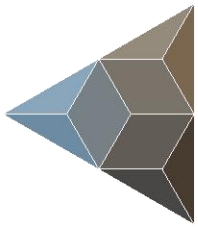
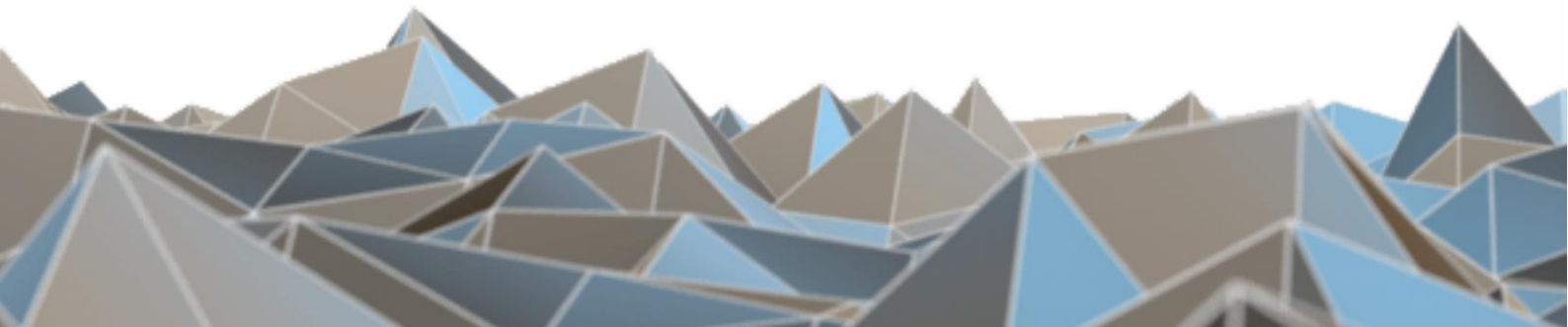


BLUETECHNIX
Embedding Ideas

BItTofApi v2.1 SDK

User Manual

Version 1



Contact

Bluetechnix R&D GmbH

Gutheil-Schoder-Gasse 17, 1230 Wien

AUSTRIA

office@bluetechnix.com

http://www.bluetechnix.com

Date: 2016-02-09

Table of Contents

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 API.

2 Overview

In order to create a common interface for our products we define the interfaces between a ToF device and an application. The main part of this model is the BltTofApi which is written in C for platform independency. The BltTofApiExt (used by BltTofSuite) is able to access the BltTofApi interface and will therefore be compatible with any device with existing lib implementing the BltTofApi. Any other user application can access the cameras via the BltTofApi.

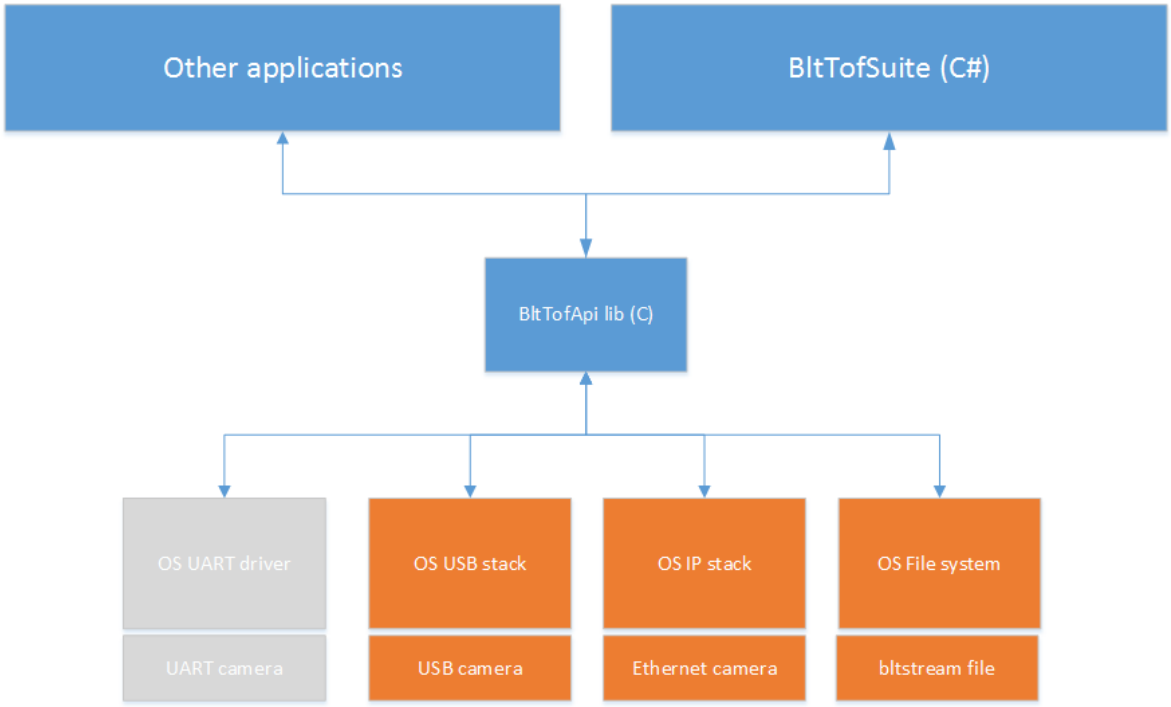


Figure 2-1: Interfacing concept

Every ToF system built by or for Bluetechnix shall be accessible by this common interface. The Interface is kept as simple as possible and covers all functionalities of all ToF sensors.

The pixel order and coordinate system is defined as described in the following image.

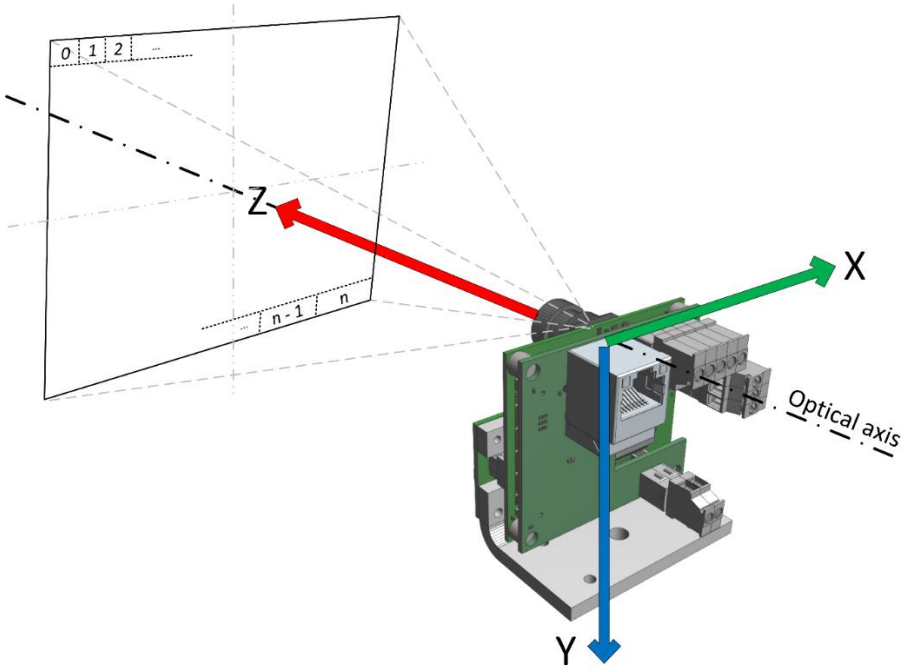


Figure 2-2: ToF coordinate system (here with a Sentins-ToF-M100)

The ToF device specific coordinate system orientation and origin can be found in its corresponding hardware user manual.

3 BTA_Config parameters

3.1 General

This section only explains parameters that behave the same with all camera interfaces. For device/interface specific information, please refer to the corresponding section.

- **DeviceType**: In order for the lib to know what interface to use to connect to a camera, a generic or specific device type can be specified. If none is specified (i.e. 0), all interfaces are tried sequentially.
- **Framemode**: Setting this parameter is the same as calling BTAsetFrameMode.
- **InfoEventEx**: Register a callback for the identification of the source of an error and for logging purposes. You are very welcome to include these logs when contacting Bluetechnix support.
- **Verbosity**: Change the amount of infoEvents.

3.2 USB connection

When operating USB cameras please be aware of the behavior of the SDK described below.

3.2.1 BTAgetFrame vs. frameArrived callback

Q… frameQueueing enabled
C… frameArrived callback registered

Q	C	Behaviour	BTAgetFrame	frameArrived
		Internal capturing is disabled. No thread is running in background.	Calling BTAgetFrame triggers the readout of a frame via USB. The call might take even longer when no frame is available.	Disabled.
	X	Internal capturing by the SDK is enabled. A thread repeatedly reads frames from USB and delivers them via callback. With very low frame rates or slow frame triggering the internal thread blocks the USB interface. **Do not use this configuration in that case**.	Disabled.	Frames are delivered as soon as the capture thread gets them. This is the fastest way for a frame from camera to the user.
X		Internal capturing by the SDK is enabled. A thread repeatedly reads frames from USB and queues them. With very low frame rates or slow frame triggering the internal thread blocks the USB interface. **Do not use this configuration in that case**.	Calling BTAgetFrame delivers a frame from the queue. The call is fast, but the frame might be old.	Disabled.

X	X	Internal capturing by the SDK is enabled. A thread repeatedly reads frames from USB, delivers them via callback and queues them. With very low frame rates or slow frame triggering the internal thread blocks the USB interface. **Do not use this configuration in that case**.	Calling BTAgetFrame delivers a frame from the queue. The call is fast, but the frame might be old.	Frames are delivered as soon as the capture thread gets them. This is the fastest way for a frame from camera to the user.

3.2.2 Other BTA_Config parameters

- All parameters starting with tcp, udp, uart or bltstream are ignored.
- PON and serial number must match the value in the camera's register exactly.
- calibFileName is supported. Please use files provided by Bluetechnix.
- zFactorsFileName is supported. Please use files provided by Bluetechnix.
- AverageWindowLength: If > 1 the camera is configured for sequencing and a filter is instantiated in order to combine sequences of the same frame into one frame, averaging valid values.

3.2.3 Hotplugging

The USB cable must be connected and the camera must be up and running before BTAopen is called. If the USB connection is somehow lost, the BTA_Handle must be closed and reopened.

3.2.4 LibParams

EnableTestPattern: Enable and disable the test pattern. Frames retrieved from the camera are overwritten with test data. The phase data received from the camera is overwritten with values resulting in a distance equal to the pixels index (rounding errors may occur). The X, Y and Z channels are generated by those distances resulting in a rather odd shape. Raw phases and intensities channels are overwritten with the pixel index value.

3.3 Ethernet connection

When operating Ethernet cameras please be aware of the behavior of the SDK described below.

3.3.1 BTAgetFrame vs. frameArrived callback

Q… frameQueueing enabled
C… frameArrived callback registered

Q	C	Behaviour	BTAgetFrame	frameArrived
		If a UDP data connection is configured, BTAopen will return an error because it is	Disabled.	Disabled.

		not possible to retrieve a frame		
	X	The internal thread handling UDP packets delivers the frames via callback.	Disabled	Frames are delivered as soon as the UDP packets are parsed. This is the fastest way for a frame from camera to the user.
X		The internal thread handling UDP packets queues the frames.	Calling BTAgetFrame delivers a frame from the queue. The call is fast, but the frame might be old.	Disabled
X	X	The internal thread handling UDP packets delivers the frames via callback and queues them.	Calling BTAgetFrame delivers a frame from the queue. The call is fast, but the frame might be old.	Frames are delivered as soon as the UDP packets are parsed. This is the fastest way for a frame from camera to the user.

3.3.2 Other BTA_Config parameters

- All parameters starting with uart or bltstream are ignored.
- The three connections udpData, udpControl and tcpControl can be configuered completely individually. At least one of the three connections must be provided. If tcpControl and udpControl are configured both, then both connections are tried to establish.
- PON: The device itself is only aware of its serial number which is matched to a string of one or several PONs separated by a slash. The PON parameter must be contained in that string in order to be a match.
- Serial number: must match the value in the camera's register exactly.
- calibFileName is not supported. Please use BTAflashUpdate.
- zFactorsFileName is not supported.
- AverageWindowLength is not supported.

3.3.3 Hotplugging

The UDP data connection is established regardless of the presence of a camera. The UDP/TCP control connection can only be established if the camera responds to the first alive message. If the control connection is subsequently lost, the library tries to reconnect until BTAclose is called or the connection was reestablished.

3.3.4 LibParams

KeepAliveMsgInterval: Set and get the interval at which alive messages are sent in order to keep the connection alive / check if the connection is alive.
CrcControlEnabled: Enable and disable the usage of a crc checksum control interface communication. When disabled, the crc checksum is only used for file transmissions, not for register operations and the like.
EnableTestPattern: Enable and disable the test pattern. Frames retrieved from the camera are overwritten with test data. Regardless of the channel id, every pixel is set to the value of its index (position).

3.4 Bltstream connection

When reading a stream from file please be aware of the behavior of the SDK described below.

3.4.1 BTAgetFrame vs. frameArrived callback

Q… frameQueueing enabled
C… frameArrived callback registered

Q	C	Behaviour	BTAgetFrame	frameArrived
		BTAopen will return an error because it is not possible to retrieve a frame	Disabled.	Disabled.
	X	The internal thread reading the frames from the bltstream file delivers the frames via callback.	Disabled	Frames are delivered as recorded (with the same timing as they were grabbed).
X		The internal thread reading the frames from bltstream queues the frames.	Calling BTAgetFrame delivers a frame from the queue.	Disabled
X	X	The internal thread reading the frames from the bltstream file delivers the frames via callback and queues them.	Calling BTAgetFrame delivers a frame from the queue.	Frames are delivered as recorded (with the same timing as they were grabbed).

3.4.2 Other BTA_Config parameters

All parameters except bltstreamFilename, frameQueueMode, frameQueueLength, frameArrived and frameArrivedEx are ignored.

3.4.3 Bltstream handling

The bltstream file has a plain text header at the beginning. In BTAopen the library reads some information from the bltstream and starts a thread. That thread starts reading frames and (after some buffering) provides them via the API. Jumping to specific index inside the stream rather than accessing the frames sequentially can take longer.
File format v1 limitation: Total frame count is not supported, i.e. it is not known how many frames there are in the file.

3.4.4 LibParams

StreamTotalFrameCount: Get the total amount of frames stored in the current bltstream file (only supported since file format v2).
StreamAutoPlaybackSpeed: Get and set the playback speed. Playback is timed by the frame timestamps (as recorded) times this factor.
StreamPos: Get and set the index of the current frame. The stream, i.e. frames in the file are sequentially numbered starting with 0.
StreamPosIncrement: Set the index of the current frame relatively to the current index.

4 Build instructions

4.1 Windows

Please take a look at the example project for a reference on how to compile an application using a BTA library.

Dependencies:

- Define the macro #define PLAT_WINDOWS
- Microsoft Visual C++ Redistributable for Visual Studio 2015 (can be downloaded from the Microsoft homepage)

When using a **USB (P100)** camera, consider:

- The libusb0 driver must be installed (Provided by Bluetechnix)

4.2 Linux

Please refer to the Makefile of the provided example for building your own applications.
Dependencies:

- libusb-0.1.0
- libpthread-2.15
- libjpeg
- libopencv-core
- libopencv-imgproc

When using a **USB (P100)** camera, consider:

- Copy the .rules file provided by Bluetechnix to /lib/udev/rules.d/

4.3 ARM (Linux)

See section 4.2

5 Recommended documents

The newest version of this document can be downloaded from:

http://datasheets.bluetechnix.at/goto/BltTofApi/v2.1/doc/

The reference manual can be downloaded from:

http://datasheets.bluetechnix.at/goto/BltTofApi/v2.1/doc/BltTofApi_v2.1_reference_manual/index.html

6 Document Revision History

Version	Date	Author	Description
1	2015 12 22	AFA	Initial Draft

Table 6.1: Revision history

A List of Figures and Tables

Figures

Tables