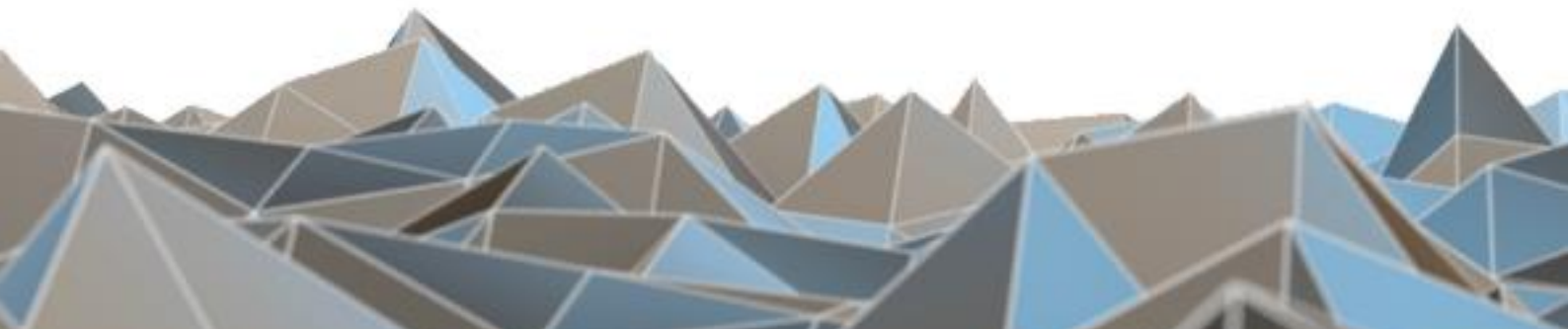


BLUETECHNIX
Embedding Ideas

BltTofApi MATLAB SDK

Software User Manual

Version 4



Contact

Bluetechnix Group GmbH
Gutheil-Schoder-Gasse 17, 1230 Wien
AUSTRIA
office@bluetechnix.com
<http://www.bluetechnix.com>

Date: 2017-05-17

Table of Contents

1	Introduction	6
1.1	Purpose of the document	6
2	Software Architecture.....	7
2.1	Overview	7
2.2	Interfaces.....	7
2.2.1	API Version.....	7
2.2.2	Initialization	7
2.2.3	Connection Parameters	8
2.2.4	Frame Mode Parameter	8
2.2.5	Connecting.....	8
2.2.6	Get Device Information	9
2.2.7	Frame Retrieval	9
2.2.8	Frame Cleanup	10
2.2.9	Get Distances	10
2.2.10	Get Amplitudes	11
2.2.11	Get XYZ coordinates.....	11
2.2.12	Get Phases.....	12
2.2.13	Get Flags.....	12
2.2.14	Get 2D Data	13
2.2.15	Get Intensities	13
2.2.16	Disconnecting	14
2.2.17	Get Frame Rate.....	14
2.2.18	Set Frame Rate	14
2.2.19	Get Integration Time	15
2.2.20	Set Integration Time.....	15
2.2.21	Get Global Offset	16
2.2.22	Set Global Offset	16
2.2.23	Set Frame Mode	16
2.2.24	Read Register	17
2.2.25	Write Register	17
2.2.26	Save current configuration.....	18
2.2.27	Restore default configuration	18
2.2.28	Reset Device	18
2.2.29	Get channel data.....	19
2.2.30	Get metadata	19

2.2.31	Set library parameters	20
2.2.32	Get library parameters	20
2.2.33	Error code	21
3	References	22
4	Document Revision History	23
A	List of Figures and Tables	24

© Bluetechnix 2017
All Rights Reserved.

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.

Terms of delivery and rights of technical change reserved.

We hereby disclaim any warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Bluetechnix makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. Bluetechnix specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Bluetechnix takes no liability for any damages and errors causing of the usage of this board. The user of this board is responsible by himself for the functionality of his application. He is allowed to use the board only if he has the qualification. More information is found in the General Terms and Conditions (AGB).

Information

For further information on technology, delivery terms and conditions and prices please contact Bluetechnix <http://www.bluetechnix.com>.

Warning

Due to technical requirements components may contain dangerous substances.

1 Introduction

1.1 Purpose of the document

This document explains the usage of the Bluetechnix ToF MATALAB API. It does not cover device specific information or any implementation related information. It only describes the interface.

2 Software Architecture

2.1 Overview

The BltTofApi MATLAB SDK gives the possibility to get access to the BltTofApi interface in MATLAB. Every ToF system built by or for Bluetechnix shall be accessible by this common interface. The BltTof MATLAB SDK V2.2.7 is build up on the C libraries BltTofApi V2.2.7

2.2 Interfaces

The BltTofApi MATLAB SDK supports the essential interfaces of the BltTofApi.

2.2.1 API Version

In order to get the version of the API call BTAgetVersion.

Calling syntax:

```
[status, version, buildDateTime, supportedDeviceTypes] = BTAgetVersion;
```

Input parameter:

none

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
version	- Version number
buildDateTime	- Build date and time
supportedDeviceTypes	- Supported device types
	0xA9C1 - Sentis-ToF-M100
	0xB320 - Argos 3D P320
	0x9BA6 - Argos 3D P310
	0xA3C4 - Argos 3D P100
	0x5032 - Sentis-ToF-P510

2.2.2 Initialization

First, the library must be configured via the configuration c-structure. The configuration structure must be initialized with standard values using the function BTAinitConfig. The specific implementation of the library defines which parameters are required and which can be left out. The required parameters must then be set to a valid value before calling BTAopen.

Calling syntax:

```
[status, configStruct] = BTAinitConfig;
```

Input parameter:

none

Output parameter:

status - Error code for error handling. Refer to 2.2.33.
configStruct - Configuration structure

2.2.3 Connection Parameters

A library might define different parameter sets for different connection modes (For example if only TCP configuration interface parameters are set, no data interface connection will be established to the sensor).

Example for Argos3D - P310 and connection mode UDP/TCP:

```
configStruct.udpDataIpAddr = [224, 0, 0, 1];  
configStruct.udpDataIpAddrLen = 4;  
configStruct.udpDataPort = 10002;  
configStruct.tcpDeviceIpAddr = [192, 168, 0, 10];  
configStruct.tcpDeviceIpAddrLen = 4;  
configStruct.tcpControlPort = 10001;  
configStruct.frameQueueMode = 1;  
configStruct.frameQueueLength = 5;  
configStruct.deviceType = 1;
```

2.2.4 Frame Mode Parameter

The frame mode can be configured in order to get the desired data channels from the sensor / library:

```
configStruct.frameMode = 1;
```

Possible frame modes:

- 1 - Distances and amplitudes
- 2 - Distance, amplitudes and flags (only supported by USB based devices)
- 3 - XYZ coordinates
- 4 - XYZ coordinates and amplitudes
- 5 - Distances, amplitudes and 2D color (only supported by Ethernet based devices)
- 6 - XYZ coordinates, amplitudes and flags (only supported by USB based devices)
- 7 - Raw Phases
- 8 - Intensities (only supported by USB based devices)

2.2.5 Connecting

Now that the configuration structure is filled in, the connection is ready to be opened:

Calling syntax:

```
[status, deviceHandle] = BTAopen(configStruct);
```


Input parameter:

`configStruct` - Configuration structure

Output parameter:

`status` - Error code for error handling. Refer to 2.2.33.
`deviceHandle` - Handle to the device

2.2.6 Get Device Information

For querying information about the device call `BTAgetDeviceInfo`.

Calling syntax:

```
[status, deviceType, productOrderNumber, serialNumber, firmwareVersion] =
BTAgetDeviceInfo(deviceHandle);
```

Input parameter:

`deviceHandle` - Device handle

Output parameter:

`status` - Error code for error handling. Refer to 2.2.33.
`deviceType` - Device type:
 0xA9C1 - Sentis-ToF-M100
 0xB320 - Argos 3D P320
 0x9BA6 - Argos 3D P310
 0xA3C4 - Argos 3D P100
 0x5032 - Sentis-ToF-P510
`productOrderNumber` - Product order number of the device
`serialNumber` - Serial number of the device
`firmwareVersion` - Firmware version

2.2.7 Frame Retrieval

A frame can be actively requested from the library. The function `BTAgetFrame` is blocking, so a timeout can be specified. A timeout == 0 results in endless waiting for a frame.

Calling syntax:

```
[status, frameHandle, frameCounter, timeStamp] = BTAgetFrame(deviceHandle,
timeout);
```

Input parameter:

deviceHandle	- Device handle
timeout	- Timeout to wait if no frame is yet available in [ms]. If timeout = 0 the function waits endlessly for a frame.

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
frameHandle	- Frame handle (needs to be free'd with BTAfreeFrame)

2.2.8 Frame Cleanup

A successful call to BTAgetFrame always demands for a call to BTAfreeFrame.

Calling syntax:

```
[status] = BTAfreeFrame(frameHandle);
```

Input parameter:

frameHandle	- Frame handle
-------------	----------------

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
--------	---------------------------------------------------

2.2.9 Get Distances

BTAgetDistances extracts distances from a provided frame. If there is no channel with distances data present in the frame, an error is returned.

Calling syntax:

```
[status, distData, integrationTime, modulationFrequency, unit] =  
BTAgetDistances(frameHandle);
```

Input parameter:

frameHandle	- Frame handle
-------------	----------------

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
distData	- Array which contains the distances data.
integrationTime	- Integration time
modulationFrequency	- Modulation frequency
unit	- Unit of the data. 0 - unit less 1 - meter 2 - centimeter

3 - millimeter

2.2.10 Get Amplitudes

BTGetAmplitudes extracts amplitudes from a provided frame. If there is no channel with amplitudes data present in the frame, an error is returned.

Calling syntax:

```
[status, distData, integrationTime, modulationFrequency, unit] =
BTGetAmplitudes(frameHandle);
```

Input parameter:

frameHandle - Frame handle

Output parameter:

```
status                         - Error code for error handling. Refer to 2.2.33.
ampData                       - Array which contains the amplitudes data.
integrationTime               - Integration time
modulationFrequency          - Modulation frequency
unit                           - Unit of the data.
                              0 - unit less
                              1 - meter
                              2 - centimeter
                              3 - millimeter
```

2.2.11 Get XYZ coordinates

BTGetAmplitudes extracts 3D-Coordinates from a provided frame. If there is no channel with 3D-coordinates present in the frame, an error is returned.

Calling syntax:

```
[status, xData, yData, zData, integrationTime, modulationFrequency, unit] =
BTGetXYZcoordinates(frameHandle);
```

Input parameter:

frameHandle - Frame handle

Output parameter:

```
status                         - Error code for error handling. Refer to 2.2.33.
xData                          - Array which contains the cartesian x coordinates.
yData                          - Array which contains the cartesian y coordinates.
zData                          - Array which contains the cartesian z coordinates.
integrationTime               - Integration time
modulationFrequency          - Modulation frequency
unit                           - Unit of the data.
                              0 - unit less
                              1 - meter
```

- 2 - centimeter
- 3 - millimeter

2.2.12 Get Phases

BTAGetDistances extracts the phases from a provided frame. If there is no channel which contains phases, an error is returned.

Calling syntax:

```
[status, phase0, phase90, phase180, phase270, integrationTime,
modulationFrequency, unit] = BTAGetPhases(frameHandle);
```

Input parameter:

frameHandle - Frame handle

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
phase0	- Array which contains the phase0
phase90	- Array which contains the phase90
phase180	- Array which contains the phase180
phase270	- Array which contains the phase270
integrationTime	- Integration time
modulationFrequency	- Modulation frequency
unit	- Unit of the data.
	0 - unit less
	1 - meter
	2 - centimeter
	3 - millimeter

2.2.13 Get Flags

BTAGetFlags extracts the flags from a provided frame. If there is no channel which contains flags, an error is returned.

NOTE: This function is supported only by USB based devices.

Calling syntax:

```
[status, flags, integrationTime, modulationFrequency, unit] =
BTAGetFlags(frameHandle);
```

Input parameter:

frameHandle - Frame handle

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
flags	- Array which contains the flags
integrationTime	- Integration time

```

modulationFrequency - Modulation frequency
unit                - Unit of the data.
                    0 - unit less
                    1 - meter
                    2 - centimeter
                    3 - millimeter

```

2.2.14 Get 2D Data

BTAgget2DData extracts the 2D color data from a provided frame. If there is no channel which contains 2D color data, an error is returned.

NOTE: This function is supported only by Ethernet based devices.

Calling syntax:

```

[status, colorData2D, integrationTime, modulationFrequency, dataFormat] =
BTAgget2DData(frameHandle);

```

Input parameter:

```

frameHandle        - Frame handle

```

Output parameter:

```

status            -      0      Success
                   -      1      The frame mode is selected correctly but the
                                given frame doesn't contain 2D color data.
                                The reason for this are different frame
                                rates of the ToF and the 2D sensor.
                   - Others Refer to 2.2.33.
colorData2D       - Array which contains the 2D color data
integrationTime    - Integration time
modulationFrequency - Modulation frequency
dataFormat         - BTA_DataFormat, see bta_frame.h

```

2.2.15 Get Intensities

BTAggetIntensities extracts the intensities from a provided frame. If there is no channel which intensity data, an error is returned.

NOTE: This function is supported only by USB based devices.

Calling syntax:

```

[status, intensities, integrationTime, modulationFrequency, unit] =
BTAggetIntensities(frameHandle);

```

Input parameter:

```

frameHandle        - Frame handle

```

Output parameter:

```

status            -      Error code for error handling. Refer to 2.2.33.

```

intensities	- Array which contains intensities
integrationTime	- Integration time
modulationFrequency	- Modulation frequency
unit	- Unit of the data.
	0 - unit less
	1 - meter
	2 - centimeter
	3 - millimeter

2.2.16 Disconnecting

In order to disconnect the sensor and stop the service, simply call BTAClose.

Calling syntax:

```
status = BTAClose(deviceHandle);
```

Input parameter:

deviceHandle	- Device handle
--------------	-----------------

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
--------	---------------------------------------------------

2.2.17 Get Frame Rate

BTAGetFrameRate returns the current frame rate of the device.

Calling syntax:

```
[status, frameRate] = BTAGetFrameRate(deviceHandle);
```

Input parameter:

deviceHandle	- Device handle
--------------	-----------------

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
frameRate	- Frame rate

2.2.18 Set Frame Rate

The frame rate can be changed with BTASetFrameRate.

Calling syntax:

```
status = BTASetFrameRate(deviceHandle, frameRate);
```

Input parameter:

deviceHandle	- Device handle
frameRate	- Frame rate to be set

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
--------	---------------------------------------------------

2.2.19 Get Integration Time

BTAGetIntegrationTime returns the current integration time.

Calling syntax:

```
[status, integrationTime] = BTAGetIntegrationTime (deviceHandle);
```

Input parameter:

deviceHandle	- Device handle
--------------	-----------------

Output parameter:

status	- Error code for error handling. Refer 2.2.33.
integrationTime	- Integration time

2.2.20 Set Integration Time

The integration time can be changed with BTASetIntegrationTime.

Calling syntax:

```
status = BTASetIntegrationTime (deviceHandle, integrationTime);
```

Input parameter:

deviceHandle	- Device handle
integrationTime	- Integration time in [ms] to be set

Output parameter:

status	- Error code for error handling. Refer 2.2.33.
--------	------------------------------------------------

2.2.21 Get Global Offset

BTAGetGlobalOffset returns the distance offset being applied to all pixels equally.

Calling syntax:

```
[status, globalOffset] = BTAGetGlobalOffset(deviceHandle);
```

Input parameter:

deviceHandle	- Device handle
--------------	-----------------

Output parameter:

status	- Error code for error handling. Refer 2.2.33.
globalOffset	- Global offset in milimeter

2.2.22 Set Global Offset

The distance offset being applied to all pixels equally can be changed with BTASetGlobalOffset.

Calling syntax:

```
status = BTASetGlobalOffset (deviceHandle, globalOffset);
```

Input parameter:

deviceHandle	- Device handle
globalOffset	- Global offset in milimeter

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
--------	---------------------------------------------------

2.2.23 Set Frame Mode

BTASetGlobalOffset allows specifying which channels will be included in a BTA_Frame, see the BLT_FrameMode for possible options.

Calling syntax:

```
status = BTASetFrameMode(deviceHandle, frameMode);
```

Input parameter:

deviceHandle	- Device handle
frameMode	- The desired frame-mode

Possible frame modes:

- 1 - Distances and amplitudes
- 2 - Distance, amplitudes and flags (not supported by all devices)
- 3 - XYZ coordinates
- 4 - XYZ coordinates and amplitudes
- 5 - Distances, amplitudes and 2D color (not supported by all devices)
- 6 - XYZ coordinates, amplitudes and flags (not supported by all devices)
- 7 - Raw Phases
- 8 - Intensities (not supported by all devices)

Output parameter:

status - Error code for error handling. Refer to 2.2.33.

2.2.24 Read Register

BTAreadRegister reads out the data of one register.

Calling syntax:

```
[status, data] = BTAreadRegister(deviceHandle, address);
```

Input parameter:

deviceHandle - Device handle
address - The address in the register map to read from

Output parameter:

status - Error code for error handling. Refer to 2.2.33.
data - Register data

2.2.25 Write Register

BTAwriteRegister writes a value into the specified register address.

Calling syntax:

```
[status] = BTAwriteRegister(deviceHandle, address, data);
```

Input parameter:

deviceHandle - Device handle
address - The address in the register map to write to.
data - Data to write

Output parameter:

`status` - Error code for error handling. Refer to 2.2.33.

2.2.26 Save current configuration

BTWriteCurrentConfigToNvm writes the current configuration (i.e. register settings) to non-volatile memory.

Calling syntax:

```
[status] = BTWriteCurrentConfigToNvm(deviceHandle);
```

Input parameter:

`deviceHandle` - Device handle

Output parameter:

`status` - Error code for error handling. Refer to 2.2.33.

2.2.27 Restore default configuration

BTRestoreDefaultConfig restores the default configuration (but does not save it).

NOTE: This function is not supported by all devices. Refer to the software user manual of your device.

Calling syntax:

```
[status] = BTWriteCurrentConfigToNvm(deviceHandle);
```

Input parameter:

`deviceHandle` - Device handle

Output parameter:

`status` - Error code for error handling. Refer to 2.2.33.

2.2.28 Reset Device

BTSendReset initiates a reset of the device.

NOTE: This function is not supported only by all devices. Refer to the software user manual of your device.

Calling syntax:

```
[status] = BTAsendReset(deviceHandle);
```

Input parameter:

```
deviceHandle      - Device handle
```

Output parameter:

```
status            - Error code for error handling. Refer to 2.2.33.
```

2.2.29 Get channel data

BTAGetChannelData extracts data of a specific channel from a provided frame. If the requested channel is not available, an error is returned.

Calling syntax:

```
[status, channelData, channelID, integrationTime, modulationFrequency,  
unit] = BTAGetChannelData(frameHandle, channelNr);
```

Input parameter:

```
frameHandle      - Frame handle  
channelNr        - Number of the channel
```

Output parameter:

```
status            - Error code for error handling. Refer to 2.2.33.  
channelData       - Array which contains the data from the given channel.  
channelID         - ID of the channel.  
integrationTime   - Integration time  
modulationFrequency - Modulation frequency  
unit              - Unit of the data.  
                   0 - unit less  
                   1 - meter  
                   2 - centimeter  
                   3 - millimeter
```

2.2.30 Get metadata

BTAGetMetadata extracts metadata of a specific channel from a provided frame. If the requested channel is not available, an error is returned. If there is no metadata with the right id present in the channel, an error is returned.

Calling syntax:

```
[status, metadata, metadataLen] = BTAGetMetadata(frameHandle, channelNr,  
metadataId);
```

Input parameter:

frameHandle	- Frame handle
channelNr	- Number of the channel
metadataID	- ID of metadata

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
metadata	- Byte array which contains the data from the given channel.
metadataLen	- length of the metadata in bytes

2.2.31 Set library parameters

Function for setting a parameter for the library. Library parameters do not directly affect the camera's configuration.

Calling syntax:

```
[status] = BTASetLibParam(deviceHandle, libParam, value);
```

Input parameter:

deviceHandle	- Device handle
libParam	- Identifier for the parameter (refer to the BltTofApi header files for a description of its function)
value	- The value to be set for the library parameter

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
--------	---------------------------------------------------

2.2.32 Get library parameters

Function for getting a parameter from the library.

Calling syntax:

```
[status value] = BTAGetLibParam(deviceHandle, libParam);
```

Input parameter:

deviceHandle	- Device handle
libParam	- Identifier for the parameter (refer to the BltTofApi header files for a description of its function)

Output parameter:

status	- Error code for error handling. Refer to 2.2.33.
value	- The value of the library parameter

2.2.33 Error code

-32768	-	BTA_StatusInvalidParameter
-32767	-	BTA_StatusIllegalOperation
-32766	-	BTA_StatusTimeOut
-32765	-	BTA_StatusDeviceUnreachable
-32764	-	BTA_StatusNotConnected
-32763	-	BTA_StatusInvalidVersion
-32762	-	BTA_StatusRuntimeError
-32761	-	BTA_StatusOutOfMemory
-32760	-	BTA_StatusNotSupported
-32779	-	BTA_StatusCrcError
-32778	-	BTA_StatusUnknown

3 References

https://support.bluetechnix.at/wiki/Bluetechnix_ToF_API_v2

https://support.bluetechnix.at/wiki/BltTofApi_Matlab_SDK

4 Document Revision History

Version	Date	Author	Description
0	2014 08 22	BRA	Initial Draft
1	2014 09 16	BRA	SW Release V1.0.0
1.3	2015 02 02	BRA	SW Release V1.3.0
2	2016 10 14	BRA	SW Release V2.1.3
3	2016 11 18	BRA	SW Release V2.2.3
4	2017 05 17	BRA	SW Release V2.2.7

Table 4.1: Revision history

A List of Figures and Tables

Figures

No table of figures entries found.

Tables

Table 4.1: Revision history	23
-----------------------------------	----