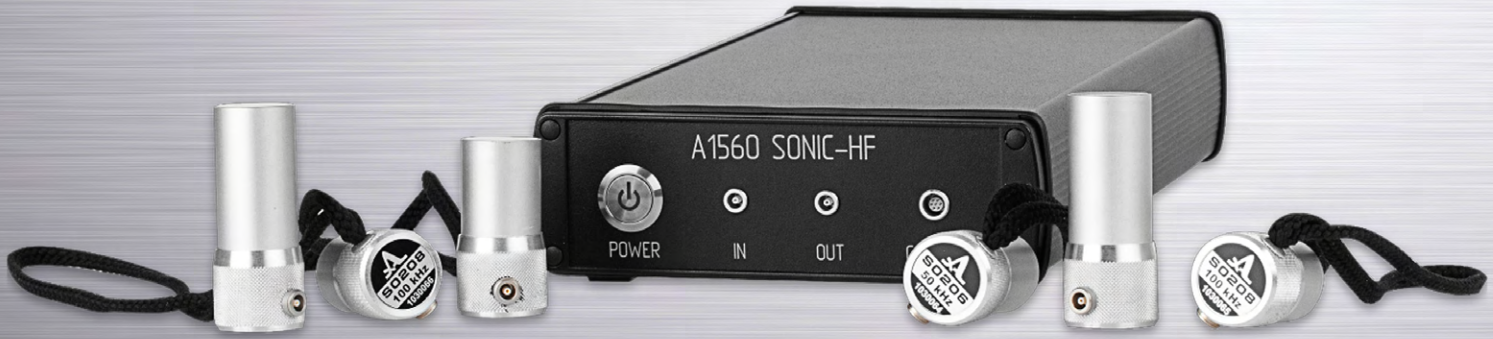
PROGRAMMING MANUAL

A1560 SONIC

OEM ULTRASONIC PULSER/ RECEIVER FRONT-END UNIT

APPLICABLE FOR A1560 SONIC-LF, A1560 SONIC-HF, A1560 SONIC-AIR



Acoustic Control Systems – ACS Group
Saarbrücken, Germany 2019

CE

Issue date: 27.06.2019

TABLE OF CONTENTS

1

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 the A1560 from their software application via the A1560 API. This document covers interaction with the A1560 API version 1.0. 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
NOTICE	Indicates a potentially harmful situation. If it is not avoided, the device or something in its vicinity may be damaged.

2

A1560 API DESCRIPTION

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

The A1560 API can also be used via its COM interop interface, allowing it to be used from unmanaged code and from software products allowing COM interaction with external objects.

2.1 A1560 API METHODS

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

2.1.1 Connect(string ip) – Open network connection

This method opens a network connection with the A1560.

Argument

ip - IP address of the unit being connected to.

Exceptions

ArgumentException, IOException

Notes

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

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

2.1.2 Start() – Start acquisition

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

Exceptions

InvalidOperationException, IOException

Notes

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

2.1.3 Stop() – Stop acquisition

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

Exceptions

InvalidOperationException, IOException

2.1.4 Flush() – Clean A1560 internal data buffers

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

Exceptions

InvalidOperationException, IOException

2.1.5 Disconnect() – Close network connection

Exceptions

No exceptions are raised by the Disconnect call.

Notes

Closed connections may be reopened on the same A1560 object.

The user can call a Disconnect on an already disconnected object.

2.1.6 SetNewIp(string ip) – Reprogram IP address

This method stores a new IP address in the A1560's nonvolatile memory.

Argument

ip - IP address to be assigned to the unit.

Exceptions

ArgumentException, IOException

Notes

The 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.

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

2.2 A1560 API PROPERTIES

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

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

2.2.1 int TriggeringMode – Acquisition triggering mode

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

Acceptable values:

0 – Manual measurement mode: One acquisition will be performed upon every Start method call. The internal timer, external encoder or CTP source will be ignored, as well as the 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] from a triggering device or on every TTL pulse submitted to the CTP input of the A1560.

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

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

Notes

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

[1] *A new CTP message is a message with an ID that is different from that of the previous CTP message.*

2.2.2 int TriggeringIntervalUs – Periodic acquisition interval

This property sets or gets time in microseconds between two consecutive acquisitions 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.

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

2.2.3 int 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.

2.2.4 int 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 the period set in the A1560 can be obtained by reading BurstFrequencyKhz and BurstPeriodNs properties, respectively.

2.2.5 int 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 the A1560 can be obtained by reading BurstPeriodNs and BurstFrequencyKhz properties, respectively.

2.2.6 int BurstLengthNumber – Transmitter burst duration

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

2.2.7 bool TgcActive – Time gain compensation mode

When set to false, the time gain compensation curve defined by TgcCurve is inactive, and the 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.

2.2.8 TgcPoint [] TgcCurve – Time gain compensation curve

This property sets or gets an array of TgcPoint defining variable signal amplification 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.

2.2.9 int TgcOffsetUs – Time gain compensation offset

TgcOffsetUs is specified in microseconds and allows a delay of the TgcCurve application during acquisition.

2.2.10 int 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

2.2.11 int InputFilterNumber – Analog high-pass filter

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

Acceptable values

Version-specific:

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

2.2.12 int DigitalFilterHiPassKhz– Digital high-pass filter

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

Acceptable values

10 – 20000

Notes

This property is not supported in API version 1.0

2

2.2.13 int DigitalFilterLoPassKhz– Digital low-pass filter

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

Acceptable values

10 – 20000

Notes

This property is not supported in API version 1.0

2.2.14 int SamplingFrequencyMhz – Input sampling rate

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

Acceptable values

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

2.2.15 int VectorLengthSamples– Acquired data length

This property gets or sets the 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 a mismatch between the expected/buffered vector length. After a connection to the A1560 is established with the Connect command, acquisition is assumed to be previously started. Thus, a user must call a Stop after a Connect in order to set the VectorLengthSamples.

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

2.2.16 int AveragingFactor – Acquisitions per averaged vector

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

Acceptable values

0-10

Notes

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

The AveragingFactor does not affect the quantity of acquired data vectors sent to a client by the A1560. For example, if TriggeringMode = 1, int TriggeringIntervalUs – Periodic acquisition interval = 1000000 and AveragingFactor = 3, one vector is sent by the 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 AveragingPeriodNs and AveragingRandomValueNs for timings).

NOTICE 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 * $2^{AveragingFactor}$

2.2.17 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 the 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.

2.2.18 int 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 the 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 the **Signal Averaging** section of the **A1560 User Manual document** for further information.

AveragingRandomValueNsis rounded with 10ns precision.

2.2.19 int TransducerType – Connected transducer type

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

Acceptable values:

0 – Dual-crystal transducer or through transmission. The pulse burst will be generated at the "OUT" socket of the A1560.

1 – Single-crystal transducer. The pulse burst will be generated at the "IN" socket of the A1560.

2.2.20 int CurrentBitRate – Incoming data rate

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

Possible values

0 – 100000000

Notes This is a read-only property.

2.2.21 int CurrentVectorRate – Incoming vector rate

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

Possible values

0 – 3000

Notes This is a read-only property.

2.2.22 int 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.

2.2.23 bool Connected – A1560 connection state

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

Possible values

False – no connection to the A1560
True – connection to the A1560 established

Notes This is a read-only property.

2

Programming manual

2.3 A1560 DATATYPES

2.3.1 AcquisitionData Class

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

Public read-only properties:

- short[] Vector – 12-bit signed samples of an acquired vector.
- Int Id – vector number, counting from the Start method call.
- Int[3] Ctp – vector coordinate (spatial, temporal) provided by a CTP interface or internal trigger.

2.3.2 TgcPoint 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 the OffsetUs moment.

Possible values

- OffsetUs – 0-65000
- GainDb – 0-40

Notes

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

2.4 A1560 API EVENTS

2.4.1 DataReceived Event

Acquired data is provided to the user's application via the DataReceived event, which bears the 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 the A1560 internal hardware buffer for unsent vectors, the A1560 API has its own software buffer that can grow very large if a subscriber processes the incoming DataReceived events too slowly.

2

2.5 A1560 API EXCEPTIONS

Upon error, standard exceptions are thrown:

2.5.1 InvalidOperationException

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

2.5.2 ArgumentOutOfRangeException

Thrown when an unacceptable value for a property is set.

2.5.3 ArgumentException

An argument passed to the Connect or SetNewIp method is not a usable IPv4 address.

2.5.4 IOException

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

2.5.5 NotSupportedException

The instrument version is unknown to the A1560 API (the exception is raised on the Connect method call).

A1560 APPLICATION EXAMPLES

3.1 C# EXAMPLE

The code below illustrates essential concepts for working with the A1560 API from a C# application. Exception handling, buffer flushing and other important parts of the 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");
        }

    }
}
```

3.2 VBA EXAMPLE

This example can be run in Microsoft® Excel and presents every acquired vector as a column of values. The ACS A1560 COM API component should be registered in OS and referenced in the VBA script.

```
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
```


ACS

A1560 SONIC

OEM ULTRASONIC PULSER/RECEIVER FRONT-END UNIT

PROGRAMMING MANUAL

Acoustic Control Systems – ACS Group
Saarbrücken, Germany 2019