A1560 SONIC

OEM ULTRASONIC PULSER/

RECEIVER FRONT-END UNIT

Programming manual

Applicable for A1560 Sonic-LF, A1560 Sonic-HF, A1560 Sonic-Air

Supported Firmware: 1.5, 1.6



Acoustic Control Systems – ACS Group

Saarbrücken, Germany

October 2019

Table of Contents

[1 Introduction 4](#_Toc53354498)

[2 A1560 API Description 5](#_Toc53354499)

[2.1 A1560 API Methods 5](#_Toc53354500)

[2.1.1 Connect(string ip) – Open network connection 5](#_Toc53354501)

[2.1.2 Start() – Start acquisition 5](#_Toc53354502)

[2.1.3 Stop() – Stop acquisition 6](#_Toc53354503)

[2.1.4 Flush() – Clean A1560 internal data buffers 6](#_Toc53354504)

[2.1.5 Disconnect() – Close network connection 6](#_Toc53354505)

[2.1.6 SetNewIp(string ip) – Reprogram IP address 6](#_Toc53354506)

[2.2 A1560 API Properties 8](#_Toc53354507)

[2.2.1 int TriggeringMode – Acquisition triggering mode 8](#_Toc53354508)

[2.2.2 int TriggeringIntervalUs – Periodic acquisition interval 8](#_Toc53354509)

[2.2.3 int PulseVoltage – Transmitter pulse amplitude 9](#_Toc53354510)

[2.2.4 int BurstFrequencyKhz – Transmitter burst frequency 9](#_Toc53354511)

[2.2.5 int BurstPeriodNs – Transmitter burst period 9](#_Toc53354512)

[2.2.6 int BurstLengthNumber – Transmitter burst duration 10](#_Toc53354513)

[2.2.7 int BurstDampDrationNs – Transmitter damping. 10](#_Toc53354514)

[2.2.8 bool TgcActive – Time gain compensation mode 10](#_Toc53354515)

[2.2.9 TgcPoint [] TgcCurve – Time gain compensation curve 10](#_Toc53354516)

[2.2.10 int TgcOffsetUs – Time gain compensation offset 11](#_Toc53354517)

[2.2.11 int [] TgcRawCurve – Time gain compensation raw curve 11](#_Toc53354518)

[2.2.12 int GainDb – Constant gain at input 11](#_Toc53354519)

[2.2.13 int InputFilterNumber – Analog high-pass filter 11](#_Toc53354520)

[2.2.14 int DigitalFilterHiPassKhz– Digital high-pass filter 12](#_Toc53354521)

[2.2.15 int DigitalFilterLoPassKhz– Digital low-pass filter 12](#_Toc53354522)

[2.2.16 int SamplingFrequencyMhz – Input sampling rate 12](#_Toc53354523)

[2.2.17 int VectorLengthSamples– Acquired data length 12](#_Toc53354524)

[2.2.18 int AveragingFactor – Acquisitions per averaged vector 12](#_Toc53354525)

[2.2.19 int AveragingPeriodNs – Constant averaging interval 13](#_Toc53354526)

[2.2.20 int AveragingRandomValueNs – Random averaging interval 13](#_Toc53354527)

[2.2.21 int TransducerType – Connected transducer type 14](#_Toc53354528)

[2.2.22 int TransducerMode – transducer polarity mode 14](#_Toc53354529)

[2.2.23 int CurrentBitRate – Incoming data rate 14](#_Toc53354530)

[2.2.24 int CurrentVectorRate – Incoming vector rate 15](#_Toc53354531)

[2.2.25 int UnitVersion –A1560 instrument version 15](#_Toc53354532)

[2.2.26 int SerialNumber –A1560 instrument serial number 15](#_Toc53354533)

[2.2.27 int FirmwareVersion –A1560 firmware version 15](#_Toc53354534)

[2.2.28 bool Connected – A1560 connection state 16](#_Toc53354535)

[2.3 A1560 Datatypes 17](#_Toc53354536)

[2.3.1 AcquisitionData Class 17](#_Toc53354537)

[2.3.2 TgcPoint Class 17](#_Toc53354538)

[2.4 A1560 API Events 18](#_Toc53354539)

[2.4.1 DataReceived Event 18](#_Toc53354540)

[2.5 A1560 API Exceptions 19](#_Toc53354541)

[2.5.1 InvalidOperationException 19](#_Toc53354542)

[2.5.2 ArgumentOutOfRangeException 19](#_Toc53354543)

[2.5.3 ArgumentException 19](#_Toc53354544)

[2.5.4 IOException 19](#_Toc53354545)

[2.5.5 NotSupportedException 19](#_Toc53354546)

[3 A1560 Application Examples 20](#_Toc53354547)

[3.1 C# Example 20](#_Toc53354548)

[3.2 VBA Example 21](#_Toc53354549)

[3.3 Delphi Example 22](#_Toc53354550)

[4 Document Versions 26](#_Toc53354551)

# Introduction

The A1560 SONIC (hereinafter “A1560”) is a product line of ultrasonic flaw detectors designed for stand-alone and integrated use in laboratory and industrial measurement and testing applications, facilitating ultrasonic data acquisition for different materials. The instrument possesses an integrated battery and wireless data communication for long-term, stand-alone operation with a high on-off time ratio. The A1560 can be operated in manual single-shot mode as well as in automatic mode with internal or external triggering to allow up to 2500 ultrasonic data acquisitions per second.

The main purpose of the A1560 is system integration and customization for a wide range of measurement and inspection tasks. The user is provided with a software development kit (SDK) for writing their own software applications.

The A1560 series is available in the following configurations:

Table 1. Available products

|  |  |  |
| --- | --- | --- |
| Instrument version | Frequency Range | Transducers |
| A1560 SONIC-LF | 10 kHz – 500 kHz | To be operated with DPC and regular low-frequency piezo-transducers. |
| A1560 SONIC-HF | 400 kHz – 15 MHz | Piezo-electric transducers with contact and immersion coupling. |
| A1560 SONIC-AIR | 10 kHz – 750 kHz | Air-coupled piezo-electric transducers. Modifications for different transducers are available. |

The main purpose of the A1560 is system integration and customization for a wide range of measurement and inspection tasks. The user is provided with a software development kit (SDK) for writing their own software applications. The SDK consists of:

* A set of dynamic-link libraries (hereinafter “A1560 API”)
* Sample software sources illustrating usage of the A1560 API
* A1560 Programming manual (this document)

Users can control A1560 from their software application via A1560 API. This document covers interaction with A1560 API version 1.1. It is highly recommended to familiarize yourself with the following A1560 documents:

* *A1560 User Manual*
* *A1560 Getting Started*

Safety symbols used in this Manual:

|  |  |
| --- | --- |
| Symbol | Description |
|  | Indicates a potentially harmful situation. If it is not avoided, the device may be damaged. |

# A1560 API Description

When referenced from a managed CLR code, A1560 API exposes A1560 class. An instance of this class provides the user with the means to connect to A1560 instrument and control it via A1560 API Methods, change parameters via A1560 API Properties and subscribe to an acquired data via an DataReceived Event.

A1560 API can also be used via its COM interop interface, allowing to use it from unmanaged code and from software products allowing COM interaction with external objects. To make the COM interface and type library visible to calling software it should be registered with regasm utility, included in .Net framework redistributable. For example, if version 4.0.30319 is installed, the following command executed with Administrator rights from the folder containing 32-bit API dlls (*A1560SDK1.1\A1560API\x86*) will perform the required actions.

C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm /codebase /tlb A1560APICOM.dll

The regasm.bat residing in the .dll folders will find the .NET framework installation and register the library automatically (Administrator privileges required).

## A1560 API Methods

Methods don’t return a value. Errors that might occur during an invocation of a method are indicated with exception. Handling the exceptions is a task of the user program.

### ­ConnectConnect()Connect(string ip) – Open network connection

This method opens network connection with A1560.

#### Argument

ip - IP Address of the unit to connect to

#### Exceptions

ArgumentException, ­IOException, NotSupportedException

#### Notes

Connect is the only method available after A1560 object instantiation. After the object creation or disconnection, you should call Connect method prior to calling any method or accessing any property (except the Connected property).

By default, A1560 units are shipped with IP = 192.168.1.2 for both Wi-Fi and Ethernet interface.

### StartStart() – Start acquisition

Calling Start() method will start a single acquisition or a sequence of acquisitions depending on the value of TriggeringMode property.

#### Exceptions

#### InvalidOperationException, ­IOException

#### Notes

Once started, a sequence of acquisitions will be continued by the instrument even if Disconnect is performed. Acquired data will be stored in internal buffer of A1560 until it is powered off. Therefore it is recommended to perform Stop and Flush after Connect, if the data possibly remaining in buffer is not needed.

### StopStop() – Stop acquisition

The method is used to stop a sequence of measurements. Stop is not needed in manual mode (TriggeringMode = 0).

#### Exceptions

#### InvalidOperationException, ­IOException

### FlushFlush() – Clean A1560 internal data buffers

Clean A1560 internal buffer as well as the input buffer of the API Library. See a note for the Start command.

#### Exceptions

#### InvalidOperationException, ­IOException

### DisconnectDisconnect() – Close network connection

#### Exceptions

No exceptions are raised by the Disconnect call.

#### Notes

Closed connection may be reopened on the same A1560 object.

User can call Disconnect on already disconnected object.

### SetNewIpSetNewIp(string ip) – Reprogram IP address

This method stores new IP address in A1560 nonvolatile memory.

#### Argument

ip - IP Address to assign to the unit.

#### Exceptions

ArgumentException, ­IOException

#### Notes

A1560 will become unavailable at the old IP address immediately after calling this method. The new IP address may not be applied until the next power on.

While the basic sanity checks are performed inside A1560 API before writing a new IP to the flash memory of A1560, it is still possible to program wrong IP rendering A1560 inaccessible via network. Caution is advised when changing the IP address. Please refer to the ***A1560 Advanced Configuration*** document for the information about fixing a misconfigured IP issue.

## A1560 API Properties

Acquisition parameters can be read or altered by reading and writing properties of A1560 object. The current state of A1560 can be polled with a subset of read-only properties. Please note than data acquisition parameters are stored in a RAM of A1560, therefore they must be set again once the instrument has been power-cycled.

InvalidOperationException, ArgumentOutOfRangeException or ­IOException will be raised when setting of any property is failed. Reading a property does not raise any exceptions.

### TriggeringModeint TriggeringMode – Acquisition triggering mode

This property is used to choose which event initiates an acquisition.

#### Acceptable values:

0 – Manual measurement mode: One acquisition will pe performed upon every Start method call. Internal timer, external encoder or CTP source will be ignored as well as device chain master sync signal. Please note that taking long data vectors might be slow in manual mode.

1 – Periodic mode: One acquisition will be performed every TriggeringIntervalUs microseconds.

2 – CTP mode: An acquisition will be started on every new CTP message[[1]](#footnote-1) from a triggering device.

3 – Encoder mode: An acquisition will be started on every increment of external encoder.

4 – Device chain slave: An acquisition will be initiated by a device chain master. Please note that A1560 behaves like the device chain master in all triggering modes except this.

5 – TTL mode: An acquisition will be started on every TTL pulse submitted to the CTP input of A1560.

#### Notes

TriggeringMode property should be set only when an acquisition is stopped. After a connection to A1560 is established with Connect command, acquisition is assumed to be previously started. Thus, a user must call Stop after Connect in order to set TriggeringMode.

Mode 5 (TTL) is supported in API 1.1 and higher.

### TriggeringIntervalUsint TriggeringIntervalUs – Periodic acquisition interval

This property sets or gets time in microseconds between two consecutive acquisition in periodic mode.

#### Acceptable values

100 - 1000000000

#### Notes

The property is applicable only when TriggeringMode = 1

It is not recommended to set a value lower than 400uS.

Highest acquisitions-per-second rates can be achieved only on relatively short vectors (VectorLengthSamples< 4000) due to connection bandwidth limitations.

### PulseVoltageint PulseVoltage – Transmitter pulse amplitude

This property sets or gets an amplitude in volts for a pulse burst sent to a transmitting transducer.

#### Acceptable values

20, 100, 200

#### Notes

In this context, pulse voltage is the highest voltage of the half-wave. For example, a one-period-long 20V pulse peak-to-peak voltage is 40V (-20V to +20V).

Due to the way voltage switching is implemented in hardware, this command takes ~1s to return.

### BurstFrequencyKhzint BurstFrequencyKhz – Transmitter burst frequency

This property sets or gets the frequency in kilohertz for a pulse burst sent to a transmitting transducer.

#### Acceptable values

for A1560 SONIC-LF: 25 – 500

for A1560 SONIC-HF: 500 – 10000

for A1560 SONIC-AIR: 400 – 600

#### Notes

Due to implementation the period of the pulse is always a multiple of 20nS. For that reason, the real pulse frequency might differ from the requested one. For example, when BurstFrequencyKhz = 800 (period = 1250nS) is set, the real period of the impulse is 1240nS and the real pulse frequency is ~806kHz. Real pulse frequency and period set in A1560 can be obtained by reading BurstFrequencyKhz and BurstPeriodNs properties respectively.

### BurstPeriodNsint BurstPeriodNs – Transmitter burst period

This property sets or gets the period in nanoseconds for a pulse burst sent to a transmitting transducer.

#### Acceptable values

for A1560 SONIC-LF: 2000 – 40000

for A1560 SONIC-HF: 100 – 2000

for A1560 SONIC-AIR: 1660 – 2500

#### Notes

BurstFrequencyKhz and BurstPeriodNs change the same physical pulse parameter and the second is introduced for user’s convenience. BurstFrequencyKhz= 106/ BurstPeriodNs .

Due to implementation a pulse period is always a multiple of 20nS. For that reason, a real pulse period might differ from the requested one. For example, when BurstPeriodNs= 125 (frequency = 8000kHz) is set, the real period of the impulse is 140nS and the real pulse frequency is ~8333kHz. Real pulse period and pulse frequency set in A1560 can be obtained by reading BurstPeriodNs and BurstFrequencyKhz properties respectively.

### BurstLengthNumberint BurstLengthNumber – Transmitter burst duration

This property sets or gets number of half-periods in transmitter burst.

#### Acceptable values

1 - 10

#### Notes

For example, when BurstLengthNumber =2, the pulse duration is one full period (negative and positive half waves), or 1uS for BurstFrequencyKhz = 1000.

### BurstLengthNumberint BurstDampDrationNs – Transmitter damping.

A transducer damping circuit can be activated after a pulse burst transmission is finished for a period of time set by this property.

#### Acceptable values

0 - 2000000000

#### Notes

When BurstDampDrationNs = 0, transducer damping won’t take place.

Supported in API 1.1 and higher.

### TgcActivebool TgcActive – Time gain compensation mode

When set to false, time gain compensation curve defined by

TgcCurve is inactive, and flat fixed gain defined by GainDb is applied to all portions of input signal. When set to false, amplification for every sample is defined by both and GainDb .

### TgcCurveTgcPoint [] TgcCurve – Time gain compensation curve

This property sets or gets an array of TgcPoint defining variable signal gain for the different portions of an acquired data vector.

#### Notes

When TgcActive = true, each element of the TgcCurve array sets an individual gain for the specific moment of an acquisition. The gain is set implicitly for every sample by TgcCurve approximation.

This property is not supported in API version 1.0 and 1.1

### TgcOffsetUsint TgcOffsetUs – Time gain compensation offset

TgcOffsetUs is specified in microseconds and allows to delay TgcCurve application during acquisition.

#### Notes

This property is not supported in API version 1.0 and 1.1

### int [] TgcRawCurve – Time gain compensation raw curve

An array of compensation constants, each of which defines the gain for an acquired data vector point with the same index.

#### Acceptable values

1272 (for the highest gain of 40 dB) – 15128 (for the lowest gain of 0 dB)

#### Notes

TgcRawCurve is applied to an input signal at the moment of its acquisition only when TgcActive is true.

TgcRawCurve size should correspond to VectorLengthSamples (curve length and acquired vector length should be equal).

GainDb values of -20, 0, 20 and 40 will add the corresponding gain to the gain set by TgcRawCurve

Supported in API 1.1 and higher.

### GainDbint GainDb – Constant gain at input

This property sets or gets analog amplification in decibels for the signal obtained from a receiving transducer.

#### Acceptable values

-20 - +80 When TgcActive = false

-20 - +40 when TgcActive = true

### InputFilterNumberint InputFilterNumber – Analog high-pass filter

This property selects an analog filter at the input of A1560.

#### Acceptable values

Actual cut-off frequency is version-specific:

|  |  |  |  |
| --- | --- | --- | --- |
| Value | A1560 SONIC-LF | A1560 SONIC-AIR | A1560 SONIC-HF |
| 0 | 10 kHz | 10 kHz | 400 kHz |
| 1 | 20 kHz | 20 kHz | 800 kHz |
| 2 | 40 kHz | 40 kHz | 1600 kHz |
| 3 | 100 kHz | 100 kHz | 3200 kHz |

### DigitalFilterHiPassKhzint DigitalFilterHiPassKhz– Digital high-pass filter

This property gets or sets digital high-pass filter cut-off frequency in kilohertz.

#### Acceptable values

10 – 20000

#### Notes

This property is not supported in API version 1.0 and 1.1

### DigitalFilterLoPassKhzint DigitalFilterLoPassKhz– Digital low-pass filter

This property gets or sets digital low-pass filter cut-off frequency in kilohertz.

#### Acceptable values

10 – 20000

#### Notes

This property is not supported in API version 1.0 and 1.1

### SamplingFrequencyMhzint SamplingFrequencyMhz – Input sampling rate

This property gets or sets frequency in MHz for AD conversion of the input signal.

#### Acceptable values

1, 2, 5, 10, 25, 50, 100

### VectorLengthSamplesint VectorLengthSamples– Acquired data length

This property gets or sets number of samples in an acquired data vector.

#### Acceptable values

1024 – 131072

#### Notes

VectorLengthSamples property should be set only when an acquisition is stopped in order to prevent data loss due to mismatch between expected/buffered vector length. After a connection to A1560 is established with Connect command, acquisition is assumed to be previously started. Thus a user must call Stop after Connect in order to set VectorLengthSamples.

Samples count is directly related to the amount of data generated by A1560 on each acquisition and therefore affects maximum acquisition rate per second.

### AveragingFactorint AveragingFactor – Acquisitions per averaged vector

A1560 can make several pulses/acquisitions in a row and internally calculate an averaged vector from the results, when AveragingFactor > 0.

#### Acceptable values

0-8

#### Notes

Number of acquisitions required to produce one averaged data vector is calculated as 2AveragingFactor.

AveragingFactor does not affect the quantity of acquired data vectors sent to a client by A1560. For example, if TriggeringMode= 1, int TriggeringIntervalUs – Periodic acquisition interval

= 1000000 and AveragingFactor = 3, one vector is sent by A1560 each second, and every sent vector is a result of internal averaging of eight acquisitions performed over a relatively short period of time (see AveragingPeriodNsand AveragingRandomValueNs for timings).

 A safe pulse repetition rate of 3000 pulse cycles per second should not be exceeded. It is calculated e.g. in timed mode as  
BurstLengthNumber \* 2 \* TriggeringIntervalUs \* 2AveragingFactor

### AveragingPeriodNs AveragingPeriodNs int AveragingPeriodNs – Constant averaging interval

This property is defined in nanoseconds and gets or sets a constant part of an interval between acquisitions in averaging mode.

When A1560 performs several pulses/acquisitions in a row for the following averaging a pause will take place after an acquisition is finished. It is calculated as *FixedDelay* + AveragingPeriodNs + *RandomInterval*, where FixedDelay is a hardware delay of 22μs and *RandomInterval* is a random number in a range from 0 to AveragingRandomValueNs

#### Acceptable values

0 – 65535

#### Notes

AveragingPeriodNs is rounded with 10ns precision.

### AveragingRandomValueNs AveragingRandomValueNsint AveragingRandomValueNs – Random averaging interval

This property is defined in nanoseconds and gets or sets a random part of an interval between acquisitions in averaging mode.

When A1560 performs several pulses/acquisitions in a row for the following averaging a pause will take place after an acquisition is finished. It is calculated as *FixedDelay* + AveragingPeriodNs + *RandomInterval*, where FixedDelay is a hardware delay of 22μs and *RandomInterval* is a random number in a range from 0 to AveragingRandomValueNs

#### Acceptable values

0 – 65535

#### Notes

For example, when AveragingFactor = 1, AveragingPeriodNs = 10000 and AveragingRandomValueNs= 10000, the second acquisition will take place in 32-42μs after the first acquisition is finished.

Please refer to ***Signal Averaging*** section of ***A1560 User Manual*** document for further information.

AveragingRandomValueNsis rounded with 10ns precision.

### TransducerTypeTransducerTypeint TransducerType – Connected transducer type

This property defines the type of transducer(s) used with the unit.

#### Acceptable values:

0 – Dual-crystal transducer or trough transmission. Pulse burst will be generated at the “OUT” socket of A1560.

1 – Single-crystal transducer. Pulse burst will be generated at the “IN” socket of A1560.

2 – Output disabled (receive-only mode). Pulse burst will not be generated neither at the “IN” socket nor at the “OR” socket of A1560.

#### Notes

TransducerType = 2 is supported in API 1.1 and higher.

### TransducerTypeTransducerTypeint TransducerMode – transducer polarity mode

This property defines the initial polarity of the pulse burst generated at the transducer output.

#### Acceptable values:

0 – The burst starts with the positive pulse.

1 – The burst starts with the negative pulse.

#### Notes

Supported in API 1.1 and higher.

### CurrentBitRateCurrentBitRateint CurrentBitRate – Incoming data rate

This property can be used to estimate the amount of data sent by A1560 in bits per second.

#### Possible values

0 – 100000000

#### Notes

This is a read-only property

### CurrentVectorRateint CurrentVectorRate – Incoming vector rate

This property can be used to estimate the number of valid data vectors sent by A1560 in a second

#### Possible values

0 – 3000

#### Notes

This is a read-only property

### UnitVersionDeviceVariantint UnitVersion –A1560 instrument version

This property is used to detect the version of the connected instrument.

#### Possible values

0 – Unknown device

1 – A1560 (unknown version)

2 – A1560 SONIC-LF

3 – A1560 SONIC-AIR

4 – A1560 SONIC-HF

#### Notes

This is a read-only property

### UnitVersionDeviceVariantint SerialNumber –A1560 instrument serial number

This property is used to read the serial number of the connected instrument.

#### Notes

This is a read-only property

Supported in API 1.1 and higher.

### UnitVersionDeviceVariantint FirmwareVersion –A1560 firmware version

This property is used to detect the firmware version installed on the connected instrument.

#### Possible values

0 – Firmware version below 1.3

13 – version 1.3

14 – version 1.4

15 – version 1.5

16 – version 1.6

17 – version 1.7

#### Notes

This is a read-only property.

Supported in API 1.1 and higher.

### Connectedbool Connected – A1560 connection state

This property is used to determine if the connection to a unit is established.

#### Possible values

False – no connection to A1560

True – connection to A1560 established

#### Notes

This is a read-only property

## A1560 Datatypes

### AcquisitionDataAcquisitionData Class

Each acquired data vector is represented by an AcquisitionData object containing an array of samples along with an additional information.

#### Public read only properties:

short[] Vector – 12-bit signed samples of an acquired vector.

Int Id – vector number, counting from Start method call.

Int[3] Ctp – vector coordinate (spatial, temporal) provided by CTP interface or internal trigger.

### TgcPointTgcPoint Class

An object of TgcPoint class defines one point of the time gain compensation curve.

#### Public read only properties:

int OffsetUs – defines time in microseconds passed from the start of an acquisition.

double GainDb – defines the gain in decibels for a sample taken at OffsetUs moment.

#### Possible values

OffsetUs – 0-65000

GainDb – 0-40

#### Notes

Eventual temporal coordinate set by OffsetUs is affected by TgcOffsetUs global offset.

## A1560 API Events

### DataReceivedDataReceived Event

Acquired data is provided to the user’s application via the DataReceived event which bears AcquisitionData object:

public delegate void DataReceivedDelegate(AcquisitionData data);

public event DataReceivedDelegate DataReceived;

#### Notes

User code should be subscribed to the DataReceived event in order to obtain the acquired data.

The event is blocking. In addition to A1560 internal hardware buffer for unsent vectors, A1560 API has its own software buffer that can grow very large if a subscriber processes the incoming DataReceived events too slowly.

## A1560 API Exceptions

On error, standard exceptions are thrown:

### InvalidOperationException

Raised when a method is called, or a property is changed before opening connection to A1560 with Connect method. TriggeringMode or VectorLengthSamples setting will also raise InvalidOperationException when attempted without stopping automatic acquisition first.

### ArgumentOutOfRangeException

Thrown when unacceptable value for a property is set.

### ArgumentException

Argument passed to Connect or SetNewIp method is not a usable IPv4 address.

### ­IOException

Thrown in case of communication problems, e.g. connection cannot be open or was closed by the device or failed in some other way (A1560 was switched off, etc.).

### NotSupportedException

Instrument version is unknown to A1560API or firmware version is not supported by API. Compatible firmware versions are 1.5 and 1.6. The exception is raised on Connect method call.

# A1560 Application Examples

## C# Example

The code below illustrates essential concepts of working with A1560 API from a C# application. Exception handling, buffer flushing and other important parts of code are skipped for brevity.

Remember to reference A1560APICOM.dll in the project.

A1560ApiTestCsharpApp example project included in SDK distribution provides more details.

using System;

using System.Linq;

using A1560APICOM;

namespace MinimalExample

{

class Program

{

private static readonly A1560 A1560 = new A1560();

static void Main(string[] args)

{

A1560.Connect("192.168.1.2");

//Automatic acquisition is assumed running on connect, stop it.

A1560.Stop();

//Set Device to make one measurement per Start() command

A1560.TriggeringMode = 0;

//Subscribe dedicated processing function to be called for each vector

A1560.DataReceived += A1560OnDataReceived;

//Launch

A1560.Start();

//Keep application alive for a moment (so we can see the vector event)

Console.ReadKey();

A1560.Disconnect();

}

private static void A1560OnDataReceived(AcquisitionData data)

{

var vector = data.Vector;

Console.WriteLine($"Vector received, Length: {vector.Length} samples,  
CTP={data.Ctp[0]:X08}:{data.Ctp[1]:X08}:{data.Ctp[2]:X08},   
maximum sample value = {vector.Max()} ");

//Unsubscribe, since we’re getting only one vector in this app.

A1560.DataReceived -= A1560OnDataReceived;

//Inform users they can leave now

Console.WriteLine("Press any key to exit");

}

}

}

## VBA Example

This example can be run in Microsoft® Excel and presents every acquired vector as a column of values. ACS A1560 COM API component should be registered in OS and referenced in VBA script. Refer to Chapter 2 for the details of COM assembly registration with regasm.

Option Explicit

Private WithEvents m\_A1560 As A1560

'event handler writes vector sample values to column A of an Excel sheet

Private Sub m\_A1560\_DataReceived(ByVal data As AcquisitionData)

Dim Vector() As Integer

Dim maxindex As Integer

Dim i As Integer

Vector = data.Vector

maxindex = UBound(Vector)

For i = 0 To maxindex

Cells(i + 1, 1).Value = Vector(i)

Next i

End Sub

'Connect to A1560 and set it up to acquire a vector every second.

Private Sub StartAcquisition()

Set m\_A1560 = New A1560APICOM.A1560

m\_A1560.Connect ("192.168.1.2")

m\_A1560.Stop

m\_A1560.VectorLengthSamples = 1024

m\_A1560.TransducerType = 1

m\_A1560.SamplingFrequencyMhz = 100

m\_A1560.GainDb = 0

m\_A1560.BurstFrequencyKhz = 10000

m\_A1560.BurstLengthNumber = 2

m\_A1560.PulseVoltage = 20

m\_A1560.TriggeringIntervalUs = 1000000

m\_A1560.TriggeringMode = 1

m\_A1560.Start

End Sub

Private Sub CommandButton1\_Click()

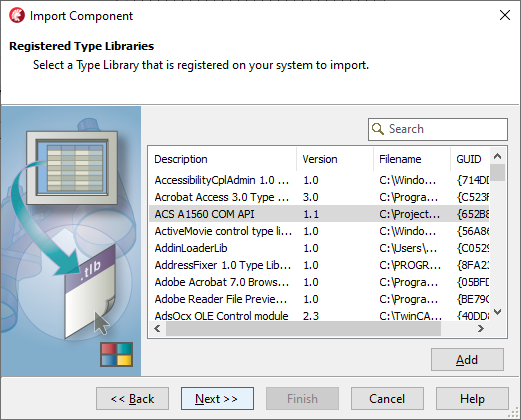
StartAcquisition

End Sub

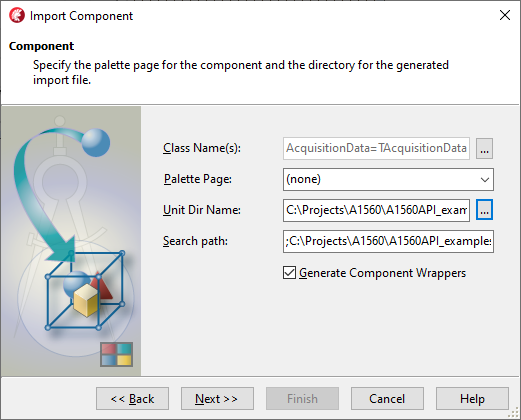
## Delphi Example

Delphi provides a fast and convenient way to import COM type library and generate a wrapper incapsulating and dealing with all details of COM interaction. ACS A1560 COM API component should be registered in OS before importing. Refer to Chapter 2 for the details of COM assembly registration with regasm. Alternatively, library registration may be performed from RAD IDE itself (Administrator privileges required).

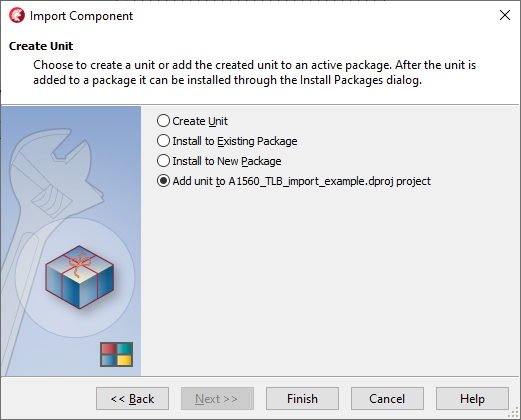
To import, Use Components -> Import Component menu entry, select “Import a Type Library” radio button and click “Next”. A list showing all registered type libraries will appear. Select “ACS A1560 COM API” and click “Next”.



Further, check the “Generate Component Wrappers” checkbox and click “Next”



After selecting “Add unit to project” radio button and clicking “Finish” TA1560 wrapper class can be used to interact with A1560 hardware with properties, methods and events described in this document.



The code fragments below demonstrate how to read and write properties, invoke methods, and process events from A1560 COM object wrapped in such a manner. It is a part of A1560API\_Delphi\_TLB\_import code example included in the SDK.

{Event “Callback” processing and displaying data coming from A1560}

procedure TForm1.ShowAscan(ASender: TObject; const data: IAcquisitionData);

var i : Integer;

var vector: OleVariant;

begin

vector := data.Vector;

Chart1.Series[0].Clear;

for i := VarArrayLowBound(vector, 1) to VarArrayHighBound(vector, 1) do

begin

Chart1.Series[0].AddXY(i, vector[i], '', clTeeColor);

end;

end;

{TCP connection to A1560 is established with “Connect” button click”

procedure TForm1.ButtonConnectClick(Sender: TObject);

begin

A1560.Connect1(EditIp.Text);

end;

{Single pulse/acquisition sequence is performed when “Start” button is clicked}

procedure TForm1.Button1Click(Sender: TObject);

begin

A1560.Stop(); //Most of the settings can be changed only when stopped.

A1560.VectorLengthSamples := 1024; //make vector shorter (default is 32768)

A1560.TransducerType := 1; //Set single transducer mode

A1560.TriggeringMode := 0; //Set manual mode

A1560.Start(); //Start in manual mode performs one

end;

{When an application is started, A1560 object is initialized, and a processing procedure is subscribed to its OnDataReceived event}

procedure TForm1.FormCreate(Sender: TObject);

begin

A1560 := TA1560.Create(nil); //All the COM functionality is hidden behind TA1560

A1560.OnDataReceived := self.ShowAscan;//ShowAscan is called for every new vector

end;

# Document Versions

Table 2 *A1560 Programming Manual* Document Versions

|  |  |  |
| --- | --- | --- |
| Version number | Issue date | Comment |
| 0.9 | 01.07.2019 | Initial release. |
| 1.0 | 07.08.2019 | COM Interface mentioned, TGC – related properties introduced, some property names changed, VBA example added |
| 1.1 | 19.08.2019 | Naming changes according to current SDK/API version. Hyperlinking. |
| 1.2 | 17.12.2019 | Updated for API 1.1 – added FirmwareVersion, SerialNumber, TgcRawCurve, BurstDampDurationNs, TransducerMode properties, modified TriggeringMode, TransducerType properties. |
| 1.3 | 11.10.2020 | COM object registration notes in Chapter 2, Delphi example in section 3.3 |

1. New CTP message is a message with an ID that is different from that of previous CTP message [↑](#footnote-ref-1)