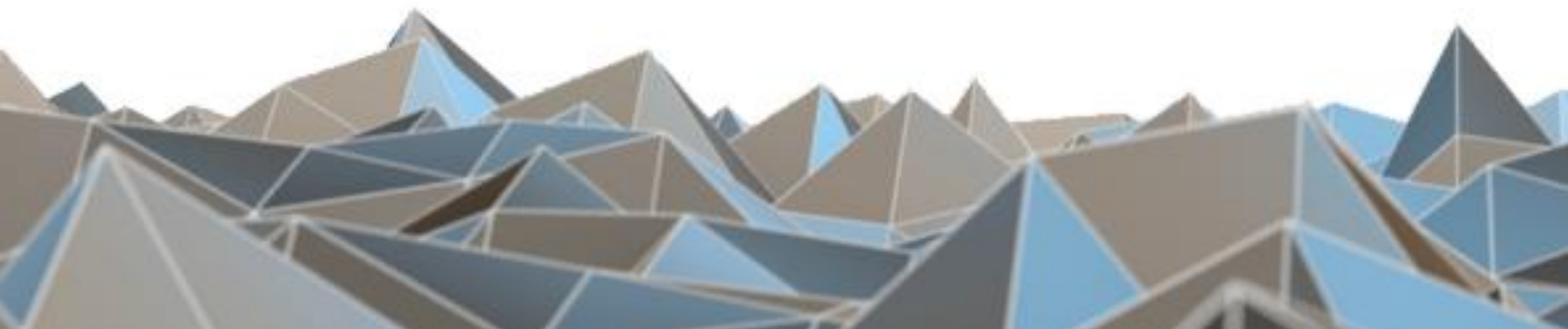


BLUETECHNIX
Embedding Ideas

BltTofApi MATLAB SDK

Software User Manual

Version 1





Contact

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

Date: 2014-08-22

Table of Contents

1	Introduction	5
1.1	Purpose of the document	5
2	Software Architecture.....	6
2.1	Overview	6
2.2	Interfaces.....	6
2.2.1	Initialization	6
2.2.2	Connection Parameters	6
2.2.3	Frame Mode Parameter	7
2.2.4	Connecting.....	7
2.2.5	Keep Alive Messages Interval.....	7
2.2.6	Get Frame Rate.....	7
2.2.7	Set Frame Rate	8
2.2.8	Get Integration Time	8
2.2.9	Set Integration Time.....	9
2.2.10	Get Global Offset	9
2.2.11	Set Global Offset	9
2.2.12	Set Frame Mode	10
2.2.13	Register Read	10
2.2.14	Register Write	11
2.2.15	Frame Retrieval	11
2.2.16	Get Distances	11
2.2.17	Get Amplitudes	12
2.2.18	Get XYZ coordinates.....	12
2.2.19	Frame Cleanup	13
2.2.20	Disconnecting	13
2.2.21	Get Version	13
2.2.22	Get Device Info	14
3	References	15
4	Document Revision History.....	16
A	List of Figures and Tables	17

© Bluetechnix R&D GmbH 2014

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 MATLAB 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.

2.2 Interfaces

The BltTofApi MATLAB SDK supports the essential interfaces of the SDK defined in bta.h. The headers bta_frame.h, bta_status.h, bta_event.h, bta_discovery.h and bta_flash_update.h contain more declarations and are included by bta.h.

2.2.1 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 BTA_Status in bta_status.h.
configStruct	- Configuration structure

2.2.2 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;
```

2.2.3 Frame Mode Parameter

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

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

In order to get to know the supported frame modes refer to BTA_FrameMode in bta_frame.h.

2.2.4 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 BTA_Status in bta_status.h.
deviceHandle	- Handle to the device

2.2.5 Keep Alive Messages Interval

The interval of the keepalive messages can be set with BTASetKeepAliveMsgInterval.

Calling syntax:

```
status = BTASetKeepAliveMsgInterval(deviceHandle, interval);
```

Input parameter:

deviceHandle	- Device handle
interval	- The interval in seconds for keep-alive-messages to be sent.

Output parameter:

status	- Error code for error handling. Refer to BTA_Status in bta_status.h.
--------	---

2.2.6 Get Frame Rate

BTAGetFrameRate returns the current theoretical frame rate of the default capture sequence.

Calling syntax:

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

Input parameter:

deviceHandle - Device handle

Output parameter:

status - Error code for error handling. Refer to BTA_Status in
 bta_status.h.
frameRate - Frame rate

2.2.7 Set Frame Rate

The frame rate can be changed with BTAs setFrameRate.

Calling syntax:

```
status = BTAs setFrameRate(deviceHandle, frameRate);
```

Input parameter:

deviceHandle - Device handle
frameRate - Frame rate to be set

Output parameter:

status - Error code for error handling. Refer to BTA_Status in
 bta_status.h.

2.2.8 Get Integration Time

BTAgetIntegrationTime returns the current integration time of the default capture sequence.

Calling syntax:

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

Input parameter:

deviceHandle - Device handle

Output parameter:

status - Error code for error handling. Refer to BTA_Status in
 bta_status.h.

integrationTime - Integration time

2.2.9 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 to `BTA_Status` in `bta_status.h`.

2.2.10 Get Global Offset

Function for getting 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 to `BTA_Status` in `bta_status.h`.
globalOffset - Global offset in [mm]

2.2.11 Set Global Offset

Function for setting the distance offset being applied to all pixels equally.

Calling syntax:

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

Input parameter:

deviceHandle	- Device handle
globalOffset	- Global offset in [mm]

Output parameter:

status	- Error code for error handling. Refer to BTA_Status in bta_status.h.
--------	---

2.2.12 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

Output parameter:

status	- Error code for error handling. Refer to BTA_Status in bta_status.h.
--------	---

2.2.13 Register Read

BTAreadRegister reads out the data of one register.

Calling syntax:

```
[status, data, length] = 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 BTA_Status in bta_status.h.
data	- Register data

2.2.14 Register Write

BTWriteRegister writes a value into the specified register address.

Calling syntax:

```
[status] = BTWriteRegister(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 BTA_Status in bta_status.h.
--------	---

2.2.15 Frame Retrieval

BTAGetFrame requests a frame from the corresponding device.

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 BTA_Status in bta_status.h.
frameHandle	- Frame handle (needs to be free'd with BTAfreeFrame)

2.2.16 Get Distances

BTAgetDistances - Convenience function for extracting 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 BTA_Status in
                    bta_status.h.
distData            - Array which contains the distances data.
integrationTime     - Integration time
modulationFrequency - Modulation frequency
unit               - Unit of the data. Refer to BTA_Unit in bta_frame.h
```

2.2.17 Get Amplitudes

BTAgetAmplitudes - Convenience function for extracting 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] =
BTAgetAmplitudes(frameHandle);
```

Input parameter:

```
frameHandle          - Frame handle
```

Output parameter:

```
status              - Error code for error handling. Refer to BTA_Status in
                    bta_status.h.
ampData            - Array which contains the amplitudes data.
integrationTime     - Integration time
modulationFrequency - Modulation frequency
unit               - Unit of the data. Refer to BTA_Unit in bta_frame.h
```

2.2.18 Get XYZ coordinates

BTAgetXYZcoordinates - Convenience function for extracting the 3D-Coordinates from a provided frame. If there is no channel with coordinate data present in the frame, an error is returned

Calling syntax:

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

Input parameter:

```
frameHandle          - Frame handle
```

Output parameter:

status	- Error code for error handling. Refer to BTA_Status in bta_status.h.
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. Refer to BTA_Unit in bta_frame.h

2.2.19 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 BTA_Status in bta_status.h.
--------	---

2.2.20 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 BTA_Status in bta_status.h.
--------	---

2.2.21 Get 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 BTA_Status in bta_status.h.
version	- Version number
buildDateTime	- Build date and time
supportedDeviceTypes	- Supported device types

2.2.22 Get Device Info

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 BTA_Status in bta_status.h.
deviceType	- Device type
productOrderNumber	- Product order number of the device
serialNumber	- Serial number of the device
firmwareVersion	- Firmware version number

3 References

bta.h

bta_frame.h

bta_status.h

bta_discovery.h

bta_flash_update.h

bta_event.h



4 Document Revision History

Version	Date	Author	Description
1	2014 08 22	BRA	Initial Draft

Table 4.1: Revision history

A List of Figures and Tables

Figures

No table of figures entries found.

Tables

Table 1.1: Reference Overview	Error! Bookmark not defined.
Table 4.1: Revision history	16