINDOW_TRIGGER_HIGH_THRESHOLD`
Window Trigger Low Threshold	`IVIDIGITIZER_ATTR_WINDOW_TRIGGER_LOW_THRESHOLD`
Waveform Acquisition	
Is Idle	`IVIDIGITIZER_ATTR_IS_IDLE`
Is Measuring	`IVIDIGITIZER_ATTR_IS_MEASURING`
Is Waiting For Arm	`IVIDIGITIZER_ATTR_IS_WAITING_FOR_ARM`
Is Waiting For Trigger	`IVIDIGITIZER_ATTR_IS_WAITING_FOR_TRIGGER`
Max First Valid Point Value	`IVIDIGITIZER_ATTR_MAX_FIRST_VALID_POINT_VAL`
Max Samples Per Channel	`IVIDIGITIZER_ATTR_MAX_SAMPLES_PER_CHANNEL`
Min Record Size	`IVIDIGITIZER_ATTR_MIN_RECORD_SIZE`
Num Acquired Records	`IVIDIGITIZER_ATTR_NUM_ACQUIRED_RECORDS`
Num Records To Acquire	`IVIDIGITIZER_ATTR_NUM_RECORDS_TO_ACQUIRE`
Record Size	`IVIDIGITIZER_ATTR_RECORD_SIZE`
Sample Mode	`IVIDIGITIZER_ATTR_SAMPLE_MODE`
Sample Rate	`IVIDIGITIZER_ATTR_SAMPLE_RATE`
Time Interleaved Channel List Auto	`IVIDIGITIZER_ATTR_TIME_INTERLEAVED_CHANNEL_LIST_AUTO`

Appendix A Specific Driver Development Guidelines

A.1 Introduction

This section describes situations driver developers should be aware of when developing a specific instrument driver that complies with the IviDigitizer class.

A.2 Disabling Unused Extension Groups

Specific drivers are required to disable extension capability groups that an application program does not explicitly use. The specific driver can do so by setting the attributes of an extension capability group to the values that this section recommends. A specific driver can set these values for all extension capability groups when the `*Prefix*_init`, `*Prefix*_InitWithOptions`, or `*Prefix*_reset` functions execute. This assumes that the extension capability groups remain disabled until the application program explicitly uses them. For the large majority of instruments, this assumption is true.

Under certain conditions, a specific driver might have to implement a more complex approach. For some instruments, configuring a capability group might affect instrument settings that correspond to an unused extension capability group. If these instrument settings affect the behavior of the instrument, then this might result in an interchangeability problem. If this can occur, the specific driver must take appropriate action so that the instrument settings that correspond to the unused extension capability group do not affect the behavior of the instrument when the application program performs an operation that might be affected by those settings.

The remainder of this section recommends attribute values that effectively disable each extension capability group.

Disabling the IviDigitizerArm Extension Group

To effectively disable the IviDigitizerArm extension group, set the Arm Source attribute to None or Immediate.

Disabling the IviDigitizerBoardTemperature Extension Group

The IviDigitizerBoardTemperature extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerBoardTemperature extension group.

Disabling the IviDigitizerChannelFilter Extension Group

Attribute values that effectively disable the IviDigitizerChannelFilter extension group are shown in the following table.

Table A-1. Values for Disabling the IviDigitizerChannelFilter Extension Group

Attribute	Value
Input Filter Bypass	True

Disabling the IviDigitizerChannelTemperature Extension Group

The IviDigitizerChannelTemperature extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerChannelTemperature extension group.

Disabling the IviDigitizerDataInterleavedChannels Extension Group

Attribute values that effectively disable the IviDigitizerDataInterleavedChannels extension group are shown in the following table.

Table A-2. Values for Disabling the IviDigitizerTimeInterleavedChannels Extension Group

Attribute	Value
Data Interleaved Channel List	Empty String

Disabling the IviDigitizerTimeInterleavedChannels Extension Group

Attribute values that effectively disable the IviDigitizerTimeInterleavedChannels extension group are shown in the following table.

Table A-3. Values for Disabling the IviDigitizerTimeInterleavedChannels Extension Group

Attribute	Value
Time Interleaved Channel List	Empty String

Disabling the IviDigitizerDownconversion Extension Group

Attribute values that effectively disable the IviDigitizerDownconversion extension group are shown in the following table.

Table A-4. Values for Disabling the IviDigitizerTimeInterleavedChannels Extension Group

Attribute	Value
Downconversion Enabled	False

Disabling the IviDigitizerGlitchArm Extension Group

The IviDigitizerGlitchArm extension group affects the instrument behavior only when the Arm Type attribute is set to the TV Arm value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerGlitchArm extension group.

Disabling the IviDigitizerGlitchTrigger Extension Group

The IviDigitizerGlitchTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to Glitch Trigger value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerGlitchTrigger extension group.

Disabling the IviDigitizerMultiArm Extension Group

Attribute values that effectively disable the IviDigitizerMultiArm extension group are shown in the following table.

Table A-5. Values for Disabling the IviDigitizerMultiArm Extension Group

Attribute	Value
Arm Source Operator	Arm Source Operator None

Disabling the IviDigitizerMultiTrigger Extension Group

Attribute values that effectively disable the IviDigitizerMultiTrigger extension group are shown in the following table.

Table A-6. Values for Disabling the IviDigitizerMultiTrigger Extension Group

Attribute	Value
Trigger Source Operator	Trigger Source Operator None

Disabling the IviDigitizerPretriggerSamples Extension Group

Attribute values that effectively disable the IviDigitizerMultiTrigger extension group are shown in the following table.

Table A-7. Values for Disabling the IviDigitizerPretriggerHoldff Extension Group

Attribute	Value
Pretrigger Samples	0

Disabling the IviDigitizerProbeAutoSense Extension Group

The IviDigitizerProbeAutoSense extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerProbeAutoSense extension group.

Disabling the IviDigitizerReferenceOscillator Extension Group

Attribute values that effectively disable the IviDigitizerReferenceOscillator extension group are shown in the following table.

Table A-8. Values for Disabling the IviDigitizerReferenceOscillator Extension Group

Attribute	Value
Reference Oscillator Source	Internal

Disabling the IviDigitizerRuntArm Extension Group

The IviDigitizerRuntArm extension group affects the instrument behavior only when the Arm Type attribute is set to the Runt Arm value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerRuntArm extension group.

Disabling the IviDigitizerRuntTrigger Extension Group

The IviDigitizerRuntTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to the Runt Trigger value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerRuntTrigger extension group.

Disabling the IviDigitizerSampleClock Extension Group

Attribute values that effectively disable the IviDigitizerSampleClock extension group are shown in the following table.

Table A-9. Values for Disabling the IviDigitizerSampleClock Extension Group

Attribute	Value
Sample Clock Source	Internal

Disabling the IviDigitizerSampleClockOutput Extension Group

The IviDigitizerSampleClockOutput extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerSampleClockOutput extension group.

Disabling the IviDigitizerSampleMode Extension Group

The IviDigitizerSampleMode extension group does not affect instrument behavior (although it does affect the measured data that will be returned in subsequent read or fetch calls). Therefore, this specification does not recommend attribute values that disable the IviDigitizerSampleMode extension group.

Disabling the IviDigitizerSelfCalibration Extension Group

The IviDigitizerSelfCalibration extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerSelfCalibration extension group.

Disabling the IviDigitizerSoftwareArm Extension Group

The IviDigitizerSoftwareArm extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerSoftwareArm extension group.

Disabling the IviDigitizerSoftwareTrigger Extension Group

The IviDigitizerSoftwareTrigger extension group does not affect instrument behavior. Therefore, this specification does not recommend attribute values that disable the IviDigitizerSoftwareTrigger extension group.

Disabling the IviDigitizerTriggerHoldoff Extension Group

Attribute values that effectively disable the IviDigitizerTriggerHoldoff extension group are shown in the following table.

Table A-10. Values for Disabling the IviDigitizerTriggerHoldoff Extension Group

Attribute	Value
Trigger Holdoff	0

Disabling the IviDigitizerTVArm Extension Group

The IviDigitizerTVArm extension group affects the instrument behavior only when the Arm Type attribute is set to the TV Arm value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerTVArm extension group.

Disabling the IviDigitizerTVTrigger Extension Group

The IviDigitizerTVTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to the TV Trigger value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerTVTrigger extension group.

Disabling the IviDigitizerWidthArm Extension Group

The IviDigitizerWidthArm extension group affects the instrument behavior only when the Arm Type attribute is set to the Width Arm value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerWidthArm extension group.

Disabling the IviDigitizerWidthTrigger Extension Group

The IviDigitizerWidthTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to the Width Trigger value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerWidthTrigger extension group.

Disabling the IviDigitizerWindowArm Extension Group

The IviDigitizerWindowArm extension group affects the instrument behavior only when the Arm Type attribute is set to the Window Arm value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerWindowArm extension group.

Disabling the IviDigitizerWindowTrigger Extension Group

The IviDigitizerWindowTrigger extension group affects the instrument behavior only when the Trigger Type attribute is set to the Window Trigger value. Therefore, this specification does not recommend attribute values that disable the IviDigitizerWindowTrigger extension group.

Disabling the IviDigitizerTriggerModifier Extension Group

Attribute value that effectively disables the IviDigitizerTriggerModifier extension group is shown in the following table.

Table A-11. Values for Disabling the IviDigitizerTriggerModifier Extension Group

Attribute	Value
Trigger Modifier	No Trigger Mod

A.3 Special Consideration for Query Instrument Status

Based on the value of Query Instrument Status, the IVI Class-Compliant specific driver may check the status of the instrument to see if it has encountered an error. In IVI Class-Compliant specific driver functions, the status check should not occur in the lowest-level measurement functions Initiate Acquisition, Abort, and Fetch Waveform XXX These functions are intended to give the application developer low-level control over signal generation. When calling these functions, the application developer is responsible for checking the status of the instrument. Checking status in every function at this level would also add unnecessary overhead to the specific instrument driver.

A.4 Implementing the Trigger Holdoff attribute

This specification defines the hold-off as the length of time the digitizer waits after it detects a trigger until it responds to additional triggers. Some digitizers specify the hold-off as starting from the end of the previous waveform acquisition instead of from the previous trigger.

These differences in how digitizers specify the hold-off setting can lead to non-interchangeable instruments behavior. Therefore if your instrument defines the hold-off as starting from the end of the previous waveform acquisition, you **must** translate that hold-off time to the one defined in this specification. To do that, you may do the following:

1. Translate the value the end-user specifies for hold-off to a value your digitizer expects when you implement the attribute. You can do this by subtracting the length of time from the Trigger Event to the end of the waveform record from the hold-off value the user specifies. Then send the resulting number to the digitizer. If the number is less than 0.0, use 0.0.

2. Perform the opposite translation when obtaining the value from the instrument.

3. Since it now depends on the acquisition settings, make sure that setting the Trigger Delay, Sample Rate, and Record Size attributes invalidates the Trigger Holdoff attribute.

Appendix B Interchangeability Checking Rules

B.1 Introduction

IVI drivers have a feature called interchangeability checking. Interchangeability checking returns a warning when it encounters a situation were the application program might not produce the same behavior when the user attempts to use a different instrument.

B.2 When to Perform Interchangeability Checking

Refer to Section 3.3.6*: Interchangeability Checking in IVI-3.1: Driver Architecture Specification* for a description of the rules for interchangeability checking in IVI drivers. The remainder of this section defines additional rules and exceptions for each capability group.

Interchangeability checking occurs when all of the following conditions are met:

- The Interchange Check attribute is set to True
- The user calls one of the following functions.
    - Initiate Acquisition
    - Read Waveform Int8
    - Read Waveform Int16
    - Read Waveform Int32
    - Read Waveform Real64

B.3 Interchangeability Checking Rules

Interchangeability checking is performed on a capability group basis. When enabled, interchangeability checking is always performed on the base capability group. In addition, interchangeability checking is performed on extension capability groups for which the user has ever set any of the attributes of the group. If the user has never set any attributes of an extension capability group, interchangeability checking is not performed on that group.

In general interchangeability warnings are generated if the following conditions are encountered:

An attribute that affects the behavior of the instrument is not in a state that the user specifies.

The user sets a class driver defined attribute to an instrument-specific value.

The user configures the value of an attribute that the class defines as read-only. In a few cases the class drivers define read-only attributes that specific drivers might implement as read/write.

The remainder of this section defines additional rules and exceptions for each capability group.

IviDigitizerBase Capability Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerBase capability group.

IviDigitizerArm Extension Group

1. If the Active Arm Source attribute is set to “None”, then no attributes in the IviDigitizerArm extension group need be in a user-specified state.

IviDigitizerBoardTemperature Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerBoardTemperature extension group.

IviDigitizerChannelFilter Extension Group

1. If the Input Filter Bypass attribute is set to TRUE, then no attributes in the IviDigitizerChannelFilter extension group need be in a user-specified state.

IviDigitizerChannelTemperature Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerChannelTemperature extension group.

IviDigitizerDataInterleavedChannels Extension Group

1. If the Data Interleaved Channel List Auto attribute is set to TRUE, then no attributes in the IviDigitizerDataInterleavedChannels extension group need be in a user-specified state.

IviDigitizerTimeInterleavedChannels Extension Group

1. If the Time Interleaved Channel List Auto attribute is set to TRUE, then no attributes in the IviDigitizerTimeInterleavedChannels extension group need be in a user-specified state.

IviDigitizerDownconversion Extension Group

The driver performs interchangeability checking on the IviDigitizerDownconversion group only if the application sets the Downconversion Enabled attribute to True.

IviDigitizerGlitchArm Extension Group

The driver performs interchangeability checking on the IviDigitizerGlitchArm group only if the application sets the Arm Type attribute to Glitch Arm.

IviDigitizerGlitchTrigger Extension Group

The driver performs interchangeability checking on the IviDigitizerGlitchTrigger group only if the application sets the Trigger Type attribute to Glitch Trigger.

IviDigitizerMultiArm Extension Group

1. If the Arm Source Operator is set to “None”, then no attributes in the IviDigitizerMultiArm extension group need be in a user-specified state.

IviDigitizerMultiRecordAcquisition Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerMultiRecordAcquisition extension group.

IviDigitizerMultiTrigger Extension Group

1. If the Trigger Source Operator is set to “None”, then no attributes in the IviDigitizerMultiTrigger extension group need be in a user-specified state.

IviDigitizerPretriggerSamples Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerPretriggerSamples extension group.

IviDigitizerReferenceOscillator Extension Group

1. If the Oscillator Source attribute is set to Internal, then the External Frequency attribute in the IviDigitizerMultiTrigger extension group need not be in a user-specified state.

IviDigitizerRuntArm Extension Group

1. The driver performs interchangeability checking on the IviDigitizerRuntArm group only if the application sets the Arm Type attribute to Runt Arm.
2. The Arm Level attribute must be in a user-specified state only if the application sets the Arm Type attribute to Runt Arm.

IviDigitizerRuntTrigger Extension Group

1. The driver performs interchangeability checking on the IviDigitizerRuntTrigger group only if the application sets the Trigger Type attribute to Runt Trigger.
2. The Trigger Level attribute must be in a user-specified state only if the application sets the Trigger Type attribute to Runt Trigger.

IviDigitizerSampleClock Extension Group

1. If the Sample Clock Source attribute is set to Internal, then the Sample Clock External Frequency attribute in the IviDigitizerSampleClock extension group need not be in a user-specified state.

IviDigitizerSampleClockOutput Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerSampleClockOutput extension group.

IviDigitizerSampleMode Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerSampleMode extension group.

IviDigitizerSelfCalibration Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerSelfCalibration extension group.

IviDigitizerSoftwareArm Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerSoftwareArm extension group.

IviDigitizerSoftwareTrigger Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerSoftwareTrigger extension group.

IviDigitizerTriggerHoldoff Extension Group

No additional interchangeability rules or exceptions are defined for the IviDigitizerTriggerHoldoff extension group.

IviDigitizerTVArm Extension Group

1. The driver performs interchangeability checking on the IviDigitizerTVArm group only if the application sets the Arm Type attribute to TV Arm.
2. The TV Arm Line Number attribute must be in a user specified state only if the application sets the TV Arm Event attribute to TV Trigger Event Line Number.

IviDigitizerTVTrigger Extension Group

1. The driver performs interchangeability checking on the IviDigitizerTVTrigger group only if the application sets the Trigger Type attribute to TV Trigger.

2. The TV Trigger Line Number attribute must be in a user specified state only if the application sets the TV Trigger Event attribute to TV Event Line Number.

IviDigitizerWidthArm Extension Group

The driver performs interchangeability checking on the IviDigitizerWidthArm group only if the application sets the Arm Type attribute to Width Arm.

IviDigitizerWidthTrigger Extension Group

The driver performs interchangeability checking on the IviDigitizerWidthTrigger group only if the application sets the Trigger Type attribute to Width Trigger.

IviDigitizerWindowArm Extension Group

The driver performs interchangeability checking on the IviDigitizerWindowArm group only if the application sets the Arm Type attribute to Window Arm.

IviDigitizerWindowTrigger Extension Group

The driver performs interchangeability checking on the IviDigitizerWindowTrigger group only if the application sets the Trigger Type attribute to Window Trigger.