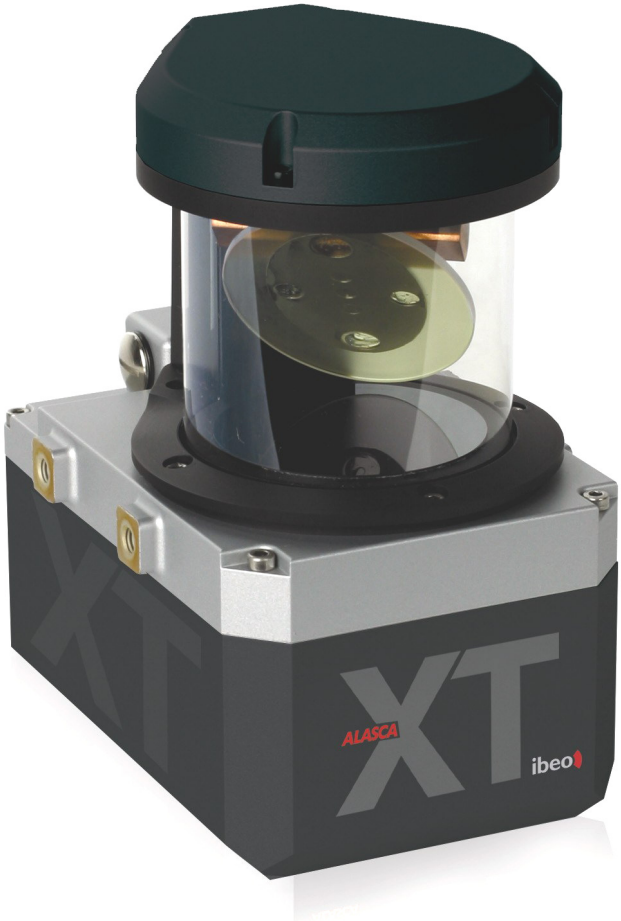


ALASCA®

User Manual

Version 2.0.0, 2006-02-15

Published by:
Ibeo Automobile Sensor GmbH
Fahrenkrön 125
22179 Hamburg
Germany

Internet: www.ibeo-as.de
Telephone: +49.40.64587-190
Fax: +49.40.64587-109

Document information
Title: ALASCA User Manual
Article number: —
Authors: J. Scholz, V. Willhoeft, Dr. R. Schulz, T. Kluge
Date: 2006-02-15
Text version: 2.0.0

Any information or suggestions for improvement are always welcome.

Table of contents

1 Introduction 1
2 Laserscanner ALASCA 2
2.1 Terminology 2
2.2 Principle of measurement 2
2.2.1 Multi-target capability 2
2.2.2 Multi-layer technology 4
2.2.3 Rotation frequency and angular resolution 6
2.3 Scan data visualisation 7
3 Handling and operating instructions 9
3.1 Eye-safety 9
3.2 Connecting the ALASCA 9
3.2.1 Power supply 10
3.2.2 ALASCA–ECU connector cable 11
3.2.3 CAN connection 12
3.2.4 RS232 connection 12
3.2.5 Ethernet connection 13
3.3 Mounting the ALASCA 13
3.4 Integration into the vehicle 14
3.4.1 Ibeo standard integration chamber 14
3.4.2 Designing a custom integration chamber 15
3.4.3 Optical window 16
3.5 Damage to the laserscanner 17
3.6 Maintenance 18
3.6.1 Cleaning the sensor 18
3.6.2 Cleaning optical parts 18
4 Electronic Control Unit (ECU) 20
4.1 Mounting the ECU 20
4.2 Connecting the ECU 21
4.2.1 Interfaces 21
4.2.2 ECU connector pinouts 22
4.3 LED signals 24
4.4 Start-up and shut-down 24
4.5 Software Updates 24
4.6 Maintenance 26
4.7 Ethernet interface 27
4.7.1 Definitions 27
4.7.2 Data Transfer 27
4.7.3 Decoding object data 28
4.7.4 Decoding scan data 28
5 SyncBox 32
5.1 Synchronisation details 33
5.2 System layout in timer mode 33
5.3 System layout in bit I/O mode 33
5.4 SyncBox LEDs and connectors 35
5.5 Signals of the SyncBox (TTL and LED) 36
5.5.1 Timing example for timer mode 37
5.5.2 Timing example for bit I/O mode 37
5.6 Setting timer or bit I/O mode 38
5.7 Interfaces and Pinouts 38
5.7.1 Power supply and bit I/O 38

5.7.2 RS232 0/1 ... 38
5.8 Electrical characteristics ... 39
5.8.1 Power supply ... 39
5.8.2 Bit I/O ... 39
5.9 Synchronisation without SyncBox ... 39
6 Object tracking ... 40
6.1 Overview ... 40
6.2 ISO 8855 coordinate system ... 40
6.3 Parameter file “AppBase.ini” ... 41
6.3.1 Syntax ... 41
6.3.2 Section “[Parameter]” ... 42
6.3.3 Section “[Sensor_*n*]” ... 44
6.3.4 Section “[RS232]” ... 44
6.3.5 Section “[Vehicle]” ... 44
6.3.6 Sections “[Velocity]” and “[SteeringAngle]” (Vehicle Data Parser) ... 45
6.4 Mounting position ... 47
6.4.1 Manual determination ... 48
6.4.2 Automatic determination ... 49
6.4.3 Vertical alignment ... 49
6.5 Vehicle Model ... 50
7 Software ... 52
7.1 Scan data pre-processing ... 52
7.1.1 Dirt detection and range estimation ... 52
7.1.2 Rain detection ... 53
7.1.3 Ground detection ... 53
7.1.4 Scan data correction ... 53
7.1.5 Scan data fusion ... 53
7.1.6 Ego-motion estimation ... 54
7.2 Object tracking ... 54
7.2.1 Segmentation ... 54
7.2.2 Contour tracking ... 54
7.2.3 Classification ... 54
7.2.4 Street detection ... 55
7.3 Applications ... 55
7.3.1 Automatic Emergency Braking ... 55
7.3.2 Other applications ... 59
8 Physical dimensions ... 60
8.1 ALASCA ... 60
8.2 Standard integration chamber ... 61
8.2.1 Housing ... 61
8.2.2 Mounting shoe (holder) ... 62
8.3 ECU ... 63
8.4 SyncBox ... 64
9 References ... 65

DISCLAIMER – PLEASE READ CAREFULLY!

THE SYSTEM AND ALL COMPONENTS, INCLUDING SOFTWARE AND THIS DESCRIPTION, HAVE BEEN MANUFACTURED WITH GREAT CARE TO ENSURE ITS PROPER FUNCTION. HOWEVER, THE SYSTEM AND OTHER HARD- AND SOFTWARE DESCRIBED IN THIS MANUAL IS CURRENTLY IN PROTOTYPE STADIUM. ALL OF THE COMPONENTS DESCRIBED HEREIN MUST BE USED IN A WAY ENSURING THAT NO HARM MAY COME TO HUMAN OPERATORS AND BYSTANDERS.

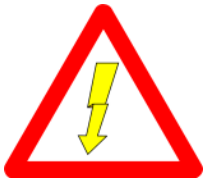
IN NO EVENT, REGARDLESS OF CAUSE, WILL IBEO AUTOMOBILE SENSOR GMBH ASSUME RESPONSIBILITY FOR OR BE LIABLE FOR INDEMNIFICATION OF THE OTHER PARTY OR FOR INDIRECT, SPECIAL: INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM PERFORMANCE OR FAILURE TO PERFORM HEREUNDER OR THE FURNISHING, PERFORMANCE OR USE OF ANY INFORMATION, PRODUCTS OR SERVICES HERETO, WHETHER DUE TO BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY OR OTHERWISE.

THE SYSTEM MUST NOT BE USED AS A LIFE-SAVING DEVICE.

Safety notification – Please read carefully!

Throughout this manual, important specifications and hints for the safety of both the human operators, bystanders and the system itself are given. In order to highlight the most important topics, these points are marked specially as shown below. **NEVER attempt to violate the rules or specifications given within this document, especially at any of these signs, as serious consequences may arise from such behaviour!**

This sign is the most important and critical warning throughout this document. It marks warnings and descriptions, which are critical for the safety of operators, bystanders or the system.



This sign warns about an electrical risk that may be present in the system. This risk may be a critical threat for the operators or the system itself.

This sign warns about a general risk or gives important information, which must be obeyed to ensure the proper operation of the system.

1 Introduction

This user manual introduces the ALASCA laserscanner. Chapter 2 gives background information to make the reader familiar with the ALASCA technology. It explains the measurement principle and introduces the visual interpretation of scan data. Chapter 3 instructs the reader how to install, handle, and operate the sensor such that safety requirements (e. g. eye-safety) will be met. Chapter 4 concentrates on the electronic control unit coming along with the ALASCA. Optional hardware components of ALASCA systems are described in chapter 5. Chapter 6 gives insight in the object tracking that the software in the electronic control unit performs based on the ALASCA's scan data. Chapter 7 focuses even more on software. It explains the software structure and gives concise descriptions of the individual software modules. Finally, chapter 8 shows the physical dimensions of the ALASCA.

Please note that this user manual cannot cover all aspects of individual configurations that are possible for different users. If in doubt the reader is encouraged to directly ask for support at Ibeo Automobile Sensor GmbH (short form: Ibeo, address see page II).

2 Laserscanner ALASCA

The ALASCA (Automotive LAserSCAnner) is a multi-layer laser-based range finding device, which measures the distances to objects in the surroundings of the sensor. The created range profiles of the different scan planes are called a “scan”. The user can configure the direction of the scan area, as well as several other parameters. The sensor is typically connected to an ECU (Electronic Control Unit) which runs the object detection, tracking and classification algorithm.

This algorithm converts the scan into a set of objects and tracks and classifies these objects. Objects have properties like size, position, velocity, type etc. They are sent from the ECU to the host computer on a CAN bus or an Ethernet interface.

2.1 Terminology

Fig. 1 visually explains the terms that are used in conjunction with the ALASCA throughout this document.

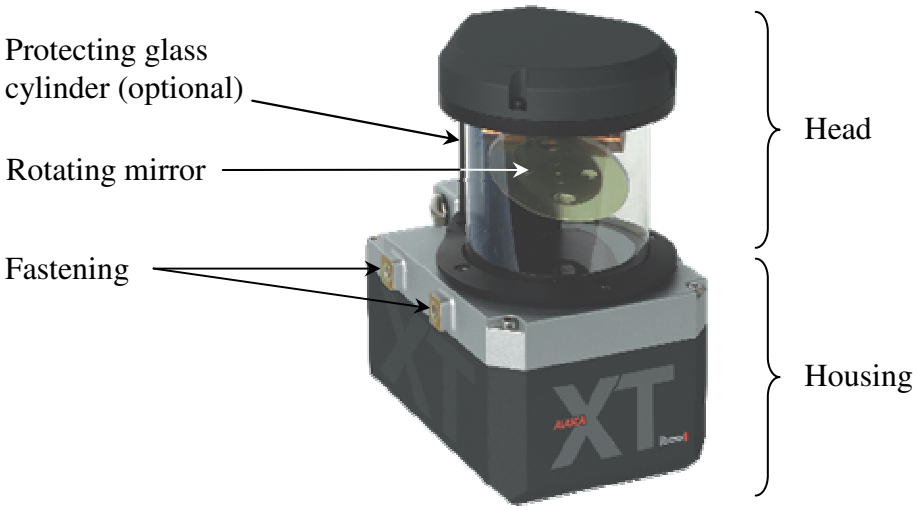


Fig. 1: ALASCA components. The protecting glass cylinder is installed only at stand-alone ALASCA sensors in place of an integration chamber.

2.2 Principle of measurement

The ALASCA laserscanner is a measuring instrument based on LIDAR technology (LIght Detection And Ranging). It scans the surroundings by means of a rotating infrared laser beam. The built-in laser transmits short rapid-fire pulses that are reflected by objects in the surroundings (Fig. 2). The laserscanner can detect the reflections, which allows for a measurement of the pulses’ times of flight. From these times and the velocity of light, the distances to the objects can be determined. In parallel, the direction to each object is known from the angular position of the rotating mirror that deflects the laser beam. Fig. 3 shows the main components inside the ALASCA that are involved in the measurement.

2.2.1 Multi-target capability

When a reflected pulse reaches the photo diode receiver, the received intensity is converted to a voltage. The reflection will be detected if its voltage exceeds a given threshold. This threshold prevents system noise from being detected as false objects.

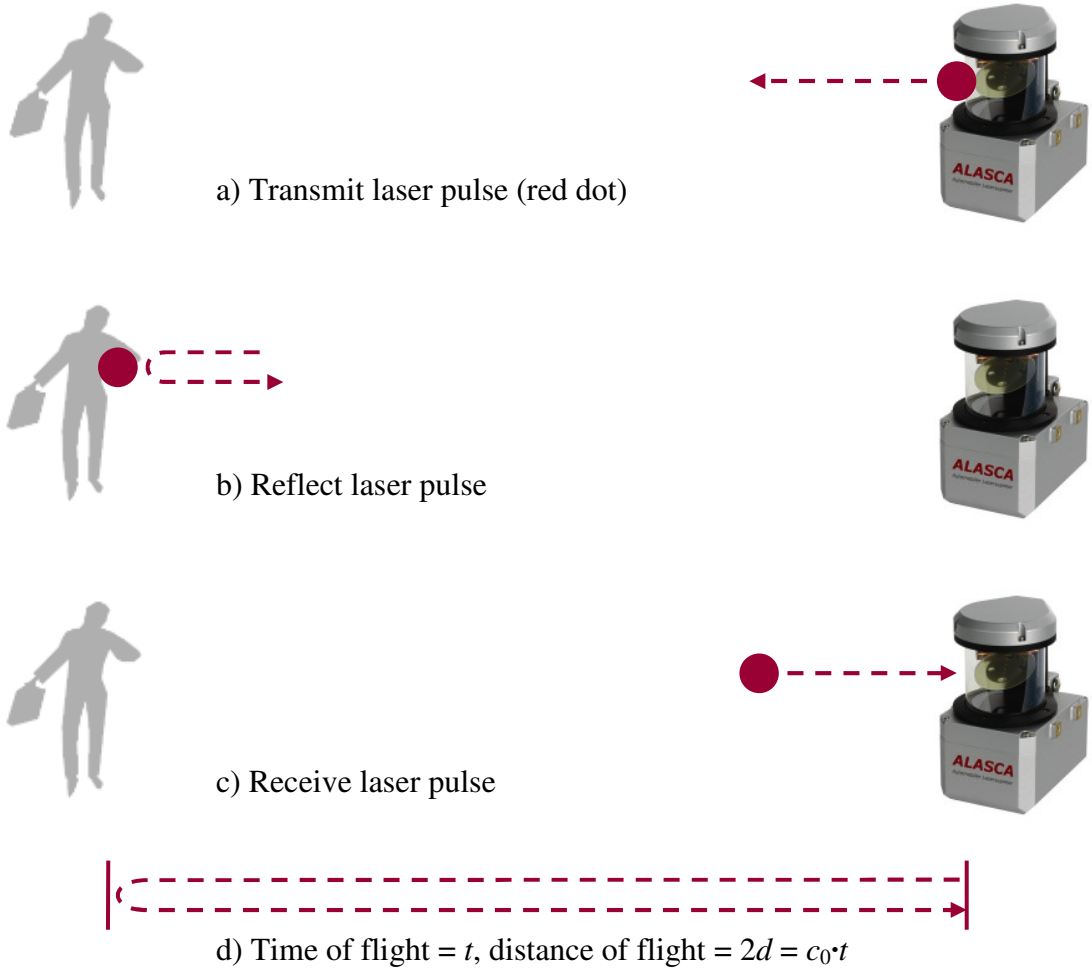


Fig. 2: Principle of time-of-flight measurements: The distance d to an object can be determined from the laser pulse's time of flight t and the velocity of light c_0.

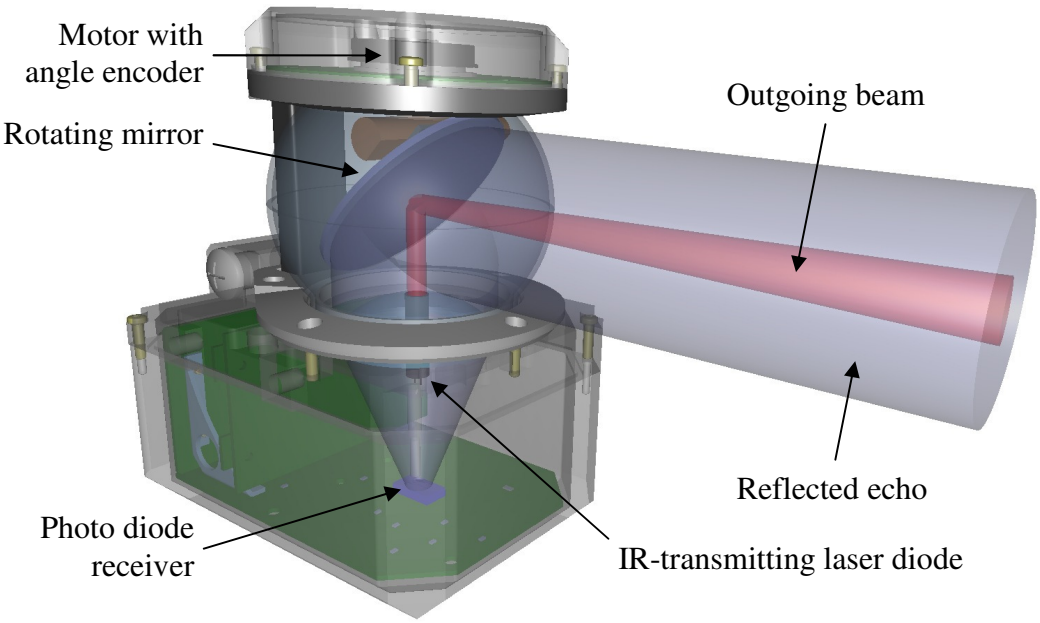


Fig. 3: Insight in ALASCA's internals

ALASCA's receiver electronics can detect up to four echoes per transmitted laser pulse, e. g. echo A coming from the pane of the ALASCA's integration chamber, echo B from a rain drop in front of the laserscanner, echo C coming from an object farther away, and echo D missing because of an opaque object at echo C. This feature is called *multi-target capability* (Fig. 4). It becomes mandatory if the ALASCA is integrated into a vehicle.

2.2.2 Multi-layer technology

The ALASCA supports four scan planes with different vertical angles. This so called *multi-layer technology* is mandatory for automotive applications as Fig. 5 illustrates.

How does the multi-layer technology work? In detail, the photo diode receiver shown in Fig. 3 is composed of a linear array of four independent receivers. Each receiver corresponds to one layer (also called *channel*). Seen through the sensor's optical components each channel has its own field of view (Fig. 6).

Since the photo diode receiver remains stationary while the mirror revolves, the fields of view are rotated. Fig. 8 shows the rotated fields of view for some exemplary mirror directions α. When α runs over a full revolution the midpoint of each field of view moves on a certain cosine-like curve (Fig. 9 and Fig. 10).

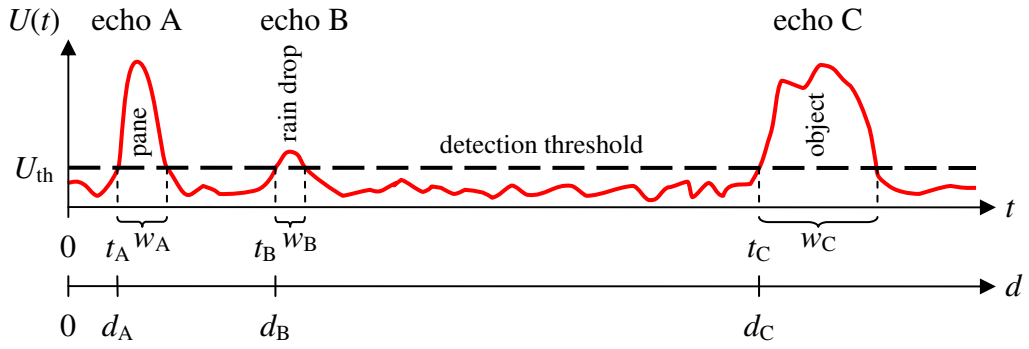


Fig. 4: Exemplary output voltage $U(t)$ of the photo diode receiver. The laser pulse was transmitted at $t = 0$ and afterwards reflected somewhere. The threshold voltage U_{th} separates system noise from the relevant laser pulse echoes. The velocity of light c_0 relates time t and distance d: $2d = c_0 \cdot t$. The echo pulse widths $w_{A/B/C}$ are also measured by the ALASCA. The echo D is not shown here.

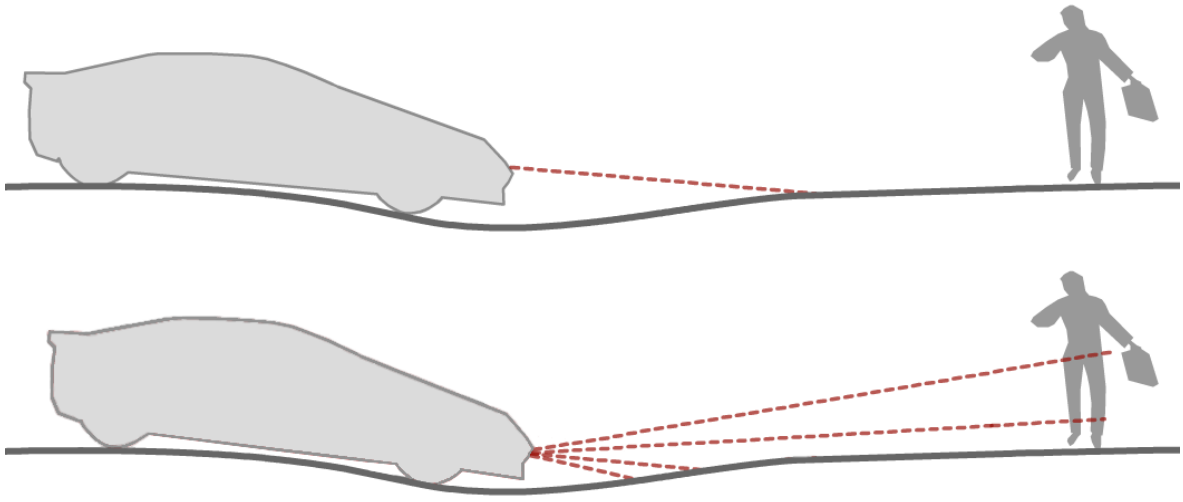


Fig. 5: Laserscanners with only one scan plane are not suitable for automotive applications because tracked objects may get lost due to pitch movements as the upper figure illustrates. ALASCA's multi-layer technology allows for pitch angle compensation by means of four scan planes at different vertical angles. For clarity, the vertical divergence is exaggerated here.

The following table explains the naming convention for the channels and shows their default colours used for visualisation. Channels are visualised by hue, whereas echoes are visualised by saturation.

	Echo Pulse			
	1st	2nd	3rd	4th
Yellow channel (top)	**4A**	**4B**	**4C**	**4D**
Green channel	**3A**	**3B**	**3C**	**3D**
Blue channel	**2A**	**2B**	**2C**	**2D**
Red channel (bottom)	**1A**	**1B**	**1C**	**1D**

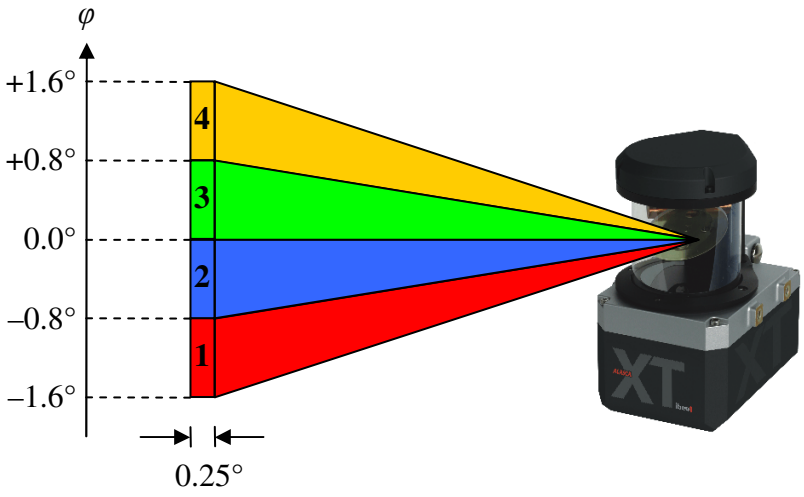


Fig. 6 shows the fields of view of ALASCA's four channels. The cross-section shown at the left end has the original aspect .

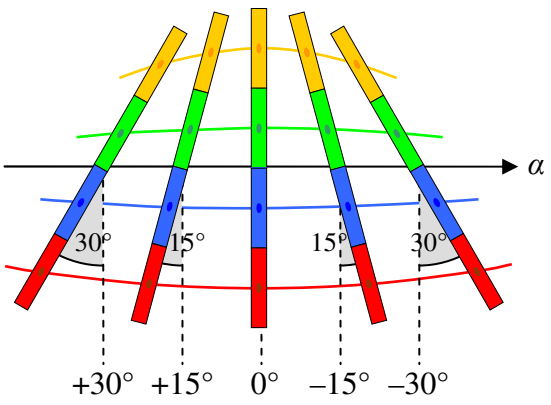


Fig. 8: The four fields of view are tilted when the mirror looks to different directions *α*. Both, the tilt angle and the mirror's direction always have the same angle *α*.

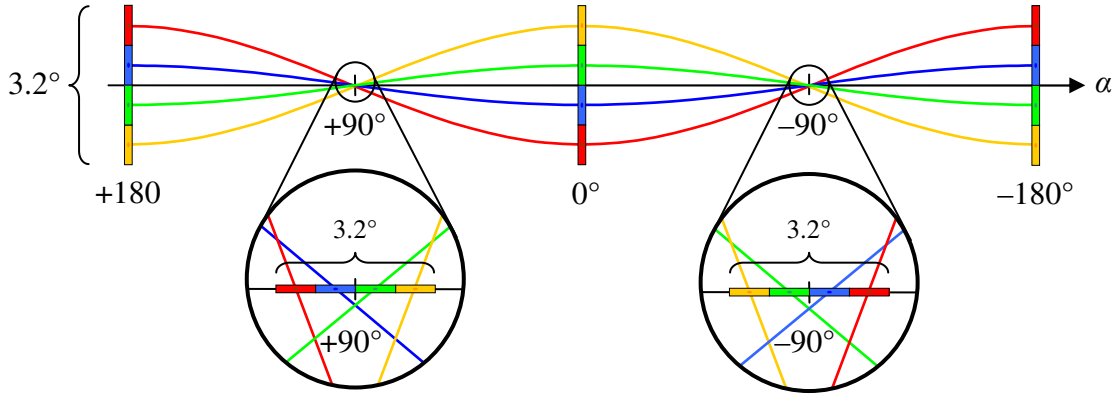


Fig. 9: The same view as before but extended to one full revolution of the mirror. The vertical axis is exaggerated for clarity. The curves intersect the vertical axis at ±0.4° and ±1.2°, whereas the horizontal axis is intersected at ±90° ± 0.4° and ±90° ± 1.2°. Though the curves resemble the graph of the cosine function, they are in fact shortened cycloids.

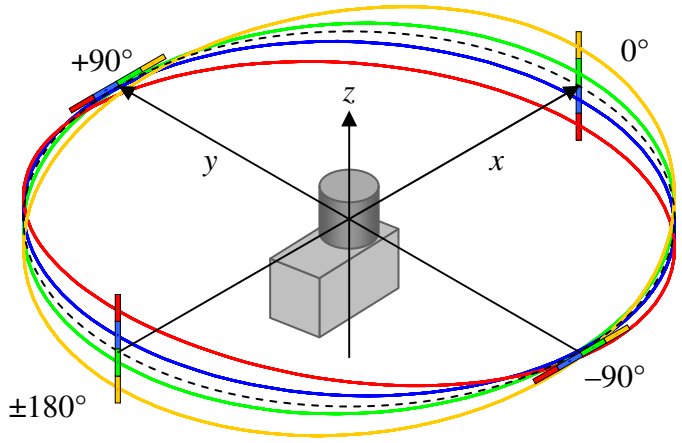


Fig. 10: The three-dimensional pendant to the previous graph. The centre of ALASCA's mirror forms the origin of the coordinate system. Again the *z*-axis is exaggerated. For details about ALASCA's coordinate system, see chapter 6.2.

2.2.3 Rotation frequency and angular resolution

The motor that drives the deflection mirror is optimised for rotation frequencies of 12.5 and 25 Hz, though it also supports any other frequency in the range from 8 to 40 Hz. This frequency is directly related to the available angular resolution as shown in the following table. The relation arises from considerations to ensure eye-safety and to protect the laser from overheating.

Rotation frequency f	Angular resolution
8.0 Hz $\leq f \leq$ 12.5 Hz	$\geq$ 0.25° continuous, 0.125° short term
12.5 Hz $< f \leq$ 25.0 Hz	$\geq$ 0.50° continuous, 0.250° short term
25.0 Hz $< f \leq$ 40.0 Hz	$\geq$ 1.00° continuous, 0.500° short term

The restriction “short term” in the table means that the high angular resolution is available only for a sequence of a few laser shots. In automotive applications, this sequence is typically located in driving direction to have a high-resolution scan of the oncoming objects.

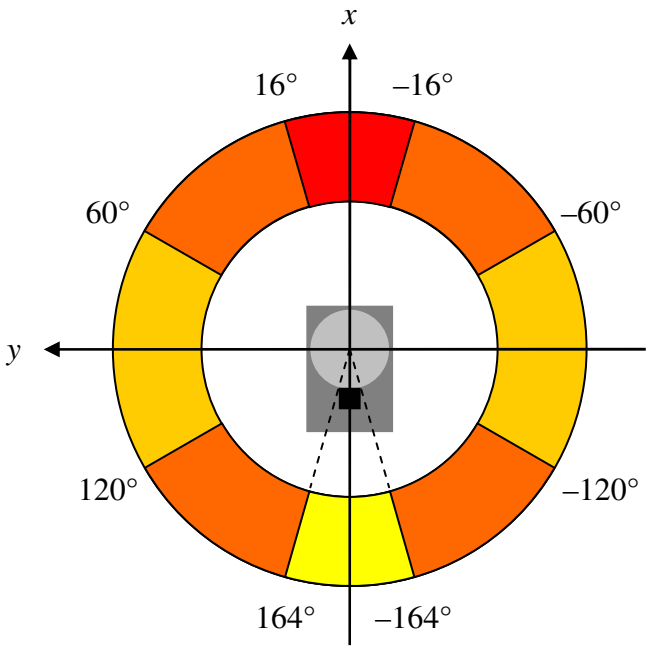
In automotive applications, there are some directions (e.g. ahead) more relevant than others (e.g. lateral). An application can put its focus on the relevant part(s) of the surroundings by using high-resolution scan data there and low-resolution data in the less relevant parts. The ALASCA XT supports *flexible angular resolution* to meet this requirement (Fig. 11).

The default setting of the flexible angular resolution puts focus especially on a range of ±16° around the driving direction (which is defined as the *x*-axis in automotive applications). Here, the short-term high resolution is used, whereas the continuous high resolution is used in a wider range of ±60° around the *x*-axis (also in backward direction). Within these ranges, most of the relevant traffic objects can be detected at a good resolution.

The lateral range around the *y*-axis has a lower resolution. Finally, the small angular range around the 180° boundary has a very low resolution. The strut that holds motor and mirror obstructs the view of the laserscanner in this range.

The resolution values shown in Fig. 11 are valid only for low rotation frequencies up to 12.5 Hz. As listed in the previous table, all resolution values are multiplied by 2 or by 4 at higher rotation frequencies so that resolution gets coarser.

Currently there is no user interface to change the default settings of the flexible angular resolution by the customer. Upon request, Ibeo can handle this task for the customer. Note that the laserscanner must be sent back to Ibeo for this purpose.



Resolution	
12.5 Hz	**25 Hz**
0.125°	0.25°
0.25°	0.5°
0.5°	1°
1°	2°

Fig. 11: The ALASCA XT supports a flexible angular resolution. The resolution depends on the relevance of each angular range for automotive applications. For instance, oncoming objects have a high relevance (red area), whereas lateral objects are usually less relevant. The yellow low-resolution area is obstructed by the strut that holds motor and mirror (small black square).

2.3 Scan data visualisation

As shown in the previous section the reflection of a single laser pulse can be converted to a point *P* given in polar coordinates (d, α, φ) relative to the sensor where

d = distance,
α = yaw angle / azimuth angle / horizontal angle,
φ = pitch angle / elevation angle / vertical angle.

Such a single point is called a *scan point*. During scan data processing, each scan point is converted to Cartesian coordinates $P = (x, y, z)$.

During one revolution of the mirror a multitude of laser pulses is transmitted resulting in a set of scan points $\{P_1, \ldots, P_n\}$, the so called *scan*. The number of detected scan points n may vary from one scan to the next, e. g. due to object movements in the surroundings.

For visualisation, the scan points are usually plotted in a rectangular two-dimensional (2D) coordinate system, the *scan view*, but they can also be overlaid to a video image projected in perspective. Fig. 12 shows an example of both visualisations for the same scene. When comparing the visualisations please keep the following items in mind:

- The scan view shows the scan in bird's-eye view whereas the video image shows the scan in camera (or human's-eye) view. In Fig. 12, exemplary groups of corresponding scan points are encircled pink and connected by arrows.
- The camera has a view angle that is much less than the laserscanner's maximum view angle. Thus not all scan points in the scan view have a corresponding point in the camera view. In Fig. 12 only those scan points can be found in both visualisations that lie between the lines labelled with ❷ and ❸.
- Like the human eye, a laserscanner neither looks through (infrared) opaque objects nor looks round the corner. Therefore, cars, trucks, and busses generally appear L-shaped in the scan because a single sensor can see at most two edges of a vehicle's outline at the same time.
- The colours of the scan points refer to ALASCA's multi-target and multi-layer capabilities, see chapter 2.2.

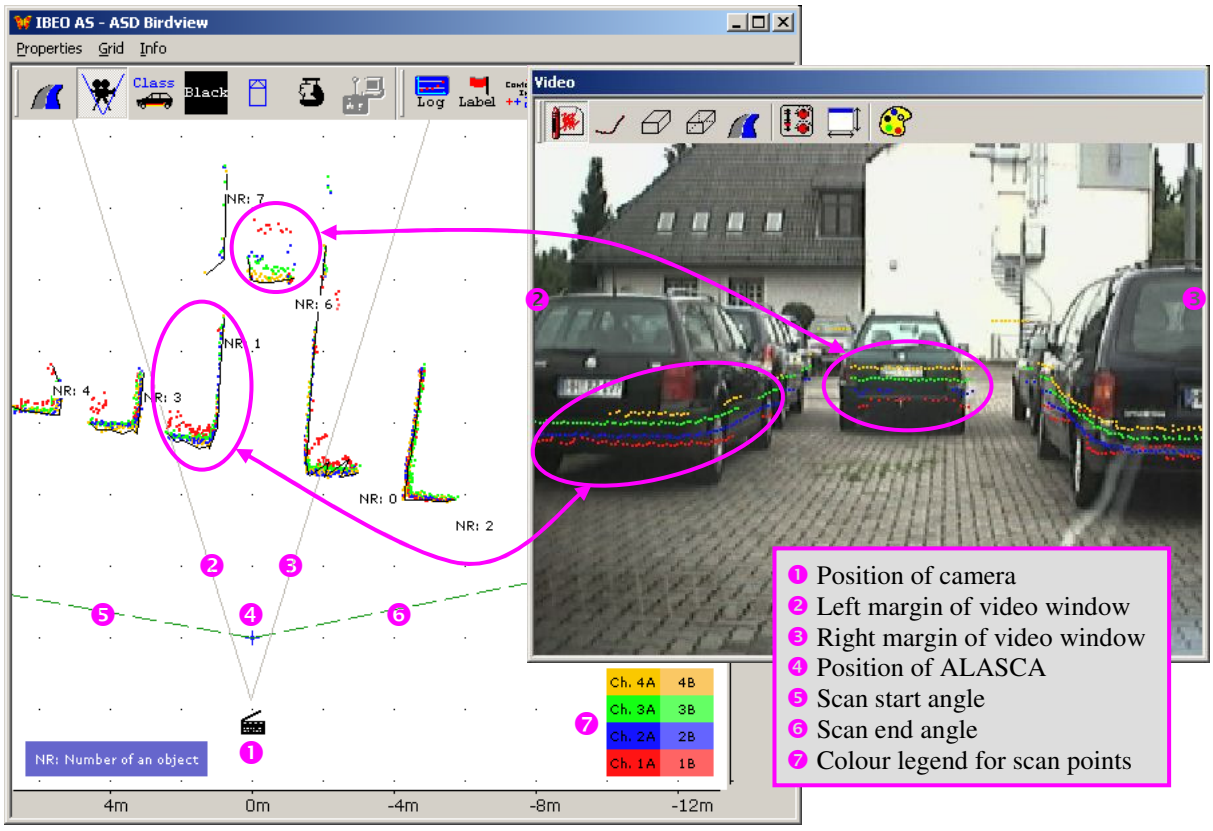


Fig. 12: Relation between the scan view (left window, bird's eye view) and the corresponding video view (right window). The video view shows only a small part of the wider scan view; the angular range is 35° (❷, ❸) out of 160° (❺, ❻). Pink annotations are overlaid for clarity; they are not part of the views.

3 Handling and operating instructions

The ALASCA is a highly sensitive optoelectronic device. Although it is designed for automotive applications, the device should be handled with care to avoid damages. Moreover, the built-in laser may cause harm due to invisible laser radiation if it is not handled as directed. This chapter gives instructions how to handle the sensor and how to operate it in an eye-safe way.

3.1 Eye-safety

The ALASCA complies with the laser class 1 requirements of the European laser standard EN 60825-1:1994 + A11:1996 + A2:2001, see [1].

The laserscanner is equipped with a scanning safeguard that shuts down the laser emission in case of failure of the scanning mechanism.

Do only use the laserscanner as directed. Do not open the housing of the sensor and do not dismount the head of the sensor because then the laserscanner will no longer be eye-safe and may cause harm due to invisible laser radiation!

3.2 Connecting the ALASCA

The ALASCA has one connector that includes both the power supply and the signal interfaces. Fig. 13 shows how to connect the ALASCA to the other system components. The individual connections are described in the following subsections. The ECU (Electronic Control Unit) is explained in chapter 4.

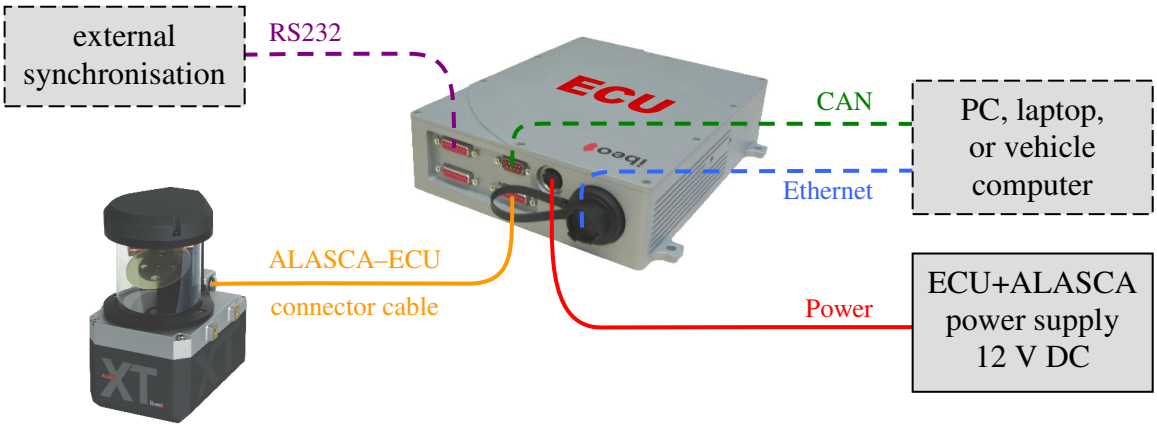


Fig. 13: Connecting the ALASCA system (dashed lines: optional components)

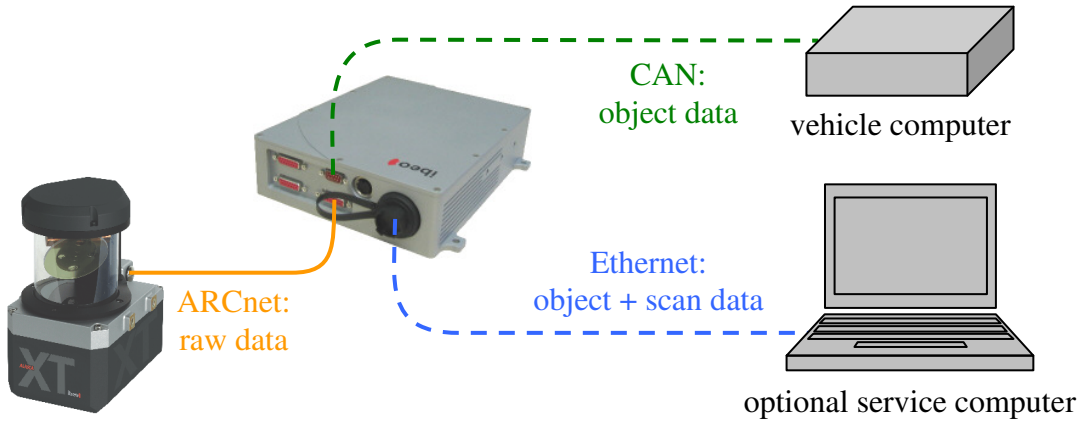


Fig. 14: Types of data busses in the ALASCA system and the corresponding kind of data

As shown in Fig. 14, each connection transmits a certain kind of data:

ALASCA–ECU connector cable, raw data: Here the ALASCA sensor sends its hardware measurement data to the ECU. This data is called *raw data* because it needs further processing by the ECU. Raw data is transmitted from the ALASCA to the ECU only, not to the vehicle computer. This connection uses an ARCnet bus for data transmission.

CAN, object data: A condensed high-level description of the scan data is sent as *object data* on the CAN bus to the vehicle computer.

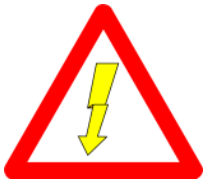
Ethernet, object and scan data: While processing in the ECU the raw data becomes *scan data*, i. e. a collection of scan points (see section 2.3). The optional Ethernet connection transmits scan data for debug or visualisation purposes. In addition to the CAN bus, object data is sent over the Ethernet interface, too.

Usually scan data is of interest for humans only. Humans can intuitively group, interpret, and assess scan data in the context of a traffic situation. A standard vehicle computer does not have this intelligence. Therefore, the ECU acts as a pre-processor for the vehicle computer. Software algorithms on the ECU transform low-level scan data to high-level object data like “object no. 4 is a car 30 metres ahead driving at 50 km/h” (see also chapter 6.1).

3.2.1 Power supply

The ECU requires a 12 V DC, 20 W power supply. Typical power consumption of the stand-alone ECU is less than 12 W. **Note that the ECU also supplies power to the connected sensor(s), which will increase the current on this power connector (approx. 12 W per sensor).**

The system may be directly connected to the vehicle power supply of 12 V nets. In order to avoid damage due to overvoltage, an additional DC/DC converter can be inserted. A suitable DC/DC converter is available from Ibeo. In order to ensure the proper function of the system, do not use this converter to supply power to other devices, too.



The supply voltage is 12 to 15 V DC.
Never operate the system outside this voltage interval!

Do not reverse polarity!

3.2.2 ALASCA–ECU connector cable

The connector cable is used for communication between ALASCA and ECU. It includes power supply for the sensor, an ARCnet bus for data transmission, and a RS232 interface for sensor synchronisation. The latter one is described separately in chapter 3.2.4.

For ALASCA sensors, a power supply only between 12 and 15 V DC may be used. The standard connector cable delivered by Ibeo directly passes the external ECU power supply on to the sensor(s). Inside the ECU, there is no special power supply unit for the sensor.

The ARCnet bus in the connector cable is used to send commands from the ECU to the sensor and to receive raw data from the sensor at the ECU. ARCnet is a bus system, which must be terminated at both ends of the bus. Termination reduces reflections on the cable that disturb the signals. The ARCnet bus system is internally not terminated to allow connecting multiple sensors to a single bus, but it must be terminated at both ends of the bus (see Fig. 15). The standard ALASCA–ECU connector cable delivered by Ibeo is correctly terminated.

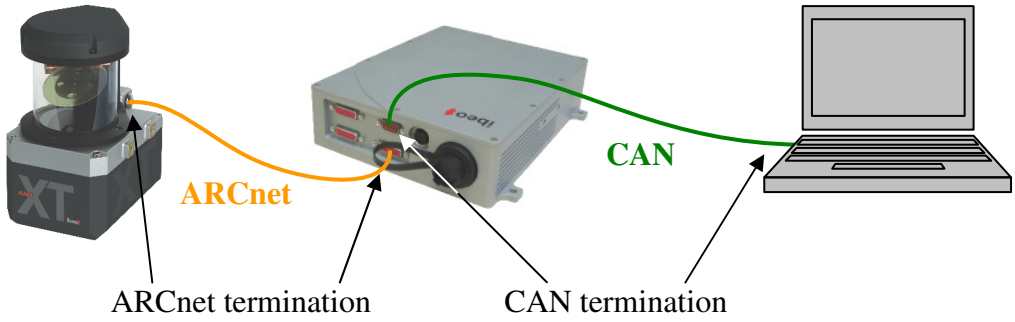


Fig. 15: ARCnet and CAN termination for a standard ALASCA system

The following table shows the pinout of ALASCA's 15-pin connector:

Pin	Description
1	ARCnet in "–"
2	ARCnet out "–"
3	Signal ground for all signals
5	RS232 RX (receive)
6	Power supply: +12 V DC
7	Power supply: +12 V DC
9	ARCnet in "+"
10	ARCnet out "+"
12	RS232 TX (transmit)
13	Power ground (= Signal GND)
14	Power ground (= Signal GND)

Note: Do not connect the unused pins that are not listed in this table.

For self-made connector cables, please take care of the following ARCnet items:

- The ARCnet bus is not terminated in the ALASCA sensor. Instead, the end of the ARCnet cable must be terminated.
- Signal cables should be as short as possible to avoid distortions. Do not exceed the maximum cable lengths specified by the interface standard.
- Never create cable stubs ("T-branches").

- ARCnet transmission runs at 10 MBit/s. Data transmission will become unstable and produce strange errors if low-quality cables are used. Ibeo recommends high-quality twisted pair cables with 120 Ω characteristic impedance. The pins *ARCnet in* "–" and *out* "–" should be connected to one twisted pair; *ARCnet in* "+" and *out* "+" to another twisted pair.
- Please also refer to the separate manual "ARCnet documentation" [2].

3.2.3 CAN connection

The CAN connection transfers data between the ECU and the vehicle computer:

Object data	ECU → vehicle computer
Vehicle data	ECU ← vehicle computer
Application data	ECU ↔ vehicle computer
Parameter data	ECU ↔ vehicle computer

CAN is a bus system that must be terminated at both ends of the bus (see Fig. 15). The terminator must be integrated into the cable or an external terminating adapter plug must be used (available from Ibeo). For details on installing and using the CAN bus see the separate CAN hardware and user manual "CAN documentation".

Each message sent on the CAN bus is labelled with a numerical message identifier (ID). This numerical value also implies the message priority where ID 0 means the highest priority. The CAN messages sent by Ibeo use only a few identifiers (see also parameter **CANBaseID** in section 6.3.2):

- one identifier for communication from the vehicle computer to the ECU,
- another identifier for the opposite direction of communication, and
- some identifiers for high-priority communication (e. g. automatic emergency braking).

To distinguish between messages sent with the same identifier, each message type is individually identified by a *message type* value inside the message.

The ECU also includes state information into a certain CAN message so that the vehicle computer can verify the state of the ALASCA system. The state information comes along with the so-called *list end* CAN message.

A complete reference of all Ibeo CAN messages can be found in the separate manual "CAN message protocol" [3].

3.2.4 RS232 connection

The RS232 connection at the ECU is used to synchronise ALASCAs and other sensors like a video camera. A periodical signal is sent on the RS232 interface to control synchronisation. Synchronised sensors look (as good as possible) straight ahead to the 0° direction each time the synchronisation signal is received. For this purpose, a fine-tuning motor speed controller is integrated into the scanner.

Note: ALASCA users with an IPC (Industrial PC, predecessor of the ECU) have no direct access to RS232 when using standard connector cables. In this case, the only way to access RS232 is to modify the ALASCA connection cable and add a RS232 connector to the cable. The pinout of the ALASCA connector is described in the table on page 11.

The RS232 connection must be set-up using the following parameters:

RS232 Parameter	Value
Baud rate	57,600 bits/s
Data bits	8
Parity	no
Stop bits	1
Handshake	no

Ibeo offers an optional hardware interface for synchronisation, the so-called *SyncBox*. For details, please refer to chapter 5.

3.2.5 Ethernet connection

The Ethernet interface connects the ECU to the local network or to a service laptop. Due to the high bandwidth of Ethernet, it is possible to visualise both scan and object data with the ASD software from Ibeo. Please refer to chapter 4.7 for details.

3.3 Mounting the ALASCA

The ALASCA should be mounted at the front of the vehicle, with a clear field of view as wide as possible. In addition, some general rules should be obeyed:

- Do not obstruct the field of view of the ALASCA. If you mount the ALASCA behind a pane (glass or acrylic plastic), make sure that the pane is transparent for infrared light (see chapter 3.4).
- Protect ALASCA's head and the interface connector from accidental damage, e.g. due to collisions. Integrate the sensor into the vehicle (as described in chapter 3.4) or use simple metal barriers to avoid damage to the sensor. The opposite figure shows an example of such a metal barrier for ALASCA's predecessor sensor LD ML. When driving on public roads please note that there may be national statutory regulations that refer to such barriers, and other restrictions may apply.

- As Fig. 6 shows, ALASCA's lower ("red") channel measures the ground at an incident angle of –1.6° (looking ahead to 0° direction). Depending on the application, the sensor should be mounted at a height suitable for the wanted range of the lower layer. The height refers to the distance of the mirror centre above the ground (see also Fig. 41).
- The ALASCA needs a sturdy mechanical mount in order to attach it to the vehicle body. This mount should allow for a vertical adjustment of about 5° to 10°. With this range, the customer will be able to adjust the proper vertical zero-degree line. Horizontal adjustment is not required if the design warrants a maximum horizontal angular error less than about 2°. Although it is possible to compensate for arbitrary angular errors by software (cf. chapter 6.4), you should keep in mind that the divergence of the scan layers is maximal only in the sensor's 0° direction (see Fig. 10). Thus significant differences between the 0° direction of the vehicle and the sensor will result in a reduced capability to compensate pitch angles.
- The ALASCA may also be mounted upside down. In this case, the processing and visualisation must be informed that the sensor is upside down, because the location of the vertical scan planes and the scan direction is reversed; see parameter **SensorUpsideDown** in section 6.3.2.

3.4 Integration into the vehicle

For professional automotive applications, the laserscanner should be integrated into the vehicle. Fig. 16 shows the integration at the front centre of a test vehicle. Other integration positions may be located at the front left and right corner or the rear of the vehicle.

The following subsections describe the standard integration chamber offered by Ibeo and give guidelines for custom designed integration chambers.

3.4.1 Ibeo standard integration chamber

Ibeo offers a standard integration chamber to customers that do not need a perfect integration into the test vehicle body. Fig. 18 shows this housing. The housing consists of the chamber and the adjustable mounting shoe. Here only the vehicle connection has to be carried out by the customer. Chapter 8.2 shows outline drawings and measures of the standard integration chamber.

The mounting shoe has six adjustment screws (see red arrows in Fig. 18). They allow for small pitch angle corrections within the range of the flexible rubber seal. **Please do not loosen any other than these six screws. The warranty is void if the seal of any screw is broken!** Moreover, the mounting shoe has six bolt holes (Fig. 18). A custom designed adaptor can be fastened here to connect the integration chamber to the vehicle body.

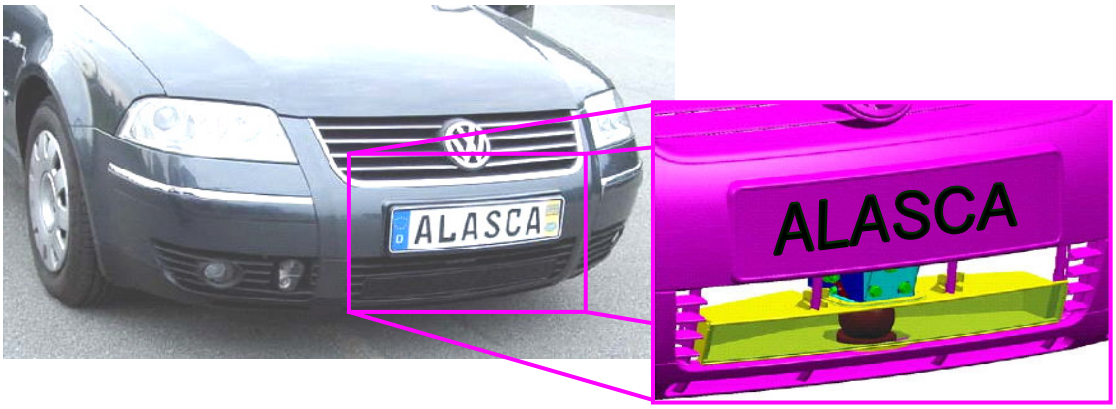


Fig. 16: Example for an integration of the ALASCA into an Ibeo test vehicle.
In this case, the ALASCA is mounted overhead (housing: light blue, head: brown).

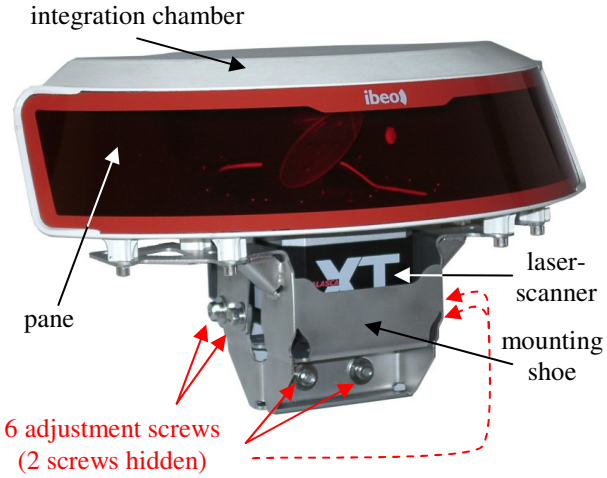


Fig. 18: Ibeo standard integration chamber
(to be integrated at the front centre of the vehicle)

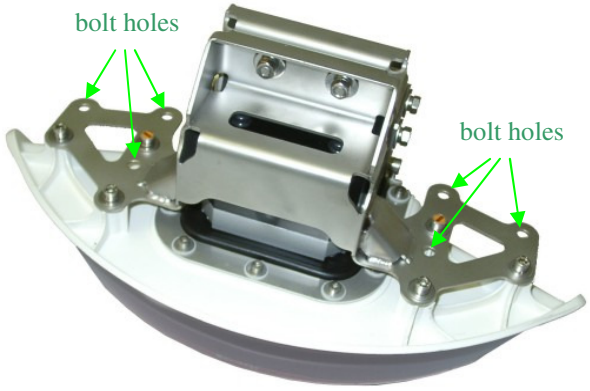


Fig. 18 shows the Ibeo standard integration chamber from below. The green arrows point to six holes in the mounting shoe that can be used to fasten custom designed adaptor, which connect the integration chamber to the car body.

For transportation safety, Ibeo delivers each integration chamber with a protective plastic film on the pane. Please remove this plastic film before using the laserscanner for the first time. Otherwise, the laserscanner will show bad performance!

Integration chambers from Ibeo are delivered with a waterproof built-in laserscanner. Neither open the integration chamber nor remove the built-in laserscanner as this will result in a leakage that may damage the laserscanner! Do not drill holes in the integration chamber.

3.4.2 Designing a custom integration chamber

The integration chamber can be designed by Ibeo (based on custom requirements) or by the customer alone. In the latter case some design rules should be observed that are described in this section. Please contact Ibeo if consulting on this topic is desired.

First, determine the optimum position of the ALASCA in the test vehicle regarding your application. Note that the ALASCA main axis must aim at that direction where the full 3.2° vertical divergence is required (cf. Fig. 10). Define the horizontal field of vision (scan range) and conclude the requirements on the dimensions of the integration chamber. Due to the vertical divergence of the field of view, consider 6 cm free field of view for the vertical extension of the optical window as illustrated in Fig. 19. The optical window should be vertically tilted at no less than 22° (see also next section for details).

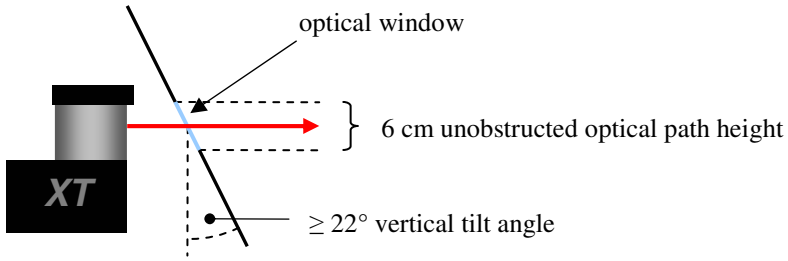


Fig. 19: Illustration of the unobstructed optical path required in the vertical direction

As true for any optical device, the ALASCA must work in a clean and dry environment. The automotive integration chamber must be designed to provide this function. At least the upper part of the ALASCA (scanning part with mirror and scan-motor unit) must be contained within this chamber in order to provide dust-free and moisture-free operation. The lower part of the ALASCA is already designed waterproof and equipped with a Gore-Tex® breather. Upon customer request, the ALASCA may be equipped with a flexible seal at the interface between its lower and upper part (see Fig. 20) which allows a waterproof connection of the integration chamber to the ALASCA scanning unit.

The integration chamber should also be equipped with a Gore-Tex® breather in order to avoid the intake of water during changes of air pressure. The breather must be located in a position where water cannot accumulate.

It is extremely important for the functioning of the ALASCA that the inner of the integration chamber is finished with a black matt coating in order to minimise unwanted internal reflections.

The connection of the ALASCA to the vehicle body must withstand automotive requirements and should therefore be carried out with a sturdy design. A mounting shoe should be designed permitting pitch-angle adjustment. Fig. 20 shows an example of mounting and integrating the ALASCA.

In addition, a specific mounting shoe is available from Ibeo upon customer request. The mounting shoe in combination with a vehicle-connector fixes the ALASCA against the vehicle body. The shape of the connector depends on the specific integration situation and must therefore be custom designed.

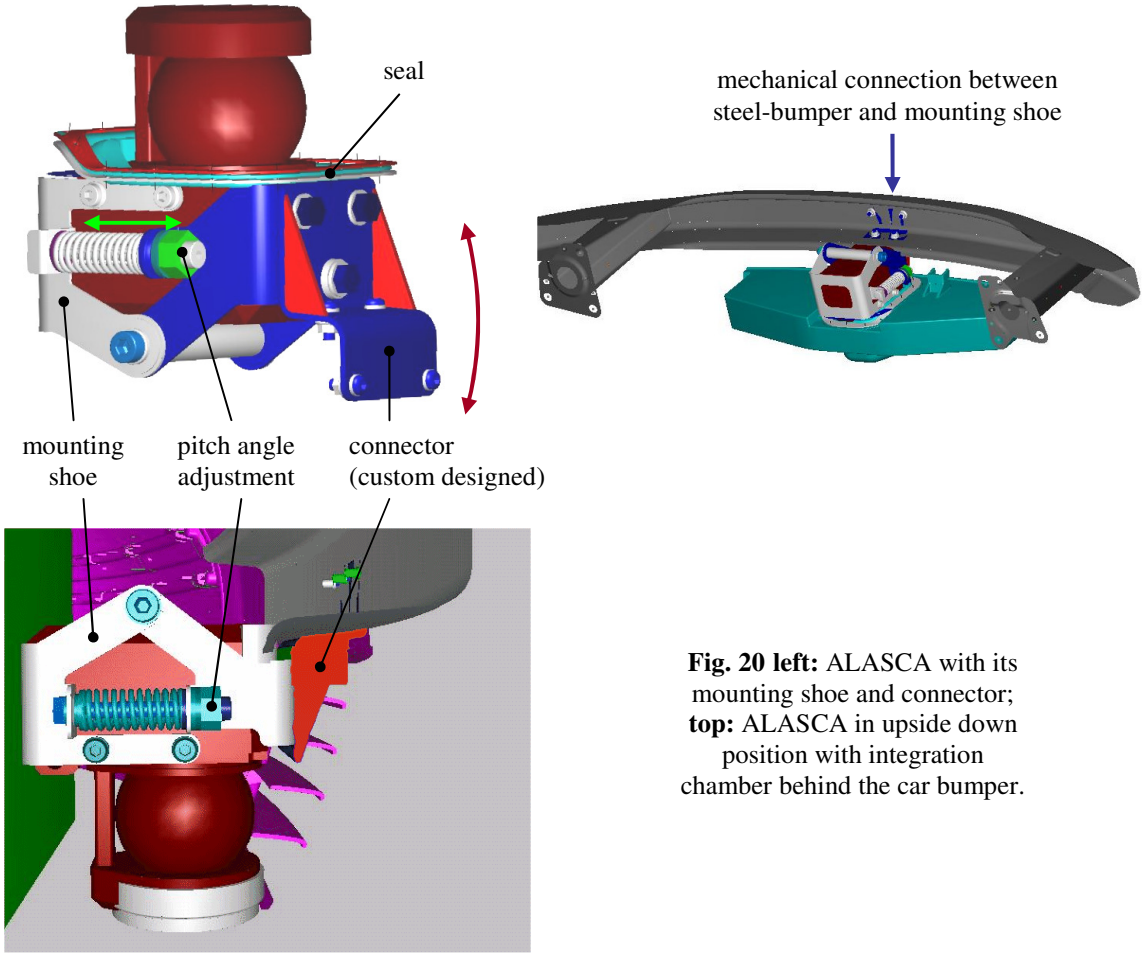


Fig. 20 left: ALASCA with its mounting shoe and connector; **top:** ALASCA in upside down position with integration chamber behind the car bumper.

3.4.3 Optical window

For the optical window of the sensor integration chamber, Ibeo recommends the infrared-transmissive *LUXACRYL-IR*, Type 1698, thickness 1.5 mm for large bend-radii, as illustrated in Fig. 21. Smaller bend-radii of the window may be necessary, i. e. for front-bumper-integration in trucks for applications like turning assistant, which require a scan angle of up to 240°. In this case, optical reasons require a reduced thickness of down to 0.5 mm for a bend radius of about 100 mm.

For purchasing the optical window, please check the TTV web pages: www.go-ttv.com/filter/filters.htm. Other colours than IR-black, except for grey, may be chosen if required for design reasons. To increase durability of the window the outer surface may be coated with a

hard coating. This coating must be professionally applied in a clean environment. For a first go, however, the plastic material may be used uncoated.

The optical window should be formed from a plane sheet of plastic. The plastic window can be bent horizontally according to the requirements of the integration chamber. The plastic window must be made from one piece without any parts in the way, e.g. heating wires. For optical reasons, one should refrain from a full three-dimensional forming of the plastic material surface because this leads to undesirable optical distortions.

Also for optical reasons, the window must not be positioned perpendicularly to the direction of the laser beam. This would lead to an optical short circuit due to direct reflections and it significantly reduces the sensitivity of the ALASCA. It must be ensured that for all scan angles the laser beam reflected back from the optical window would not re-enter the optical system of the ALASCA.

For illustration purposes, Fig. 21 shows the top view onto a left-side integration chamber. Here laser beam 1 is drawn at the critical scan angle range around 45°. At this horizontal angle, only a vertical tilt of the window can prevent the internal reflection to directly re-enter the ALASCA. For other scan angles, the reflection from the window is unable to re-enter the optical system of the ALASCA because of the large horizontal incidence angle.

In terms of numbers, the vertical tilt angle of the optical window (Fig. 19) must measure a minimum of 22° at that point where the laser beam hits the window perpendicularly (see Fig. 21, beam 1). For design reasons, it is possible to steadily reduce the vertical tilt angle from 22° (at perpendicular horizontal incidence) down to 0° on each end of the window.

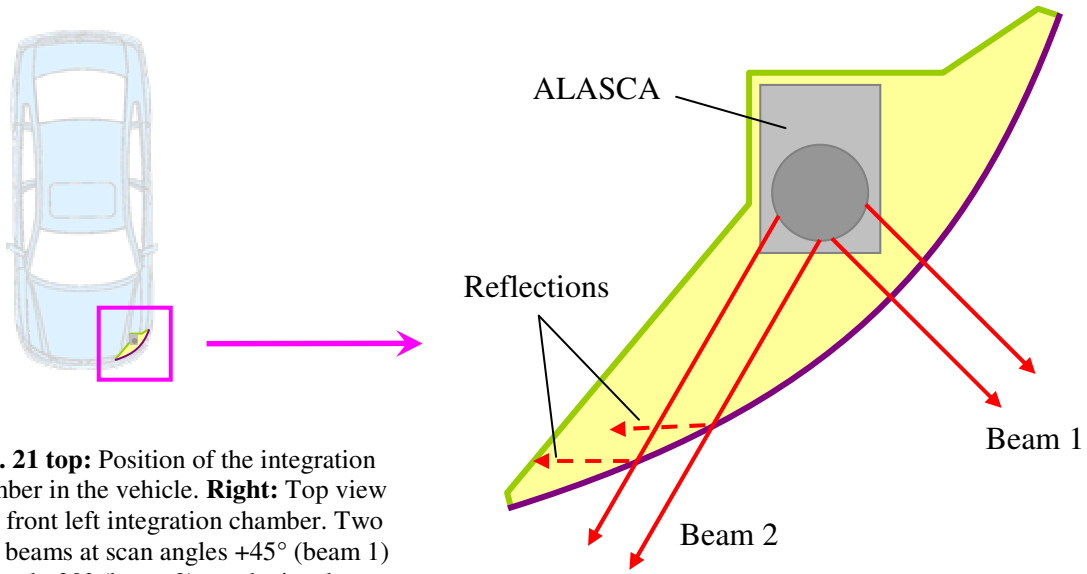


Fig. 21 top: Position of the integration chamber in the vehicle. **Right:** Top view on a front left integration chamber. Two laser beams at scan angles +45° (beam 1) and –30° (beam 2) are depicted.

3.5 Damage to the laserscanner

If the laserscanner becomes damaged it is necessary to stop working with the system immediately. In this case, disconnect the laserscanner from power to avoid further damage to the system. Please send the laserscanner back to Ibeo to have your ALASCA repaired (address see page II).

The customer must not open the ALASCA. Only use the laser-scanner as directed. Otherwise, the laserscanner may cause harm due to high voltage or invisible laser radiation.

3.6 Maintenance

In general, the ALASCA does not require maintenance. However, the items listed in the following subsections should be checked regularly.

3.6.1 Cleaning the sensor

Although the sensor is protected against water, it must not be cleaned using a jet or steam jet cleaner as water may penetrate the seals, or parts of the sensor housing may get damaged. Use a soft cloth with water and a non-aggressive and non-abrasive cleaner instead.

The sensor does not contain user serviceable parts inside. Its housing is sealed watertight to protect it from environmental influences. Do not open the sensor housing!

Do not use a jet cleaner or a steam jet cleaner to clean the sensor!

Do not clean the vehicle in a car wash when the sensor is mounted outside the vehicle (not protected by an integration chamber)!

3.6.2 Cleaning optical parts

This section applies only to special “naked” ALASCA sensors, which are not mounted inside an integration chamber and have no protecting glass cylinder. In this case, optical parts of the sensor may become dirty, for instance by remains of dust or fingerprints. The optical parts to clean are

- the deflection mirror and
- the circular pane below the mirror.

These optical parts should be cleaned regularly using an optical cleaning kit that contains cleaning solvent and paper cloth suitable for optical instruments. Such cleaning kits are available in optical supply and photo stores.

Please carry out the following steps for cleaning:

- Turn off the sensor. Disconnect it from power to prevent the mirror from spinning.

- Remove grains of sand or other abrasive particles by blowing them away. Do not rub them over the surface as this may scratch it and reduce the performance of the sensor.
- Take a fresh cloth, sprinkle it with the cleaning solvent and gently wipe mirror and pane. If stains still remain after the first cleaning repeat this procedure with a fresh cloth.

4 Electronic Control Unit (ECU)

The ECU is the platform that runs the signal processing algorithms. It reads the scan data of one or more laserscanners via the ARCnet interface, and sends the resulting output on CAN and Ethernet interfaces.

Fig. 22: The Electronic Control Unit (ECU)

4.1 Mounting the ECU

The ECU may be mounted in any desired position or location. However, some general rules should be obeyed:

- Fasten the ECU in its mounting position to prevent the housing from crashing about, causing damage to itself and other components.
- Protect the ECU from shocks and heavy vibrations. Although the system does not contain moving parts, excessive shocks may cause damage to the system or loosen connectors.
- Use at least 0.75 mm^2 cables to connect the ECU to power. Using cables with a smaller diameter may cause the cables to overheat.
- Keep the ECU away from water. Although it is waterproof, exposure to water will not increase the life expectancy of the ECU.

Note that although all interface connectors are waterproof, they will become watertight only if there is a matching, also watertight connector or end cap mounted to them!

4.2 Connecting the ECU

The ECU has the following connectors:

- 2x Sensor (15-pin D-Sub, female)
- 1x serial port (15-pin D-Sub, male)
- 1x CAN (9-pin D-Sub, male)
- 1x Ethernet (RJ45)
- 1x Power

For the basic setup of the system, please refer to Fig. 13.

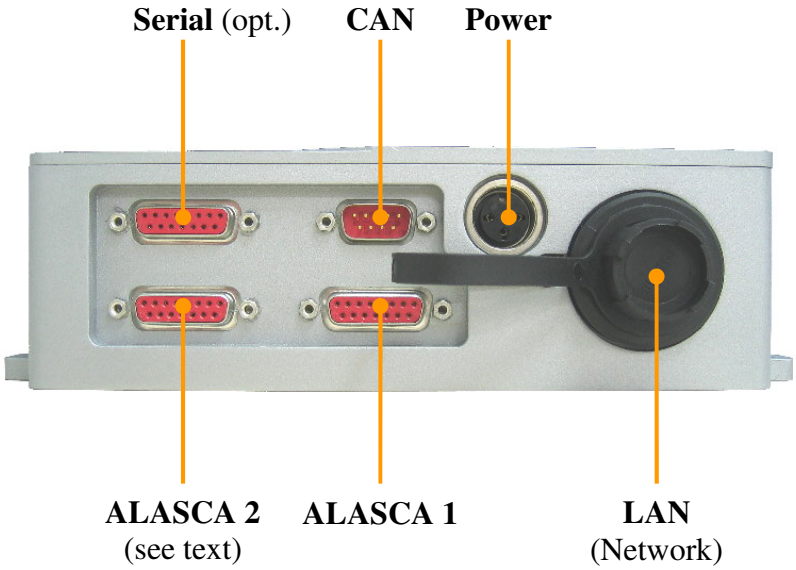


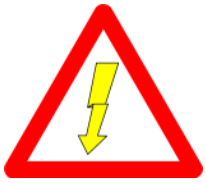
Fig. 23: Interfaces at the rear of the ECU.

4.2.1 Interfaces

The ECU has the following interfaces (see also Fig. 13):

- **Power:** The ECU requires a 12 V DC, 20 W power supply. Its connector cable has a red plug for 12 V and a black plug for ground (GND). Typical power consumption of the stand-alone ECU is less than 12 Watts. Note that the ECU may also supply power to the connected sensors, which will increase the current on this power connector (see also section 3.2.1).
- **ALASCA 1:** Connects laserscanner and ECU, see section 3.2.2. Its pinout is directly compatible with the ALASCA.
- **ALASCA 2:** This connector is for an optional second laserscanner. The pinout is the same as for connector **ALASCA 1**. To connect a second ALASCA successfully, Ibeo must prepare the ECU specially. By default, the ECU supports only one laserscanner.
- **CAN:** The ECU is equipped with a standard CAN interface. Application data is sent and received on this interface; see section 3.2.3.

- **Serial:** This connector holds the serial port of the ECU ("COM1") as well as the serial interface of the ALASCA connector. It can be used to connect the *SyncBox* for the synchronisation of the laserscanner(s), see section 5.
- **LAN:** 10/100-BaseTX Ethernet interface. This high-speed interface offers access to all data from the ECU, as well as control of the ECU software. Software updates can be downloaded via the LAN port, see section 4.5.



Do not open the housing of the ECU unless being explicitly instructed to do so! Opening the housing may break its watertight seal. Before touching any components take measures against electrostatic discharge as this might destroy the ECU!

4.2.2 ECU connector pinouts

Power connector pinout of the ECU:

Pin	Signal name	Description
1	+12 V	Positive power input, may be directly connected to vehicle power (+12 V only!)
2	GND	GND (Vehicle)
3	+12 V	Same as pin 1

The matching connector is a Lumberg Type 033203 plug. It is available from Farnell InOne (www.farnell.com), order no. 329-6611.

ALASCA connector pinout of the ECU:

Pin	Signal name	Description
1	ARC out –	ARCnet out "–"
2	ARC in –	ARCnet in "–"
3	GND	Signal ground for all signals
4	+ 5V ext.	For active ARCnet termination, max. 400 mA, via jumper, default = off
5	RxD 1 / 2	RS232 RX (receive)
6	+12 V	Power supply: +12 V DC
7	+12 V	Power supply: +12 V DC
8	Shield	Connect to cable shield
9	ARC out +	ARCnet out "+"
10	ARC in +	ARCnet in "+"
11	Shield	Connect to cable shield
12	TxD 1 / 2	RS232 TX (transmit)
13	Pwr GND	Power ground (= Signal GND)
14	Pwr GND	Power ground (= Signal GND)
15	Shield	Connect to cable shield

The signal names of the RS232 signals are identical to the ALASCA signal names, and thus the signal direction is stated from the sensor's point of view. For instance,

RxD on the ECU's **ALASCA 1** connector is the receive line of the sensor, and must be connected to an external TxD signal on the **Serial** connector.

The ECU's ARCnet controller sends and receives data on the ARC out lines (pins 1 and 9). The ARC in lines (pins 2 and 10) are used for termination of the bus only. If an external termination closer to the sensor(s) is used, the ARCnet bus must not be fed back into the ARC in lines.

Serial connector pinout of the ECU:

Pin	Signal name	Description
1	RS 1 RxD	RxD signal of **ALASCA 1**. This pin is internally connected to pin 5 of connector **ALASCA 1**. Note that signal ground is RS_GND!
2	RS 1 TxD	TxD signal of **ALASCA 1**. This pin is internally connected to pin 12 of connector **ALASCA 1**. Note that signal ground is RS_GND!
4	RS 2 RxD	RxD signal of **ALASCA 2**. This pin is internally connected to pin 5 of connector **ALASCA 2**. Note that signal ground is RS_GND!
5	RS 2 TxD	TxD signal of **ALASCA 2**. This pin is internally connected to pin 12 of connector **ALASCA 2**. Note that signal ground is RS_GND!
6	GND	GND of ECU COM1
7	RxD	Receive line of ECU COM1
8	TxD	Transmit line of ECU COM1
14	RS_GND	Sensor RS232 signal ground. This ground is internally connected to vehicle ground.
15	RS_GND	Sensor RS232 signal ground. This ground is internally connected to vehicle ground.

CAN connector pinout of the ECU:

Pin	Signal name	Description
2	CAN_L	CAN bus "low"
3	GND	Signal ground for all signals
7	CAN_H	CAN bus "high"

LAN connector of the ECU:

The LAN connector is a standard 10/100-BaseTX Ethernet connector. The matching plug is a BULGIN PX0834/B type, available from Farnell InOne, Order No. 428-5864. Bulgin also offers read-to-use Ethernet cables.

4.3 LED signals

The tracking software running on the ECU is called "*AppBase*". The LEDs of the ECU show the current state of ECU and *AppBase*:

LED	Status	Description
red	on	Power supply on
yellow	off	ECU start-up phase (approx. 45 sec), neither *AppBase* running nor sensor connected
yellow	quickly flashing	The sensor is connected but *AppBase* is not (yet) running
yellow	slowly flashing	*AppBase* tries to connect to the sensor but the sensor is not connected
yellow	on	Data transmission on the ARCnet bus between ECU and sensor
green	flashing	Data transmission on the Ethernet

4.4 Start-up and shut-down

The *AppBase* software is started automatically after booting the ECU. It takes approximately 45 seconds to boot the ECU.

Unlike a PC, the ECU needs no preparation to shut down. It can be shut down at any time simply by removing power.

4.5 Software Updates

The *AppBase* software on the ECU and consists of three files: AppBase.exe, AppBase.ini, and AppBase.dat. It is not always necessary to update all three files.

The following steps describe the update process on a Windows® system:

1. Modify the network configuration of the host PC to get the ECU connected:
   - This operation requires administrator privileges.
   - Open the network settings by clicking on *Start* → *Settings* → *Network Connections*. In the *Network Connections* dialog, click with the right mouse button on the icon of the *Local Area Connection*, and select *Properties* to open the *Local Area Connection Properties* dialog (Fig. 24, left).
   - In this dialog, first click on *Internet Protocol (TCP/IP)* and then click on *Properties*.
   - The *Internet Protocol (TCP/IP) Properties* dialog appears (Fig. 24, right). **Write down the original settings shown in this dialog to restore them later.**
   - Click on the radio button *Use the following IP address*.
   - Edit the four parts of the IP address. The first three parts must be equal to the ones of the IP address of the ECU (e.g. 10.152.10, see label at the ECU). The last part must differ at your choice from the IP address of the ECU.
   - Set the subnet mask to the same mask that is printed on the label of the ECU and close the dialog with the *OK* button.
   - **Note:** If firewall and/or virus scanner software is running on the host PC, this may cause some trouble when connecting to the ECU. In this case, try to disable the firewall and/or the virus scanner temporarily.

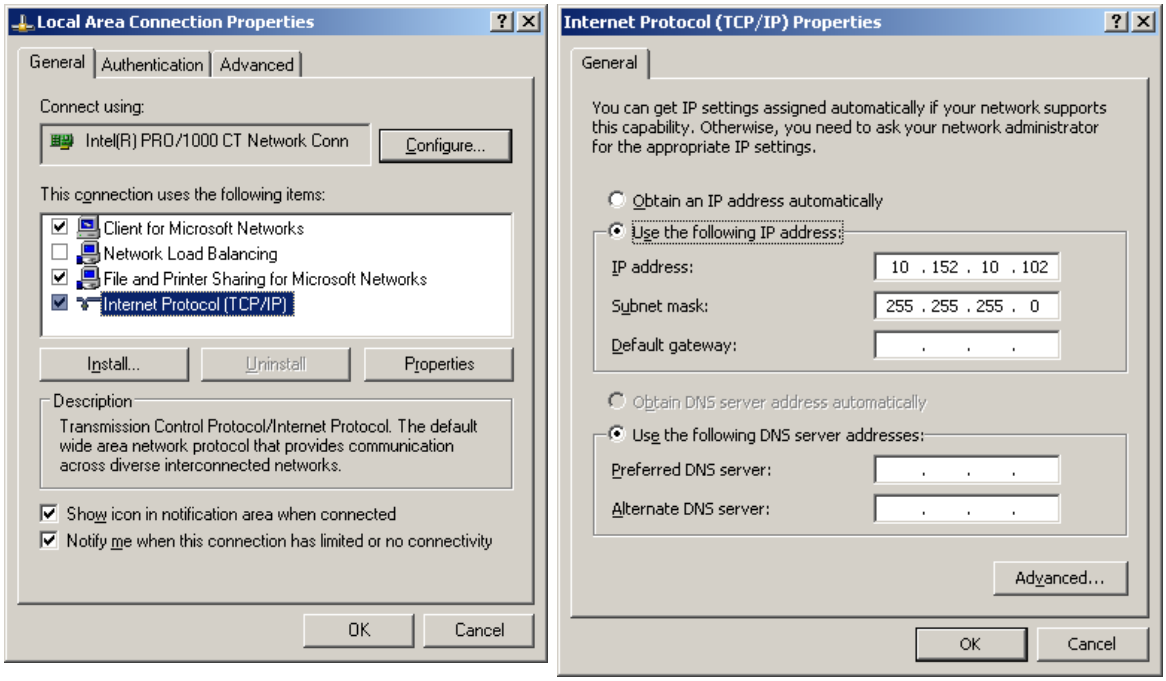


Fig. 24: Changing the IP address of the host PC to connect to the ECU

2. Connect the ECU to the LAN (local area network). Maybe you need a standard CAT-5 Ethernet patch cable. Alternatively, you may connect the ECU directly to the host computer with the delivered Ethernet cross-link cable.
3. Start the program *ServiceInterface.exe* (Fig. 25) from the Ibeo application CD-ROM, directory \Tools\ServiceInterface. Insert the IP address of the ECU (see label on the ECU) and click on the button *Stop Application* to close the *AppBase*. This operation does not stop the laserscanner.

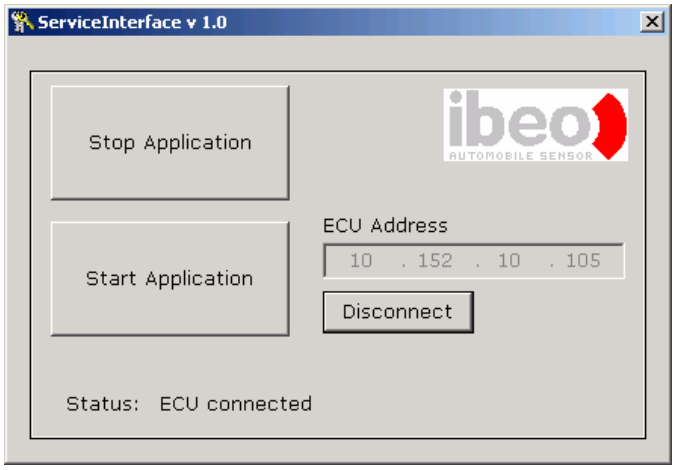


Fig. 25: Service Interface

4. If there is a FTP program on the host computer, start it. Or you can use the freeware windows based FTP program “FileZilla” (Fig. 27). The Ibeo application CD-ROM contains this program. To install the program, start the *FileZilla_x_x_xx_setup.exe* and follow the instructions. Then start the program. The following description is based on this program. Other FTP programs work similarly.

5. Insert the IP address of the ECU, the user name “Administrator”, and the password “Administrator” in the three text fields. Then click on the *Quick connect* button.

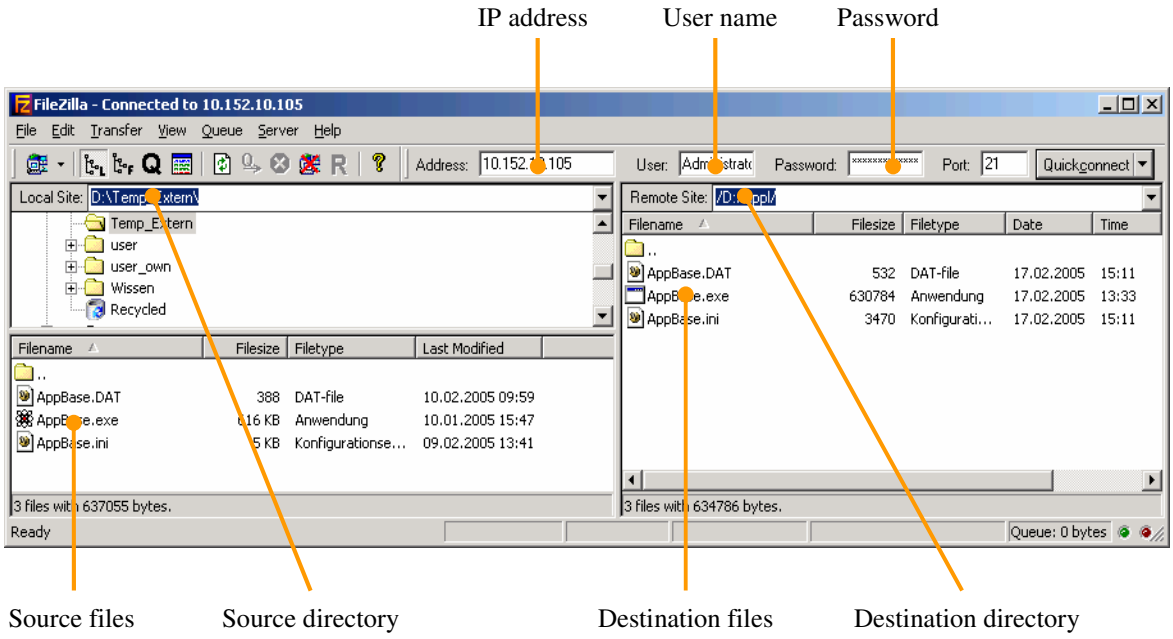


Fig. 27: FTP program “FileZilla”

6. Open the source directory where the *AppBase.** files are stored on the host computer. Open the destination directory D:\Appl\.
7. Copy all new *AppBase.** files via drag-and-drop to the destination directory (click on the file, hold left mouse button, and move the mouse to the destination directory). Another way is click with the right mouse button on the file and select *upload*. Click on the *OK* button to overwrite the existing files.
8. Close the FTP connection and restart the ECU (power-up) or click on the *Start Application* button in the program *ServiceInterface.exe*.

Do not delete any files on the ECU.

4.6 Maintenance

The ECU does not require maintenance and has no user-serviceable parts inside. In case of failure, please contact Ibeo.

4.7 Ethernet interface

This document describes the Ethernet interface of the ECU, which provides data transfer from the ECU to a host computer (e. g. for data visualisation).

4.7.1 Definitions

The following data types are used to describe the Ethernet interface:

Type	Description
INT8 **INT16** **INT32**	signed integers of 8, 16, or 32 bit length, respectively
UINT8 **UINT16** **UINT32**	unsigned integers of 8, 16, or 32 bit length, respectively

When sending multi-byte data types (i. e. data with a type other than INT8 or UINT8), care must be taken of the correct byte order (also known as *Endianness*) because any data is sent as a byte stream on the Ethernet. There are two different byte orders in common use, called *Little Endian* (e.g. Intel® x86 compatible processors) and *Big Endian* (e.g. Motorola®, TCP/IP network byte order). They differ only by the direction of reading.

The following table shows the byte streams for similar sample data that have a length of four bytes each ("0x…" denotes hexadecimal numbers):

Sample data	Little Endian	Big Endian
4 × **UINT8** : 0x01, 0x02, 0x03, 0x04	0x01, 0x02, 0x03, 0x04	0x01, 0x02, 0x03, 0x04
2 × **UINT16**: 0x0102, 0x0304	0x02, 0x01, 0x04, 0x03	0x01, 0x02, 0x03, 0x04
1 × **UINT32**: 0x01020304	0x04, 0x03, 0x02, 0x01	0x01, 0x02, 0x03, 0x04
	→ Intel® x86 standard	→ TCP/IP standard

Standard PCs with Intel® compatible x86 processors use Little Endian, whereas Big Endian is the recommended byte order for data exchange on the network. If your processor architecture is of type Little Endian, data must be converted between Little and Big Endian. Users of a Microsoft® Windows® operating system can do this e. g. using the C functions *htons*(), *htonl*(), *ntohs*(), and *ntohl*() that are declared in the Microsoft® header file "winsock2.h".

4.7.2 Data Transfer

The Ethernet interface uses the TCP/IP protocol and port number 12 000 for communication. Data is sent automatically when connecting to the port.

All data that is sent to or received from an Ibeo application has the same general structure, a 16 bytes message header followed by the message body itself:

Message header: 4 × UINT32, always Big Endian

1. **"Magic word"**, always UINT32 0xAFFEC0C0 (Note: 0 = zero, not letter O)
2. **Size** of message body in bytes (UINT32)
3. **Data type** of message body, see constants FILE_TYPE_… in the header file "ASL.h" (UINT32)
4. **Time-stamp** in milliseconds when the message was sent (UINT32)

Message body: The byte order is always Big Endian.

Different data types (e. g. scan data, object data, …) are all transmitted on the same port. Since the data packet size on the Ethernet is limited, the message may be spilt-up into several packets. This packet splitting is independent of the messages sent, thus it is possible, that the header of the next message is sent within the same packet as the end of the current message. Since the message length is known from the header, it is easy to find the beginning of the next message even if it is in the middle of a packet.

In case of a transmission error or data jam, the magic word 0xAFFEC0C0 can be used to find the next message header – just read data from the Ethernet until this byte sequence is received, and you have got the start of the next message header. (AFFEC0C0 is a German play on words which means "monkey Coco".)

For decoding the different message types, C source code is available from Ibeo. Alternatively, a self-made parser can be constructed based on the following sections.

4.7.3 Decoding object data

An object data message has the file type 1 = FILE_TYPE_OBJECTDATA. It consists of a sequence of blocks with eight byte length each. Each block is a copy of a CAN message. All CAN messages of one object data message refer to the same scan. For details about the individual CAN messages please refer to the document "Ibeo CAN Specification" [3].

The theoretical maximum size of an object data message is 3 136 bytes (16 bytes header + 390 × 8 bytes). It is recommended to provide a buffer of this size, even if the message size is much smaller during normal operation.

Since all object data are UINT8 (i. e. bytes), the Endianness does not matter here.

4.7.4 Decoding scan data

A scan data message has file type 15 = FILE_TYPE_COMPRESSED_SCAN. This message contains a subset of the scan data information, so called *compressed scan data*. The compression removes internal administrative information from the scan that is not relevant for the scan data receiver.

The message length depends on the number of scan points N in the current scan. The theoretical maximum number of scan points depends on the scanner type, the angular scan range and resolution. For an ALASCA there are $N \leq 8\,648$ points (max. 270° scan range including start and end angle × 4 channels × 2 sub-channels / 0.25° resolution = 1 081 × 4 × 2). In this case, the message size is ≤ 103 808 bytes (= 2 × 16 + 8 648 × 12, see Fig. 28).

The message body contains the compressed data of a single scan. As Fig. 28 shows, the message body is subdivided into a scan header (16 bytes) followed by a sequence of *N* scan points (12 bytes each). The following tables show the structure of the scan header and scan points.

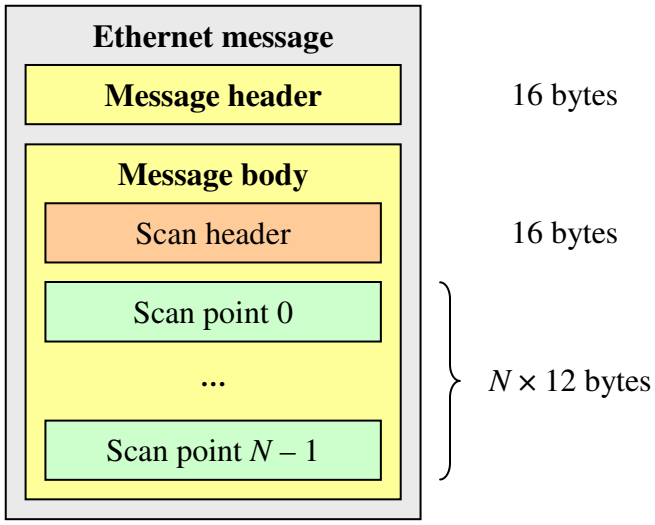


Fig. 28: Structure of the message when receiving compressed scan data.
The total message length is $12N + 32$ bytes where N is the number of scan points.

SCAN HEADER	Data Type	See note	Description (decoding instructions see following notes)
Version	UINT8	1	Version number of the data structure, current version = 1
ScannerType	UINT8	2	ALASCA has type 2
ECU_ID	UINT8	1	ID of the ECU that has sent this scan
PadByte	UINT8		Unused, just for proper memory alignment
TimeStamp	UINT32	3	Time-stamp of the scan in milliseconds
StartAngle	INT16	4	Scan start angle [rad × 10^4], ISO 8855 coordinate system [5]
EndAngle	INT16		
ScanCounter	UINT16	1	Consecutive scan number
NumPoints = *N*	UINT16	1	Number of subsequent scan points

SCAN POINT	Data Type	See note	Description (decoding instructions see following notes)
ScannerID	UINT8	1	ID of the scanner that has detected this point
Channel	UINT8	5	Zero-based channel number (0 = bottom channel)
SubChannel	UINT8	6	Zero-based sub-channel (0 = A, 1 = B, …)
PointStatus	UINT8	7	Point can be ground, rain, dirt etc.
XCoord	INT16	8	Cartesian coordinates of the point according to ISO 8855 (except for *z*-coordinate, see note 8)
YCoord	INT16		
ZCoord	INT16		
EchoPulseWidth	UINT16	9	0 = invalid or not available; arbitrary unit

Note 1: Integer value, no decoding necessary

Note 2: Currently known scanner types are: 0 = LD Automotive, 1 = LD ML, 2 = ALASCA. These values are also defined as constants SCANNER_TYPE_… in "ASL.h".

Note 3: The integer value represents a reference time-stamp given in milliseconds when the scanner measures to its 0° direction ("ahead"). *Exception:* If there is only a single laserscanner (no fusion system) and this scanner is mounted in the rear of the vehicle, the 180° direction ("backwards") is used as reference time-stamp.

Due to the rotation of the scanner's mirror it takes some time to collect the points of a scan (e. g. 50 ms at 10 Hz and 180° scan range). This recording time results in a distortion of objects that move relatively to the laserscanner. To reduce this effect, all scan data are compensated for the absolute velocity of the vehicle (so called *scan data correction*). After that step all scan points are corrected to the positions they (nearly) had at the reference time-stamp and the scan data has become an instantaneous snap-shot of the surroundings. The restriction "nearly" refers to the fact that the scan data is not compensated for the absolute velocity of the moving objects in the surroundings which also contributes to object distortions.

Note 4: Angles are specified according to the ISO 8855 coordinate system (see section 6.2) with the unit radians × 10^4. The conversion between an INT16 value n and angle α in radians is

$$\alpha = n / 10\,000, \quad -31\,416 \le n \le +31\,416$$
$$n = [10\,000\,\alpha], \quad -\pi \le \alpha \le +\pi$$

where [·] means rounding towards the nearest integer. Please note that the extreme values $n = \pm 31\,416$ slightly exceed the $\pm\pi$ boundary. The angular resolution arising from this conversion is approximately 1° / 175 ≈ 0.0057° (= 10^{-4} rad).

Note 5: Channel numbers:
3 = channel 4 (yellow, top)
2 = channel 3 (green)
1 = channel 2 (blue)
0 = channel 1 (red, bottom)

Note 6: Sub-channels:
0 = sub-channel A (first echo)
1 = sub-channel B (second echo)

Note 7: The point status can be one of the following values:

0 =	PT_STATUS_OK:	Normal scan point
1 =	PT_STATUS_INVALID:	Do not use this scan point. Note that invalid scan points are not transmitted; therefore this status should never be received.
2 =	PT_STATUS_RAIN:	For echoes from rain drops
3 =	PT_STATUS_GROUND:	For echoes from the ground
4 =	PT_STATUS_DIRT:	For echoes from dirt on the pane of the integration chamber.

Note 8: Distances are encoded such that values up to 100 m have a resolution of 1 cm and values above 100 m have a resolution of 10 cm. The conversion between an INT16 value n and the corresponding distance d in metres is as follows:

$$d = \begin{cases} n \times 0.1\,\text{m} + 900\,\text{m} & \text{if} \quad n < -10\,000 \\ n \times 0.01\,\text{m} & \text{if} \quad -10\,000 \le n \le +10\,000 \\ n \times 0.1\,\text{m} - 900\,\text{m} & \text{if} \quad n > +10\,000 \end{cases}$$

$$n = \begin{cases} 10\,(d - 900\,\mathrm{m})/\mathrm{m} & \text{if} \quad d < -100\,\mathrm{m} \\ 100\,d/\mathrm{m} & \text{if} \quad -100\,\mathrm{m} \le d \le +100\,\mathrm{m} \\ 10\,(d + 900\,\mathrm{m})/\mathrm{m} & \text{if} \quad d > +100\,\mathrm{m} \end{cases}$$

For INT16 values the ranges of n and d are:

$$\begin{aligned} -32\,768 &\le n \le +32\,767 \\ -2\,376.8\ \mathrm{m} &\le d \le +2\,376.7\ \mathrm{m} \end{aligned}$$

Note that it is *not recommended* to use the z-component of a scan point; it is only transmitted for completeness. Practically the z-component is problematic because it refers to the *local* coordinate system of the respective scanner (even in case of a fusion system). Since the pitch angle of the laserscanner is not known, a conversion to the vehicle's ISO 8855 coordinate system is impossible. Nevertheless, the z-components of all points that originate from the same scanner can be compared relatively.

Note 9: Currently an echo pulse width is given only as an integer value with an arbitrary unit. A small value corresponds to a small echo pulse and vice versa. For future sensor generations a conversion e. g. to nanoseconds (temporal pulse length) and/or metres (spatial pulse length) may be available.

Fig. 29: The *SyncBox*

5 SyncBox

Note: The *SyncBox* is an optional system component that may not be part of the ALASCA system delivered by Ibeo

The *SyncBox* (Fig. 29) is a hardware interface to synchronise laserscanners with other external devices (e.g. a camera). Synchronisation minimises time-shifts in the data collection of the connected laserscanners. This is especially useful for scan data fusion where synchronisation reduces the need for scan data corrections during processing. Thus, the induced error is also minimised, resulting in an improved performance and reliability.

During the synchronisation process, the laserscanners adapt their scan frequency to a given synchronisation frequency in such a way that each scanner measures at its 0° direction at the time of the synchronisation pulse.

Note that in a fusion system with two laserscanners *only*, the *SyncBox* is not necessary for synchronisation because both laserscanners can synchronise directly (master/slave mode, see section 5.9).

The *SyncBox* allows the synchronisation of up to two Ibeo laserscanners. Depending on the generation of the synchronisation frequency, the *SyncBox* operates either in timer mode or in bit I/O mode:

- In **timer mode**, the *SyncBox* itself generates the synchronisation frequency. This signal is also available on a bit I/O line to synchronise external sensors to the 0° direction of the scanners. For instance, a video camera can capture a video frame when the scanner(s) measure in the 0° direction.
- In **bit I/O mode**, the *SyncBox* gets the synchronisation frequency from an externally generated periodic signal that is received on a bit I/O line. This mode allows the synchronisation of the laserscanners to an external frequency. For instance, the capturing of camera frames can trigger the synchronisation.

5.1 Synchronisation details

The accuracy of synchronisation (i.e. the temporal difference between the synchronisation pulse and the crossing of the 0° direction) is about ±200 µs if no external forces act upon the laserscanner (esp. no angular acceleration). This is equivalent to an angular accuracy of ±0.9° or ±1.8° around the true 0° direction for rotation frequencies of 12.5 or 25 Hz, respectively.

The external synchronisation frequency f_{sync} may be selected from the range 10 Hz $\leq f_{sync} \leq$ 40 Hz and should have a relative accuracy better than 0.1 % (the better f_{sync} the better synchronisation). Therefore, it is not recommended to send the synchronisation signal from a PC because its timers are usually not that accurate. Instead, a dedicated microcontroller should be used to generate the synchronisation signal. Also for a correct synchronisation, the scan area must contain at least the angular range from +5° to –5°.

It is important that synchronisation frequency and scan frequency are approximately the same; otherwise, the controller will fail to synchronise. Therefore, the configuration parameter **RotationFreq** (see section 6.3.2) should be equal to f_{sync}.

5.2 System layout in timer mode

The *SyncBox* connects to the laserscanners via RS232 by means of the ECU. In the configuration shown in Fig. 30, the *SyncBox* sends its signals to the ECU that distributes them to the laserscanners. The *SyncBox* runs in timer mode, generating the synchronisation pulses internally. Each trigger time also causes a 5 ms `high` pulse on the trigger output, which allows triggering external devices at the same time.

To run in timer mode, the **SyncMaster** parameters in the file “AppBase.ini” must be set to FALSE in the sections “[Sensor_*n*]”:

```
SyncMaster = FALSE
```

Note that the same system layout applies if only one laserscanner is connected.

5.3 System layout in bit I/O mode

In bit I/O mode, the trigger signal is supplied by an external source to the trigger input of the *SyncBox* (Fig. 31). Upon receiving a `low`-to-`high` transition on the trigger input, the synchronisation commands are sent to the laserscanners. The signal is also acknowledged by setting the trigger output signal for 5 ms (see chapter 5.5 for detailed timing examples).

Note: After the transition, the external synchronisation signal should remain in `high` state for at least 1 ms in order to allow the *SyncBox* a stable detection of the signal.

Since the connected laserscanners will derive both their scan frequency and head position from the trigger input signal, care must be taken that this signal is within the specification of the connected laserscanners. If the trigger signal frequency exceeds the limits of the scan frequency, the laserscanner will go to error mode and report this in its scan data (sensor status field). Accordingly, if the input signal has a high time jitter, the laserscanners will be unable to synchronise to that signal. In this case, a jitter warning signal is generated.

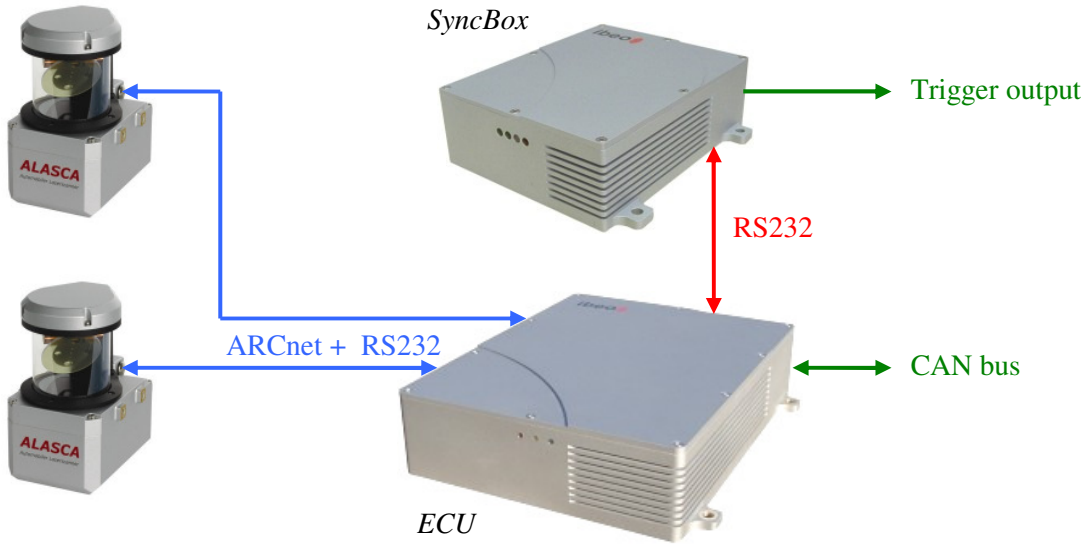


Fig. 30: System layout in timer mode (internal triggering)

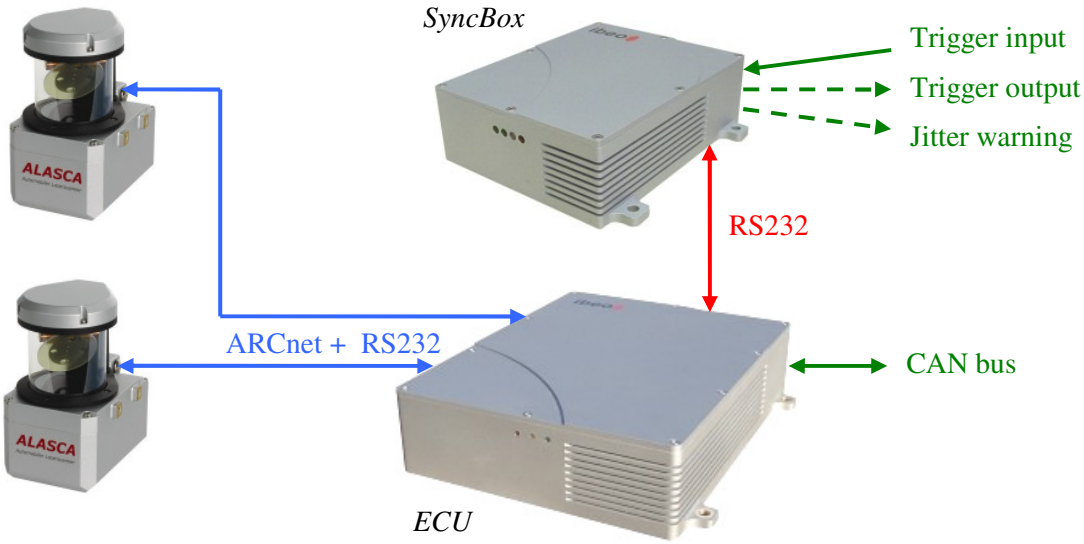


Fig. 31: System layout in bit I/O mode (external triggering)

5.4 SyncBox LEDs and connectors

This section describes the LEDs at the front of the *SyncBox* (see Fig. 32) and the connectors at the backside (Fig. 33).

The used connectors are not waterproof.
Therefore, the *SyncBox* must not be exposed to water.

LED	Status	Description
Power	on	Power supply on
Mode	on, red	Bit I/O mode
Mode	on, green	Timer mode
Sync0/1		Reserved for future use

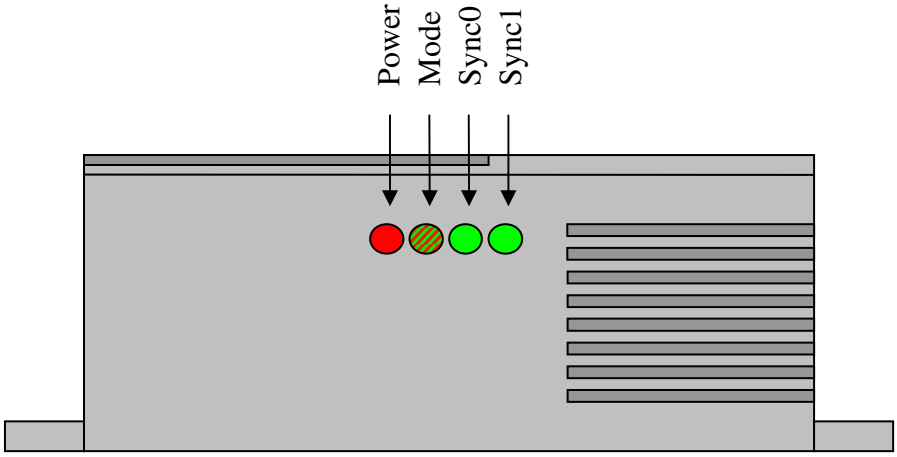


Fig. 32: Front side of the *SyncBox*

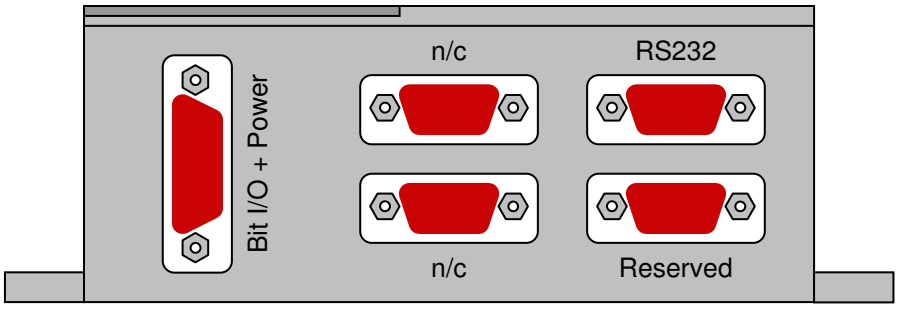


Fig. 33: Back side of the *SyncBox* (n/c = not connected)

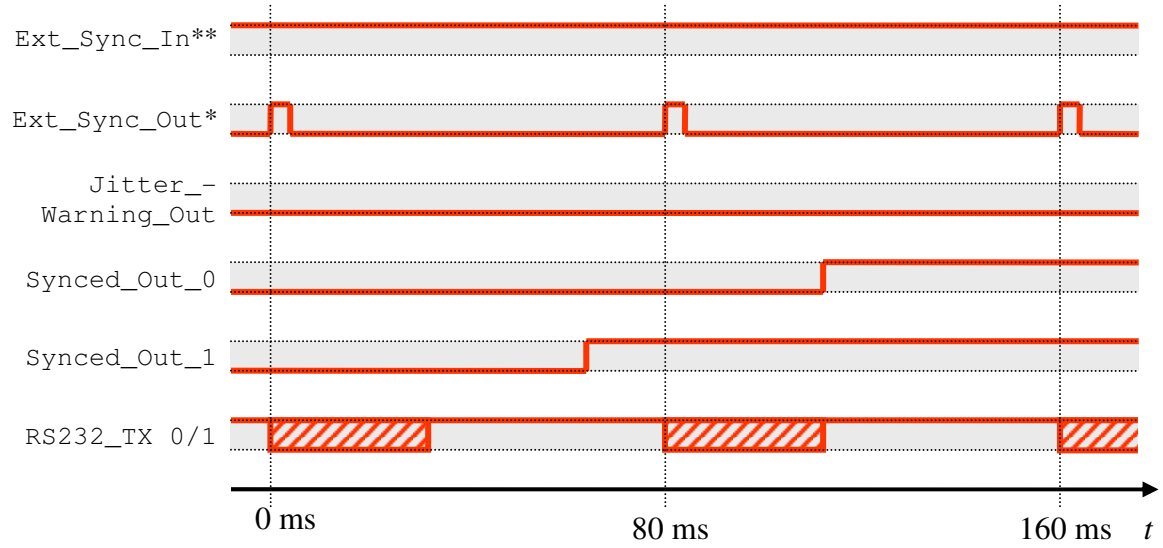
5.5 Signals of the SyncBox (TTL and LED)

These signals are available on the bit I/O connector of the *SyncBox*:

Signal name	Operation	Description
`Ext_Sync_In` ("trigger input")	bit I/O mode only	Input signal for external synchronisation. The first `low`-to-`high` transition on this input will activate the bit I/O mode. Subsequent `low`-to-`high` transitions will cause a sync signal to be sent to the laserscanners. The sync signal is acknowledged by a corresponding output on `Ext_Sync_Out`.
`Ext_Sync_In_Dbl` ("double-speed trigger input")	bit I/O mode only	Same as `Ext_Sync_In`, but only each second `low`-to-`high` transition will cause a sync signal. For instance, a camera that captures 25 frames/s can synchronize a scanner with 12.5 Hz.
`Ext_Sync_Out` ("trigger output")	always	Output signal. A 5 ms high-pulse is generated every time a sync signal is sent to the laser-scanners.
`Ext_Sync_Out_Inv`	always	Same as `Ext_Sync_Out`, but with inverse polarity. During sync, this signal generates a `low` pulse.
`Jitter_Warning_Out` ("jitter warning")	always	Output signal, valid only in bit I/O mode. If this signal is `high`, the input trigger applied at `Ext_Sync_In` has a time jitter of more than 1 ms and is not suitable for laserscanner synchronisation.
`Synced_Out_0`, `Synced_Out_1`	always	The signal is `high` if the connected laserscanner on RS232 0/1 is synchronised.

5.5.1 Timing example for timer mode

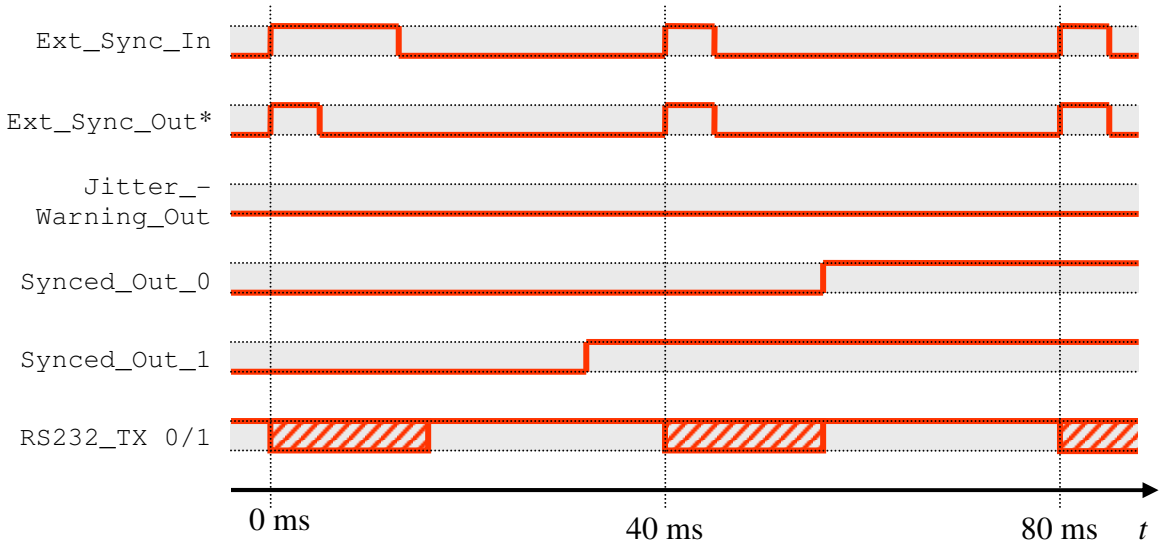
The scan frequency is 12.5 Hz. Scanner 0 notifies the *SyncBox* at t = 112 ms that it is synchronised, scanner 1 does so at t = 64 ms. `Ext_Sync_in` must be kept `high` at all time to remain in timer mode.



*) The signal `Ext_Sync_Out_Inv` is the inverse of this signal.
**) If unconnected, `Ext_Sync_In` is kept at `high` level by an internal pull-up resistor.

5.5.2 Timing example for bit I/O mode

The signal `Ext_Sync_In` is supplied by an external source. After the first `low`-to-`high` transition, the *SyncBox* activates the bit I/O mode. In the figure shown below, the externally supplied scan frequency is 25 Hz (derived from external input). After a `low`-to-`high` transition is recognised, the SyncBox sends the sync command via RS232 and sets the signal `Ext_Sync_Out` for 5 ms.



*) The signal `Ext_Sync_Out_Inv` is the inverse of this signal.

5.6 Setting timer or bit I/O mode

By default, the *SyncBox* runs in timer mode after start-up. The operation mode changes automatically to bit I/O mode when a transition is detected on the input `Ext_Sync_In` or `Ext_Sync_In_Dbl`.

5.7 Interfaces and Pinouts

The *SyncBox* uses a bit I/O port and both serial interfaces. All inputs have internal 20 kΩ pull-ups to +5 V. Therefore, it is not necessary to make any connection in order to achieve a `high` level. Signals that should have a `low` level can be connected to GND directly.

Do not connect pins that are not expressly linked to a function (see tables below) as this *will* interfere with the SyncBox!

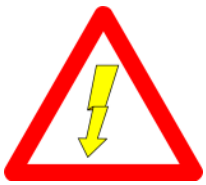
5.7.1 Power supply and bit I/O

Pin	Name	Function
4	`Synced_Out_1`	Output, `high` if the laserscanner connected on port 1 has signalled that it is synchronised.
5	`Jitter_Warning_Out`	Output, `high` if the time jitter of `Ext_Sync_In` signal is too large.
6	`Ext_Sync_Out_Inv`	Sync signal output (active `low`).
9	`Power_In`	Positive power input. The supplied voltage must be in the range of 9 to 18 V. Short-time transitions are also tolerable. This input can be directly connected to vehicle power.
10	`GND`	Ground pin corresponding to `Power_In`. It is also the `GND` reference for all output signals.
11	`GND`	
13	`Ext_Sync_Out`	Sync signal output (active `high`).
14	`Synced_Out_0`	Output, `high` if the laserscanner connected on port 0 has signalled that it is synchronised.
15	`Ext_Sync_In`	Input for external sync signal.

5.7.2 RS232 0/1

Pin	Name	Function
2	`TxD`	Transmit data. *SyncBox* sends commands on this line. Connect this line to `RxD` of the laserscanner.
3	`RxD`	Receive data. *SyncBox* receives laserscanner replies on this line. Connect this line to `TxD` of the laserscanner.
5	`GND`	Reference ground

5.8 Electrical characteristics



Do not reverse polarity on the input voltage!
Do not exceed the voltage limit!

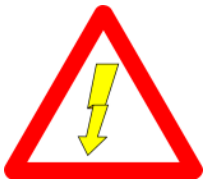
5.8.1 Power supply

The *SyncBox* requires a 12 V (9 – 18 V), 300 mA (3.6 W) power input. This input is filtered internally so that a direct connection to vehicle power is possible. It is also protected against reversed polarity. However, in case of a reversed polarity, a critical current may flow through the data lines and destroy the *SyncBox* and/or connected devices.

5.8.2 Bit I/O

The bit I/O outputs of the *SyncBox* are TTL compatible with a 5 V level and push/pull driver stage. Therefore, multiple outputs must not be connected together and outputs must not be connected to ground externally. No output signal may drive more than a 2 mA load, both to `high` or `low` level.

A `low` signal is a voltage level of 0.0 – 0.7 V, a `high` signal is a level of 2.4 – 5.0 V. The inputs are connected internally to +5 V with 20 kΩ pull-up resistors. In order to input a `low` signal, external devices must pull this level below 0.7 V.



Do not apply high voltage or transients to the signal lines, as this will destroy the internal components!

5.9 Synchronisation without SyncBox

The ALASCA also supports a self-synchronising mode. In this mode, one ALASCA (*master*) transmits its 0°-time to a second ALASCA (*slave*). The slave then adjusts its own mirror movement to match the master exactly. To enter this synchronisation mode, the AppBase.ini file must be edited as follows: One sensor must have set `SyncMaster = TRUE` (e.g. in the section [Sensor_0]) and the other one `SyncMaster = FALSE` (e.g. in section [Sensor_1]). FALSE is the default value for the parameter `SyncMaster`.

In order to synchronise in master/slave mode, the two ALASCAs must be connected to the ECU, and the special *SyncPlug* must be inserted into the serial connector of the ECU. The *SyncPlug* is delivered by Ibeo.

6 Object tracking

6.1 Overview

The ECU computer runs a complete object detection and tracking algorithm on the ALASCA scan data. This algorithm splits the scan data into objects and tracks those objects through subsequent scans. The result is, instead of the scan data, a set of object data with information for each object like position, size, outline and velocity. The object data is sent to the host system on a standard CAN bus. This frees the host computer from the task of isolating the relevant information from the huge amount of scan data that the laserscanners produce.

An overview of the standard tracking algorithm is shown on the right. After receiving the scan, it is split into segments. Segments are clusters of scan data that are believed to belong to one object. Then, the characteristics for each segment are calculated, such as the position, size, and number of scan points. At this stage of the algorithm, all those characteristics are purely static.

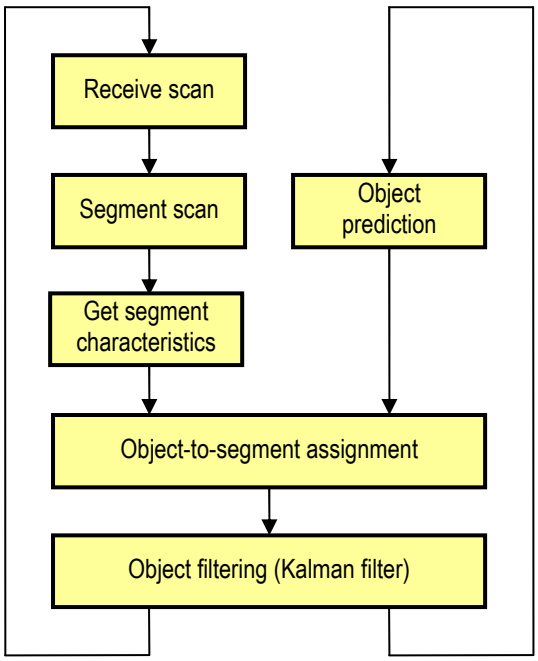


Fig. 34: Structure of the object tracking

In parallel, the prediction of the object movement is calculated using the output of a Kalman filter. All objects are extrapolated from the previous scan by one step to predict their position in the current scan. Then, the segments of the current scan are matched with the predicted objects, and the best matches are assigned. More than one segment may be assigned to one object because parts of the object may be blocked from view by some smaller object in the foreground. Finally, the object properties (position, size, velocity, uncertainties) are updated, using the precisely measured position of the assigned segment. This is done by updating the state vector of the object and running this vector through a Kalman filter. Unassigned segments are stored as new objects with default properties.

After the object detection and tracking is complete for the scan, the objects are sent to the host computer. The information for each object consists at least of a set of points on the object outline including the leftmost, rightmost and closest points, a velocity, and uncertainties for all values. All information is given in both x- and y-direction, in an ISO 8855 coordinate system (see next section).

6.2 ISO 8855 coordinate system

For input and output, the application uses a coordinate system according to ISO 8855, see [5]. All angles are given in the range from –180° to +180°, see Fig. 35.

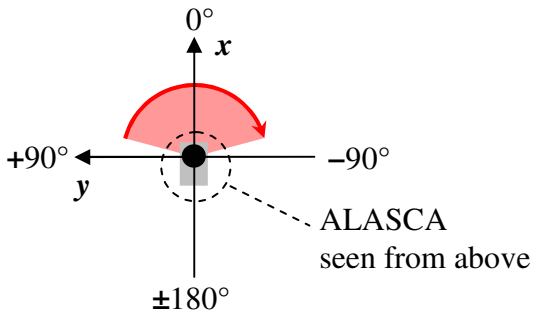


Fig. 35: Coordinate system of the ALASCA

If the ISO 8855 coordinate system is rotated by 90° clockwise, it turns into the well-known Cartesian coordinate system. ISO 8855 assumes that the vehicle moves along the *x*-axis (0° direction). This coincides with the ALASCA's 0° direction.

Since ALASCA's mirror rotates clockwise, the start angle of the scan is usually greater than the end angle. As an example in Fig. 35 a 160° scan area is drawn in red colour, starting at +80° and ending at –80°. If start and end angles were swapped this yielded the complementary scan area (crossing the ±180° boundary). The maximum usable scan area is limited to about 240°.

6.3 Parameter file "AppBase.ini"

The ECU will start the sensor automatically after booting. Before the first scan, the *AppBase* software restores the last saved parameter set from the ECU. After that, the sensor will send object data via CAN for every scan.

A number of parameters control the steps of the object tracking process. The parameters are stored in the file "D:\Appl\AppBase.ini" (short: *ini-file*) on the ECU's built-in CompactFlash card. The file can be edited with a text editor.

Note: Changing any parameters with the CAN interface has only a temporary effect until the power is switched off. In order to save the parameters so that they are restored after every power-on, use the "Store Parameters to Flash" command.

Note: Never change unknown entries in the ini-file!

6.3.1 Syntax

The AppBase.ini file is separated into sections where each section groups a number of parameters. A section begins with a line that contains the section name in brackets, e.g. "[Vehicle]". It follows a list of parameter assignments "*Parameter* = *Value*" where each assignment is placed in a separate line. Section and parameter names are not case-sensitive. A semicolon at the first column indicates a comment.

Parameter values must be entered without units. The expected units are shown in the table below:

Kind of Parameter	Unit
Distance, Offset	m
Angle	°
Velocity	km/h
Acceleration	m/s^2

To Boolean parameters TRUE or FALSE can be assigned.

6.3.2 Section "[Parameter]"

In this section, a number of parameters can be set that define the behaviour of the *AppBase* application and the object tracking. Most of the parameters for the tracking can also be set with the CAN interface; see CAN specification for details. The following lists describe the parameters using the scheme ⟨***parameter name***⟩ = ⟨*default value*⟩: ⟨*explanation*⟩

AutoStart = TRUE: Start connected sensors automatically when *AppBase* starts.

SensorUpsideDown = TRUE: Mounting direction of the sensor(s). TRUE: Sensor is mounted upside down, FALSE: Sensor is mounted normally.

ARCnetBaudRate = 5M: Baud rate of the ARCnet transmission: "5M" = 5 MBit/s, "10M" = 10 MBit/s

AppBaseID = 6: ARCnet-ID of the *AppBase* (ECU) on the ARCnet bus. Must be between 1 and 255 and must be unique on the bus.

CANBaudRate = 1: Baud rate of the CAN transmission. 1 = 1 MBit/s, 2 = 500 kBit/s

CANBaseID = 0x04F0: Base-ID of the CAN output, must be between 0x0001 and 0x07FE. The following CAN identifiers are used:

- CANBaseID – *n* for application messages
- CANBaseID for commands to the *AppBase*
- CANBaseID + 1 for object data

MinDist = 0.5: Minimum distance between two segments of scan points. If two segments lie closer together than this value, they are merged to a single segment. **MinDist** must be greater than 0.1; typical values are between 0.2 and 1.0 m. Increasing this value to greater values (e. g. 1 m) causes the segmentation and thereby the object tracking to become more coarse, meaning that objects become merged and are tracked together. Reducing this parameter to small values (e.g. 0.2 m) allows a much finer segmentation but causes objects to fall apart if there are small gaps in the outline. This parameter is only valid in *y*-direction. The minimum distance in *x*-direction is determined by the **XYFactor**, see below.

XYFactor = 3.0: Stretch factor for the segmentation in *x*-direction; must be greater than 1. In *x*-direction, the minimum distance between segments results from the product **MinDist** × **XYFactor**. Because the main direction of movement is forward it is advisable in most situations to allow the segmentation to become coarser in forward direction. This prevents objects to fall apart in longitudinal direction. Typical values are between 1 and 5.

ObjectPoints = 0: Number of points on the object outline. Valid values are between 3 and 16, 0 = adaptively select the number of object points (max. 16).

QualityCriterion = 0: After the object tracking is complete, the objects are sent on the CAN bus. Before sending the objects, they are sorted in order to send the most relevant object first. The following sorting criteria are available:

0. *Radial*: This quality criterion is the simplest of all sorting criteria. It sorts the objects by their distance from the centre of the vehicle coordinate system so that the closest object is sent first. Note that due to the possible offset between the sensor position and the origin of the coordinate system, the first object is not necessarily the object that is closest to the laserscanner.

1. *Look-Ahead*: The look-ahead criterion sorts objects by a roughly club-shaped area of priority. The axis between the origin of the coordinate system and the centre of the look-ahead criterion at position (15 m, 0 m) has the highest priority, while objects further to the sides have lower priority (see Fig. 36). This criterion takes

into account the typical forward movement of a vehicle, prioritizing objects in front of the vehicle higher.

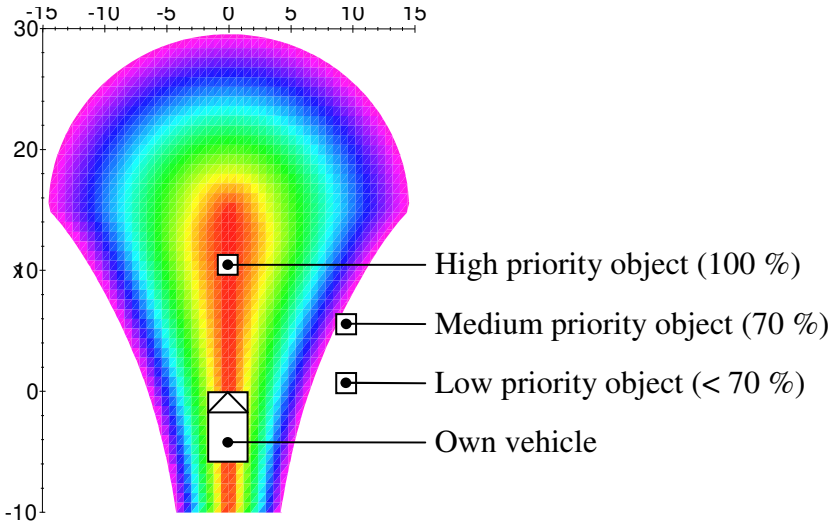


Fig. 36: Distribution of the priorities at the look-ahead criterion: red = high priority (100 %), violet = medium priority (70 %). The priority continues falling outside the coloured area; the limitation to 70 – 100 % is for better visualization only. Axes annotation is given in metres.

5. *ACC*: The ACC criterion generates the object output list according the requirements of an ACC application. This criterion filters all uninteresting objects for this application. In difference, other criterions only determine the output order by a quality and send as many objects as possible, even if they have a very low priority.

   The necessary conditions for an object to be sent are:

   - The object is tracked longer than 0.5 s.
   - The object must have the same main moving direction as the own vehicle.
   - A small object must be closer than 10 m. Small means that the result of the object classification is “unknown small”.
   - The object must be on my lane or on the lane right or left of it.
   - The object output order is similar to look-ahead.

SendObjects = 20: Maximum number of objects that may be sent for each scan (0 … 31)

OutputAreaX1 = 0.0,
OutputAreaY1 = 0.0,
OutputAreaX2 = 0.0,
OutputAreaY2 = 0.0: As Fig. 37 shows, the points (x_1, y_1) and (x_2, y_2) define a trapezoidal area, the so-called *output area*. If these points are specified only objects inside the output area are sent in the CAN or Ethernet output. An object is inside the output area if any scan point of the object is inside the output area. If all values are zero or not specified the output area filter is disabled.

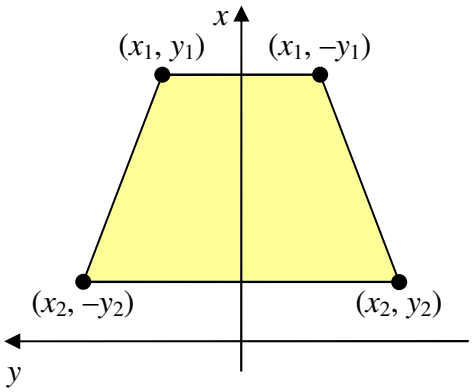


Fig. 37: The four parameters x_1, y_1, x_2, and y_2 define a trapezoidal output area.

RotationFreq = 20: Scan frequency of all connected scanners (8 … 40)

6.3.3 Section "[Sensor_*n*]"

Each sensor that is connected to the ECU must be configured by setting its scan start and end angle as well as the mounting position offsets. The section name for the first sensor is "[Sensor_0]", for the second sensor "[Sensor_1]", and so on.

StartAngle = 90.0,
EndAngle = –90.0: Start and end angle of the scan area (see Fig. 35). The maximum usable scan area is currently limited to about 240° to allow sufficient time for the ARCnet transmission of the scan data.

OffsetX = 0.0,
OffsetY = 0.0,
HorizontalAngleOffset = 0.0: Mounting position offsets (see section 6.4)

SyncMaster = FALSE: In a fusion system with two laserscanners, this flag distinguishes between the synchronisation master (TRUE) and slave (FALSE); see section 5.9.

6.3.4 Section "[RS232]"

The RS232 port can be used to transmit status and error messages in plain text. Each message comes along with the current time stamp. The RS232 protocol used is 8N1 (8 bit, no protocol, 1 stop bit). This section is optional, but if used, all four entries must exist.

SendStatus = FALSE: Turns the RS232 output on (TRUE) or off (FALSE).

SendANB = FALSE: Turns the AEB output on (TRUE) or off (FALSE). The AEB messages are for debugging purposes only. Do not activate this feature. AEB is described in detail in section 7.3.1.

ComPort = 1: Specifies the RS232 port.

BaudRate = 38400; Specifies the Baud rate (bits/s). Use only valid rates for RS232: 2400, 4800, 9600, 19200, and 38400.

6.3.5 Section "[Vehicle]"

This section of the ini-file describes the vehicle's geometry. The parameters listed below are shown together with exemplary values.

CenterToFrontAxle = 1.409: see Fig. 42

CenterToRearAxle = 1.291

DistOriginFront = 0.932

TurningCircle = 11.0

VehicleWidth = 1.745

VehicleLength = 4.682

For calculating the vehicle movement from the velocity and the steering angle some vehicle parameter are necessary. The transmission ratio of the steering (i. e. the ratio between the steering wheel angle and the front wheel angle) is given as a third-order polynomial with the four coefficients **SteerRatio0**, …, **SteerRatio3**. Two different definitions of the polynomial are available, type 0 and type 1. The choice between these types depends on the data available from the vehicle manufacturer.

Type 0: Transmission ratio

Let x be the steering wheel angle in degrees. Then the transmission ratio is defined as

$$TransmissionRatio = \mathbf{SteerRatio3} \cdot x^3 + \mathbf{SteerRatio2} \cdot x^2 + \mathbf{SteerRatio1} \cdot x + \mathbf{SteerRatio0}$$

The front wheel angle is calculated from the steering wheel angle by

$$FrontWheelAngle = x \,/\, (1.095 \cdot TransmissionRatio)$$

SteerRatioType = 0: The following coefficients belong to a transmission ratio polynomial:

SteerRatio0 = 16.155, (exemplary data)
SteerRatio1 = –4.241E–04,
SteerRatio2 = –2.178E–05,
SteerRatio3 = –2.516E–09: Coefficients of the transmission ratio polynomial

Type 1: Transfer function

Let x be the steering wheel angle in degrees. The front wheel angle is directly calculated by

$$FrontWheelAngle = \mathbf{SteerRatio3} \cdot x^3 + \mathbf{SteerRatio2} \cdot x^2 + \mathbf{SteerRatio1} \cdot x + \mathbf{SteerRatio0}$$

SteerRatioType = 1: The following coefficients belong to a transfer function polynomial:

SteerRatio0 = 0.0, (exemplary data)
SteerRatio1 = 5.383561E–2,
SteerRatio2 = 5.704158E–6,
SteerRatio3 = 1.199232E–7: Coefficients of the transfer function polynomial

6.3.6 Sections “[Velocity]” and “[SteeringAngle]” (Vehicle Data Parser)

A few modules of the application software *AppBase* need information about the vehicle like velocity, steering angle, and so on. If these parameters are not available, the modules will be disabled automatically. To be able to support CAN protocols of different vehicles *AppBase* has an integrated vehicle data parser. The parameters of the vehicle data parser are specified in the sections “[Velocity]” and “[SteeringAngle]”. Both sections need the same set of parameters as listed below. There are no default values for these parameters.

Identifier: The identifier of the CAN message that contains the vehicle parameter, e.g. the velocity. Valid values are decimal or hexadecimal numbers like 648 or 0x288.

FirstBit, **LastBit**: The range of bits where the parameter value is located in the CAN message. The first bit corresponds to the least significant bit (LSB) and the last bit to the most significant bit (MSB) of the value. A CAN message has 8 bytes with 8 bits each, thus **FirstBit** and **LastBit** must lie within the range from 0 to 63 (see examples below).

SignBitAvailable: For some parameters there is a sign bit placed in the CAN message that determines if the current value is negative or positive. If a sign bit is available set **SignBitAvailable** = TRUE, otherwise set to FALSE.

SignBit: If a sign bit is available, this value defines the position of the sign bit in the CAN message, e.g. **SignBit** = 15. If the sign bit is set, the value is considered as being negative.

ErrorValue: For some parameters, an error value is defined. If the parameter is equal to the given **ErrorValue** this parameter is invalidated and will not be used for internal calculations. If there is no error value defined, **ErrorValue** should be set to a value out of the range, e.g. 0xFFFF.

Factor, **Offset**: Parameter values must be decoded by a simple linear equation:

$$\text{decoded value} = \text{encoded value} \times \mathbf{Factor} + \mathbf{Offset}$$

The encoded value is the integer value extracted from the bits between **FirstBit** and **LastBit** and the optional **SignBit**. The units of **Factor** and **Offset** must be metric, e. g. velocities in m/s or angles in radians. (These are the internal units. Note that at the user interface of *ASD* or *AppBase* the units are converted to more common and user-friendly units like km/h or degrees.) If the parameter is not encoded, please set **Factor** to 1 and **Offset** to 0.

Take the steering angle parameter as an example. Assume a conversion factor of 0.5° per integer value and an angular offset of –10°. (The actual parameters can be found in your CAN specification.) Factor and offset must be converted to radians yielding **Factor** = 0.5° · π / 180° ≈ 0.008727 and **Offset** = –10° · π / 180° ≈ –0.174533.

UseLittleEndian: Defines if the data is encoded in Little Endian (= TRUE) or Big Endian (= FALSE). For more information about Endianness, please refer to page 27.

Example 1 (Little Endian)

The example describes how to get the velocity parameter out of a CAN message with the identifier 0x288. Let the velocity be encoded in byte 0 and byte 1 (coloured in light blue in the following table), and there is a sign bit (coloured in dark blue). The parameter is encoded using a factor of 0.02, an offset of 0, and the original unit (used by the vehicle) is km/h.

A CAN message has 8 bytes with 8 bits each. Byte 0 starts with bit 0 and ends with bit 7. Byte 1 starts with bit 8 and ends with bit 15 and so on.

	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Byte 0	7	6	5	4	3	2	1	0
Byte 1	15	14	13	12	11	10	9	8
Byte 2	23	22	21	20	19	18	17	16
Byte 3	31	30	29	28	27	26	25	24
Byte 4	39	38	37	36	35	34	33	32
Byte 5	47	46	45	44	43	42	41	40
Byte 6	55	54	53	52	51	50	49	48
Byte 7	63	62	61	60	59	58	57	56

For this example the following section should be specified in the ini-file:

```
[Velocity]
Identifier = 0x288
FirstBit = 0
LastBit = 13
SignBitAvailable = TRUE
SignBit = 14
ErrorValue = 0x3FFF
Factor = 0.0055555556     = 0.02 · (1 m/s) / (1 km/h) = 0.02 / 3.6
Offset = 0
UseLittleEndian = TRUE
```

Example 2 (Big Endian)

Note: The bit numbers 0 to 63 are independent of the number of bytes transferred in the CAN message. For example, if only one byte is transferred, only bit 56 to 63 is available.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	63	62	61	60	59	58	57	56
Byte 1	55	54	53	52	51	50	49	48
Byte 2	47	46	45	44	43	42	41	40
Byte 3	39	38	37	36	35	34	33	32
Byte 4	31	30	29	28	27	26	25	24
Byte 5	23	22	21	20	19	18	17	16
Byte 6	15	14	13	12	11	10	9	8
Byte 7	7	6	5	4	3	2	1	0

For this example, the following section should be specified in the ini-file:

```
[Velocity]
Identifier = 0x288
FirstBit = 24
LastBit = 37
SignBitAvailable = TRUE
SignBit = 38
ErrorValue = 0x3FFF
Factor = 0.0055555556      = 0.02 · (1 m/s) / (1 km/h) = 0.02 / 3.6
Offset = 0
UseLittleEndian = FALSE
```

6.4 Mounting position

The mounting position of the laserscanner(s) must be known to convert the local coordinate system of each laserscanner to the ISO 8855 coordinate system of the vehicle (see section 6.2). The coordinate transformation is composed of translations and rotations: The origin is shifted by (Δx, Δy, Δz), and there may be a rotation around each axis ($\Delta\alpha$, $\Delta\varphi$, $\Delta\psi$). From these six parameters, the parameters Δz (vertical offset), $\Delta\varphi$ and $\Delta\psi$ (roll and pitch angle) can be neglected here, because scan data is mainly evaluated using the bird's-eye view (cf. section 2.3). Therefore, the relevant parameters for detecting the mounting position are (see Fig. 38):

- (Δx, Δy) – the offset in the ground plane, and
- $\Delta\alpha$ – the horizontal angle offset (yaw angle).

The transformation from laserscanner to ISO 8855 coordinates is performed as follows:

1. Laserscanner coordinates — convert Cartesian (x, y) to polar = (r, α)
2. Rotation by $\Delta\alpha$ — (r', α') = (r, $\alpha + \Delta\alpha$), back to Cartesian = (x', y')
3. Translation by (Δx, Δy) — (x'', y'') = ($x' + \Delta x$, $y' + \Delta y$)
4. ISO 8855 coordinates — (x'', y'')

The mounting position can be determined manually or automatically. Note that – even though the automatic determination appears to be more attractive at first sight – the manual alternative is usually much faster and has a similar precision.

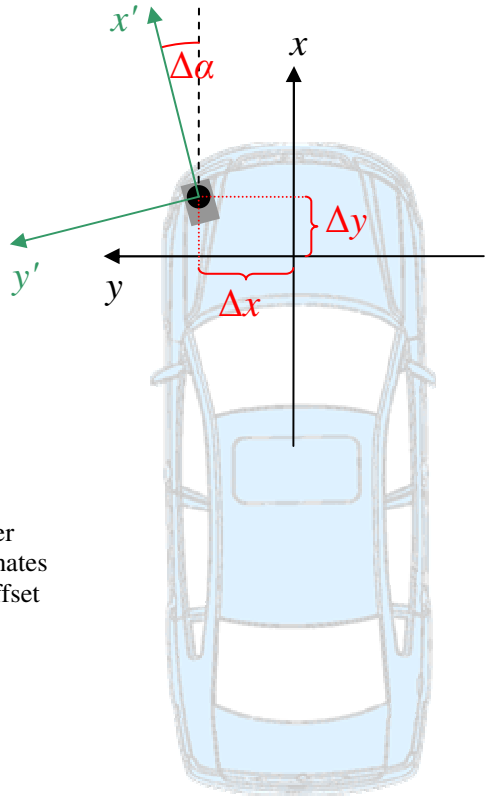


Fig. 38: For the transformation from laserscanner coordinates (green, cf. Fig. 35) to ISO 8855 coordinates (black), the displacement (Δx, Δy) and the angle offset $\Delta\alpha$ (red) must be known.

6.4.1 Manual determination

The *ASD* visualisation software from Ibeo allows setting the mounting position manually. The mounting position is entered in the *Parameter* property page of *ASD* (see Fig. 39).

- The offset in the ground plane (Δx, Δy) can be measured directly (e.g. using a tape measure). Often the offset can also be determined from CAD data. In the *Parameter* property page of *ASD*, assign Δx to the parameter "Offset X" and Δy to "Offset Y".
- To determine the horizontal angle offset $\Delta\alpha$ manually, place the vehicle next to a long plain wall (e.g. side of a house) so that the 0° direction of the vehicle is exactly parallel to the wall. In *ASD*, tune the parameter "Horizontal angle offset" until the scan data visualisation shows the wall exactly parallel to the *x*-axis of *ASD*.

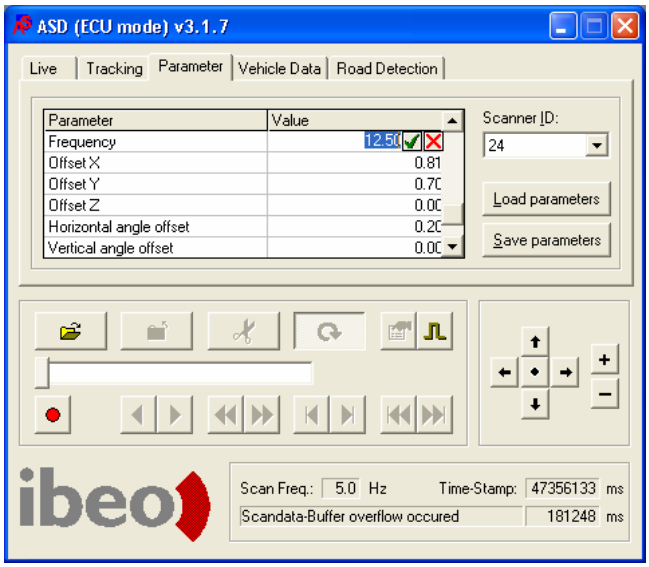


Fig. 39: The *Parameter* property page in *ASD* is used to set the mounting position of the laserscanner manually.

6.4.2 Automatic determination

Ibeo delivers a software tool called *ASystem-Setup* to detect the mounting position automatically. Note that the automatic mounting position detection requires time-consuming preparation so that you might prefer to set the mounting position manually as described in the previous section. The manual determination is usually faster than and as precise as the automatic determination.

Even in *ASystemSetup* there is an option to enter the offset (Δx, Δy) manually because in many cases this offset is known with high precision (e.g. from CAD data). *ASystemSetup* can use this knowledge to determine the horizontal angle offset $\Delta\alpha$ more precisely.

Fig. 40 shows the calibration field that *ASystemSetup* expects. The tip of the V-shaped reference target (red) must be located exactly at the point P_{ref} = (10 m, 0 m) in vehicle coordinates. Note that in contrast to Fig. 38, the origin is located at the middle of the rear axle.

If the offset (Δx, Δy) shall be determined automatically, too, the bisector of the V-angle must lie *exactly* on the x-axis (red dotted line in Fig. 40). Any angular error directly propagates from here to $\Delta\alpha$.

The main (and time-consuming) challenge of this procedure is to find the exact position of the reference point P_{ref} in front of the vehicle.

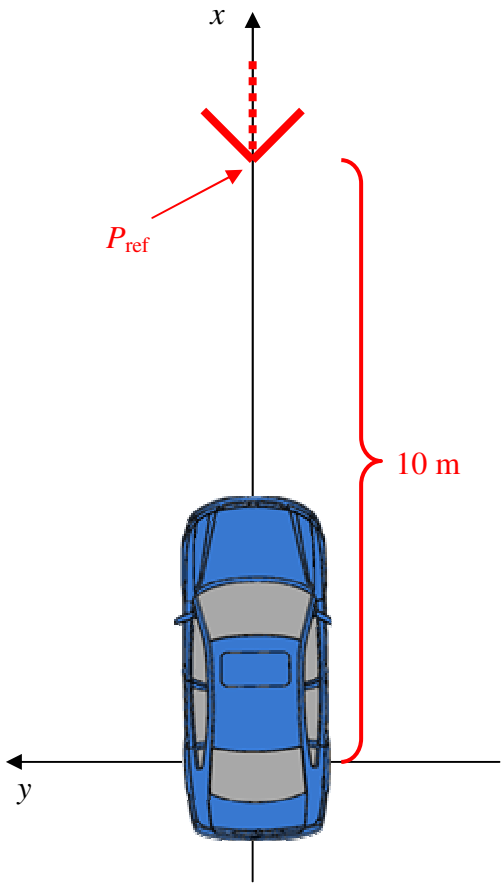


Fig. 40: Calibration field for the automatic mounting position detection with *ASystemSetup*

6.4.3 Vertical alignment

The multi-layer technology of the ALASCA is useful e.g. for pitch angle compensation (cf. Fig. 5). To work as expected, the laserscanner must be vertically adjusted so that it is aligned parallel to the ground plane. For this purpose, an adjustable mounting shoe is recommended as shown in Fig. 20. Two methods to adjust the vertical alignment are presented below.

Preparation to adjust the vertical alignment (see Fig. 41)

- Place the vehicle on flat ground and make sure the ground is level up to 10 m in front of the vehicle.
- Place the standard load in the vehicle, e.g. a driver and a passenger.
- Activate the laserscanner. It must be actively measuring. If the object tracking application *AppBase* is running on the ECU, it activates the laser automatically.
- Measure the height h_0 above ground of the beam at the position of the scanner. This is the distance from the ground to the middle of the mirror of the laserscanner.

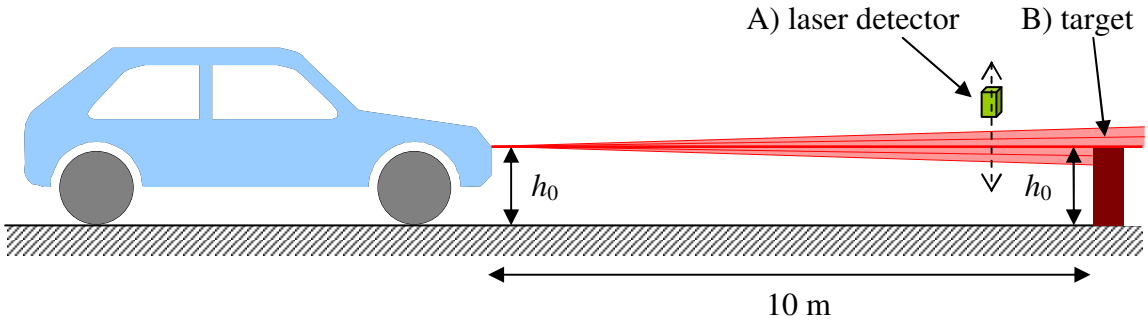


Fig. 41: Vertical alignment of the ALASCA using a laser detector (method A) or a simple reference target (method B). The red shaded area has a vertical divergence of 3.2°. The area is split up vertically into four channels of 0.8° divergence each (cf. Fig. 6).

Method A using a laser detector (available from Ibeo):

- While the ALASCA actively measures, use the detector to measure the height above ground of the beam centre in a distance of 10 m in front of the sensor (0° direction).
- Adjust the laserscanner so that the height of the beam centre is the same as the height h_0 directly at the laserscanner.
- Repeat the previous step for at least one different direction (e.g. 45° or 90°) to correct a possible roll angle of the sensor. After this correction, the scan plane should be parallel to the ground level and the mirror centre should lie inside the scan plane.

Method B using ASD and a reference target

- Construct a simple target of height h_0, e.g. a wooden log or board with a foot to stand alone. Put the target in a distance of 10 m in front of the sensor (0° direction).
- Adjust the laserscanner so that in the *ASD* visualisation only the lower two beams (default colours: red and blue) hit the target, whereas the upper two beams (green and yellow) hit the background or show no measurement at all. Note that there is no sharp border or a gap between adjacent beams. This makes the position detection procedure somewhat tricky because of "crosstalk" between adjacent beams.
- Repeat the previous step for at least one different direction (e.g. 45° or 90°) to correct a possible roll angle of the sensor. After this correction, the scan plane should be parallel to the ground level and the mirror centre should lie inside the scan plane.

6.5 Vehicle Model

Without further precautions, any non-linear ego-motion of the vehicle reduces the quality of the PreCrash information. For example, if two vehicles run the risk of colliding in a curve, the quality of the estimated time-to-collision (TTC) can get unacceptable low. Knowledge about the ego-motion helps to improve the TTC quality for non-linear motion. This knowledge comes from a vehicle model that describes the physical behaviour of the vehicle given its current velocity and steering angle. Using this model a non-linear ego-motion can be predicted better and the TTC quality is improved.

Furthermore, the known ego-motion can be subtracted from the other objects' motion to get their absolute velocities. These results are helpful input e.g. for object classification.

Fig. 42 shows the parameters of the vehicle model. If the position of the centre of gravity C is not known the centre may be shifted to C' half way between the centres of the front and rear axle. The error due to this displacement is acceptable for most applications.

The vehicle model takes as input the current velocity v and the front wheel angle α (ESP data). The latter one can be computed from the well-known angle of the steering wheel using a cubic polynomial as transfer function. Both, v and α, should be updated frequently (min. 10 Hz) to ensure a good accuracy of the model.

The model yields for the point C (or C') the changes of position (Δx, Δy) and yaw angle $\Delta\psi$ relative to the previous output data of the model.

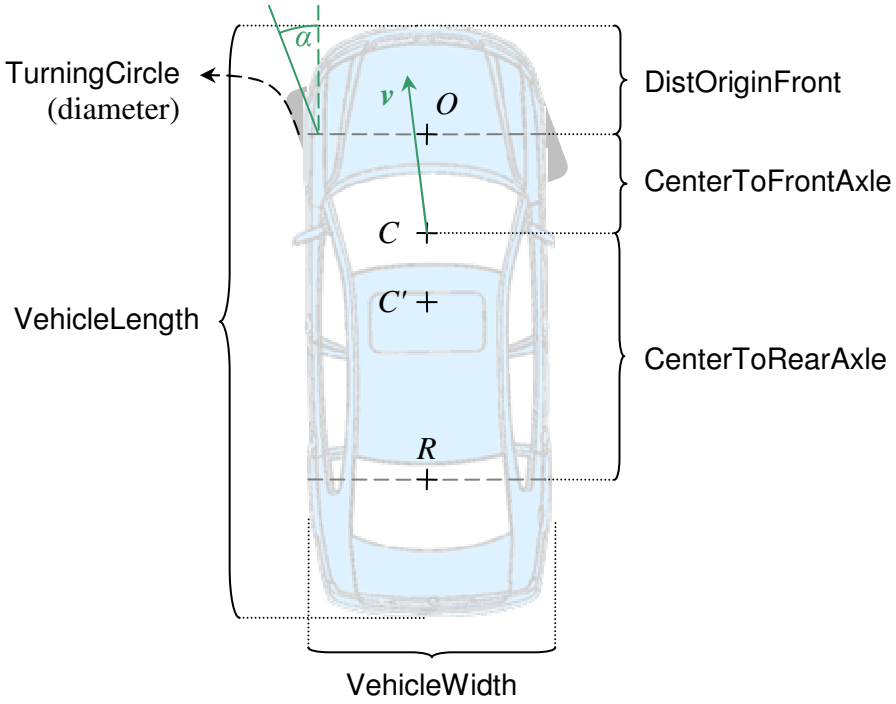


Fig. 42: Parameters of the vehicle model: O = centre of the front axle = origin, C = centre of gravity, C' = alternative centre = ½ (O + R), R = centre of rear axle. The vehicle width does not include the side mirrors.

7 Software

This chapter describes the individual software modules. The structure of this chapter resembles the data flow from the laserscanner to the application:

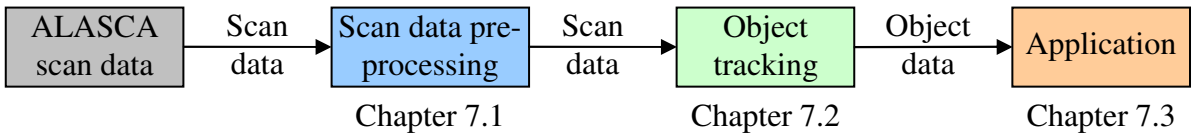


Fig. 43: Overview over the data flow from the sensor to the application

Please note that – depending on your system configuration – some of the following modules may be disabled or unavailable. This especially applies to the high-level modules or applications.

7.1 Scan data pre-processing

During scan data pre-processing, the scan data sent from ALASCA is filtered and corrected in many ways. For example, each individual measurement is checked for being dirt, rain, or ground. Coordinates have to be corrected according to the mounting position parameters and so on.

The following subsections give concise descriptions of the software modules that are part of the scan data pre-processing.

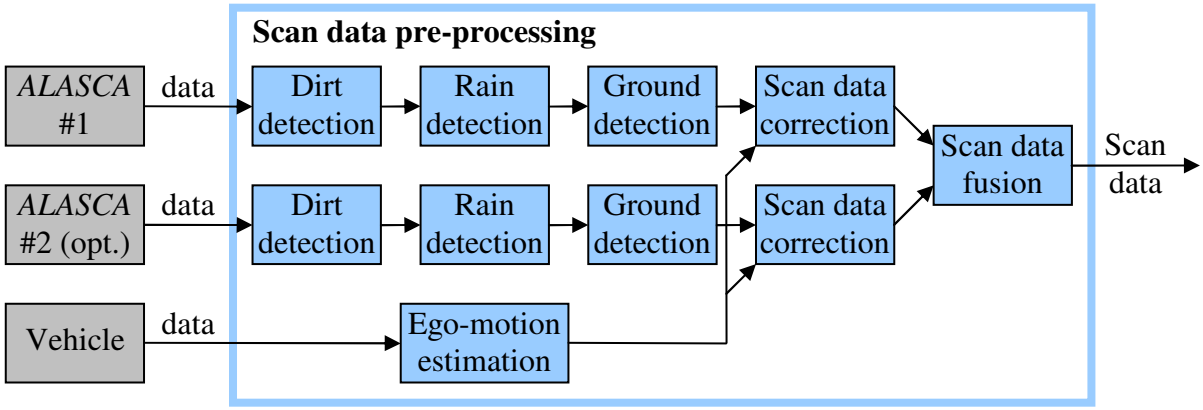


Fig. 44: Overview over the scan data pre-processing. The second ALASCA (middle row) is optional. Currently up to three ALASCA's can be merged at the scan data fusion. The scan data fusion module is available only if more than one ALASCA is connected.

7.1.1 Dirt detection and range estimation

If dirt accumulates on the cover of the integration chamber, the sensor's range of view will be reduced. In extreme cases (e.g. a layer of slush on the cover), the laserscanner even may get "blind". The vehicle computer and/or the driver must be informed about the reduced performance due to dirt. Therefore, a dirt detection module is included.

This software module defines dirt as fixed scan data close to the sensor up to 1.2 m. It recognises such fixed data and marks the sensor as being dirty. In this case all scan points up to 1.2 m are excluded from the object tracking and the "sensor dirty" flag in the *EnvironmentInfo* CAN message is set.

To verify this module the scan area can be partially covered by hand directly in front of the sensor. After five seconds (until scan data is recognised as being fixed), the dirt detection module signals a dirty laserscanner. Some seconds after removing the hand, the module signals a clean state again.

The dirt detection module also estimates the current range of view by averaging the farthest scan points of the most recent scans. This range estimation is printed as “view range” at the bottom of the *AppBase* window.

7.1.2 Rain detection

Rain may produce randomly scattered isolated scan points (like noise) in the near field in front of the sensor up to about 10 m. This range depends on the size of the rain drops and the intensity of the rain.

The rain detection module internally labels such scan points as rain. These points will be excluded from object tracking. The module also estimates a measure for the rain’s intensity. This measure is defined as the number of rainy scan points out of 1000 scan points. The intensity comes along with the *EnvironmentInfo* CAN message.

7.1.3 Ground detection

ALASCA’s multi-layer technology allows for ground detection (cf. Fig. 5). It is important for the application to distinguish between ground points and object points because ground must not be tracked as an object in automotive applications.

When scanning the ground, the different incident angles of the scan planes result in a characteristic scan pattern that is detected by the ground detection module. Scan points detected in this way are labelled internally as ground and are excluded from the object tracking.

7.1.4 Scan data correction

A scan is not an instantaneous snapshot of the sensor’s surroundings but rather takes some time to be measured (e. g. 25 ms for 180° scan area and 20 Hz scan frequency). The task of the scan data correction is to shift scan points back to a common point in time. This point is the time of the 0° measurement. For this computation the ego-motion is needed which is estimated by another module (see chapter 7.1.6). After scan data correction, the scan *is* an instantaneous snapshot.

Scan data correction becomes relevant if the sensor is moved or if the scan data of more than one laserscanner are merged (scan data fusion to overlay or extend scan areas).

Note that vehicle data must be supplied to the laserscanner system in order for this module to work.

7.1.5 Scan data fusion

If more than one laserscanner is integrated in the vehicle the individual scans have to be merged to one “big” scan before the object tracking can take place. This procedure is called *scan data fusion*. Since the scanner positions must be known for the merging of the scan data, the mounting position detection procedure must be done for the system (see chapter 6.4).

7.1.6 Ego-motion estimation

Ego-motion is the motion of the vehicle where the laserscanner is mounted. The ego-motion is estimated based on the data from the vehicle (velocity, steering angle) combined with a mathematical model of the vehicle's motion. The result of the ego-motion estimation is used by the scan data correction (chapter 7.1.4) and the computation of absolute velocities for objects tracked in the scan.

7.2 Object tracking

The object tracking modules implement a high-level abstraction from the scan data. They perform the transition from scan data (many individual measurements) to object data (condensed and interpreted information). Objects are structures in the scan data that have been recognised algorithmically as an independent entity. For each object, the geometric, dynamic, and qualitative properties are computed, e.g. position, velocity, contour, and classification.

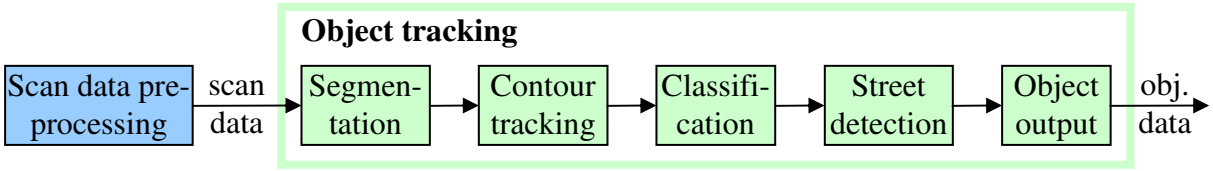


Fig. 45: Simplified overview over the object tracking.
For clarity, some internal modules have been omitted.

7.2.1 Segmentation

The segmentation procedure forms groups of scan points that are believed to belong together (e.g. due to geometrical correspondence). These groups are called segments. The separation of such segments is determined by a decision function, which is currently the distance between the scan points. (For details, refer to the parameters *MinDist* and *XYFactor* in chapter 6.3.2.) An object may be built up of many segments, but each segment may belong to at most one object.

7.2.2 Contour tracking

This module performs the actual object tracking based on the segments that have been found in the previous step. Each segment is considered as a part of one object's contour, hence by tracking segments we track object contours.

An object is tracked by first extrapolating its movement from the previous scan to the present scan. This prediction forms the seek area where in a second step the actual object position (in form of one or more segments) is found by a best match comparison.

7.2.3 Classification

Based on the properties of the objects the classification module labels each object with a type. Available classes/types are:

- Pedestrian,
- bike,
- car,
- truck,
- unknown big and unknown small.

This information is used by high-level applications that have to evaluate objects depending on their type. The object class is included in the object data of the CAN output.

Note: The classification is improved if vehicle data is available because then the objects' absolute velocities can be determined.

7.2.4 Street detection

The street detection module tries to estimate the course of the street ahead. It uses boundary objects and tries to find a clear path through these objects. At this process vehicles on the street are ignored.

The module yields the vehicle's position on the street (implying which lane the vehicle drives on), the street width and curvature, and the range of view available for street detection. These parameters are included in the CAN output.

7.3 Applications

Application modules evaluate all available data (scan, object, and vehicle data) to solve their tasks. In contrast to the object tracking the applications generate information that can be used directly for actions of the vehicle.

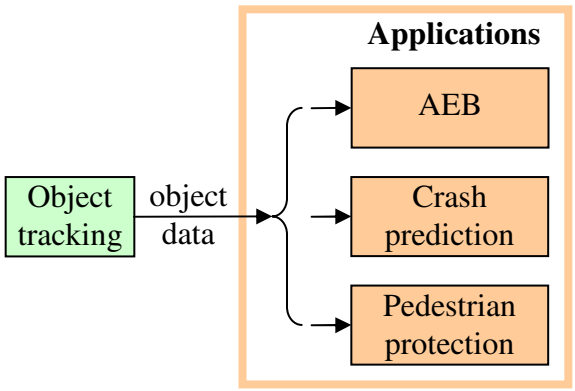


Fig. 46: Overview over some applications
(AEB = Automatic Emergency Braking)

7.3.1 Automatic Emergency Braking

Note: This application is an *optional* module that may not be part of the delivered software package. To order this module, please contact Ibeo. Moreover, the documentation in this chapter has also an exemplary character to give a deeper insight in one concrete application. Apart from AEB, Ibeo offers many other applications. Please visit the Ibeo homepage www.ibeo-as.com to get informed about more applications.

The entire laserscanner system, including Automatic Emergency Braking, is currently in prototype stadium. For safety reasons, never use this application in closed-loop on public roads!

The Automatic Emergency Braking application (AEB) is designed to reduce the threat due to frontal collisions. When AEB detects a possible crash situation, brakes are pretensioned automatically as a precaution to shorten the response time of the brakes. The AEB system will maximise the brake pressure if a collision cannot be avoided anymore. AEB aims at mitigation of the accident consequences by reducing the velocity at the time of collision.

The collision will explicitly not be avoided even though this would be possible to some extent with the current system. The reason for this approach is given by the driver as a human being. When an automatic emergency braking happens, the driver is overridden by an electronic system. It cannot be assumed that a driver instantaneously can continue driving after a collision has been avoided automatically. Consequently, the decelerated vehicle would move in the traffic as it was driverless or—after coming to standstill—would be a potential obstacle for the following traffic. These considerations give rise to legal questions concerning product liability.

In daily routine, the driver will not notice the AEB system because normally, AEB is only monitoring the surroundings for dangerous situations. As long as the traffic situation is fine, the AEB system is in the *all-clear* state. If the traffic situation becomes more dangerous and an obstacle is in the way-to-stop, the AEB switches to the *warning* state. In this state, the driver still can avoid the crash by steering, but time-consuming reversible safety measures (e.g. the brake booster pressure-up) can be prepared now. If also steering cannot avoid the crash anymore, the AEB state is set to *alarm* and an external system is signalled to initiate the emergency braking.

To check that the driver cannot avoid a crash (neither by braking nor by steering) the AEB algorithm assumes the following facts:

- The maximum cross acceleration that the driver can control while steering is constant (default: 5 m/s^2).
- Since the grip of the tyres is unknown for the way-to-stop ahead a constant coefficient of friction = 1 is assumed.
- The maximum deceleration is constant (default: -10 m/s^2).

The AEB algorithm takes vehicle and object data as input. From this information, the risk of a collision is assessed. This is done by a chain of object filters where each filter checks a certain AEB criterion. Objects that fail to pass a filter are ignored in the remaining steps of the filter chain. If any object passes all filters, a collision with this object is unavoidable and an emergency braking must be initiated. The following list explains the filter chain in detail. The filter parameters can be set in the section “[ANB]” of the file “AppBase.ini” (see example below).

1. AEB is deactivated if the own velocity is slow. In this situation, no serious damage due to collisions is expected. At such slow velocities, AEB might also interfere with other applications like e.g. automatic parking. The threshold is specified in m/s using the parameter **dEgoMinSpeed** (default: 2.78 m/s [= 10 km/h])
2. Ignore “young” objects. The number of scans while an object has been tracked is called the *object age*. AEB relevant objects must have a minimum age to be sure that their detection and tracking is stable. The corresponding parameter is **uMinObjectAge** (default: 5).
3. Ignore all objects that do not intersect with the driving path. The *driving path* is defined as the area in front of the vehicle that has the same width as the vehicle and runs parallel to the *x*-axis (driving direction, see ❺ in Fig. 47). All objects that lie completely outside the driving path are ignored by AEB. In other words, only those objects are kept that have at

least one scan point inside the driving path. Use the parameter **dVehicleWidth** [m] to set the width of the vehicle. The width does not include the side mirrors. Default: 1.75 m.

4. Ignore remaining objects directly in front of the vehicle. This filter is required to handle some rare situations: Imagine a long crash barrier next to a narrow road. While driving along the crash barrier it will be tracked as one object for a long time, i. e. it will not be filtered by step 2 due to its age. Now, if e.g. a bush suddenly penetrates through the crash barrier into the vehicle tube AEB will trigger an alarm because in the object data, the bush is merged with the crash barrier and this merged object intersects with the vehicle tube. To avoid this problem a "blind" range is installed directly in front of the vehicle (see ❷ in Fig. 47). The size of this range depends on the velocity of the vehicle. Therefore, it is specified in milliseconds, the so-called *system delay time*. The actual blind range results from the product of the system delay time and the current velocity. The parameter **uSystemDelayTime** [ms] defaults to 100 ms. It must not be greater than 10.000 ms.

5. Keep only those objects that have a distance less than the vehicle's way-to-stop (see ❸ in Fig. 47). This way is mainly composed of the way-to-reaction of the AEB system and the way-to-brake:

$$\text{Way-to-stop:} \quad x_{\text{stop}} = k \cdot (x_{\text{react}} + x_{\text{brake}} + x_0)$$

$$\text{Way-to-react:} \quad x_{\text{react}} = v\tau$$

$$\text{Way-to-brake:} \quad x_{\text{brake}} = v^2 / (2a_{x,\max})$$

k is a stretch factor that models the reduced deceleration when braking in a curve. As will be shown in the next step the AEB application checks if the vehicle can escape in a curve to the left or right of an obstacle. The parameter **nEscapeWayOffset** [%] specifies the percentage that is added to the normal (straight on) way-to-stop, i. e. $k = (1 + \textbf{nEscapeWayOffset} / 100)$. For instance, 0 % implies $x_{\text{stop}} = x_{\text{react}} + x_{\text{brake}} + x_0$ (the normal way-to-stop) and 50 % implies $x_{\text{stop}} = 1.5\ (x_{\text{react}} + x_{\text{brake}} + x_0)$. It must hold

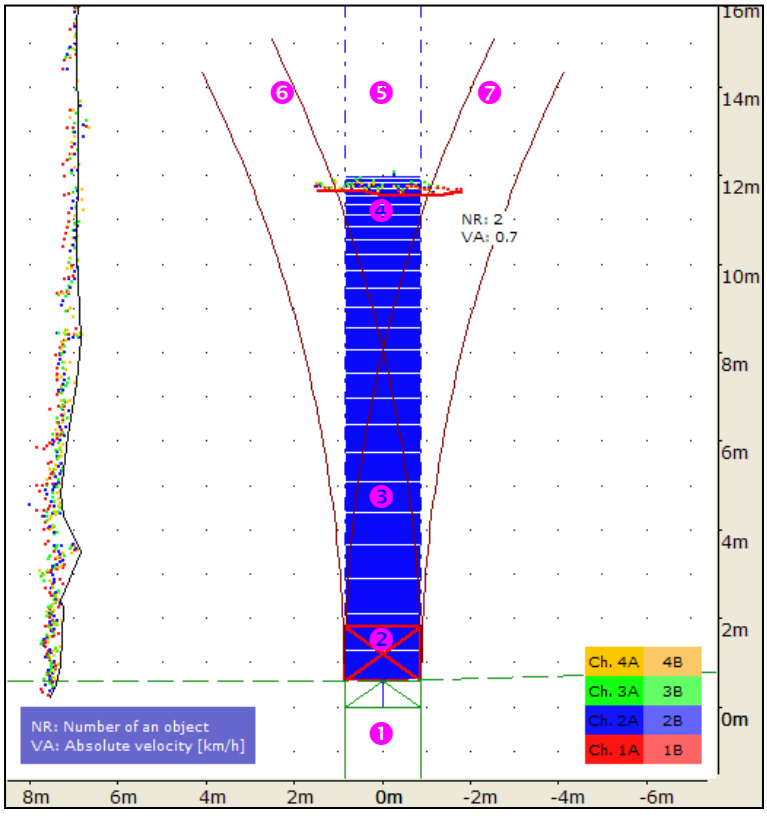


Fig. 47: Screenshot of the AEB application integrated in ASD. In this example a crash object ❹ is detected, and AEB has calculated that the crash is unavoidable by braking or steering, so AEB is activated.

- ❶ Vehicle outline (green), velocity approx. 35 km/h
- ❷ Blind range due to the system delay time (red)
- ❸ Way-to-stop (blue)
- ❹ Crash object (red outline). The object has tracking no. 2. Its velocity equals 0.7 km/h.
- ❺ Driving path (dashed lines)
- ❻ Circular curve with the left escape radius
- ❼ Circular curve with the right escape radius

$-100 \leq$ **nEscapeWayOffset** $\leq +100$. The default value is 25 %.

x_0 is the offset between the front of the car and the origin of the vehicle's coordinate system. It can be set with the parameter **dDistOriginFront** [m] (cf. Fig. 42, default: 0.6 m).

v is the vehicle's current velocity.

τ is the *system dead time* which is the sum of the parameter **uSystemDelayTime** from the previous step and the parameter **uSystemPreFiredTime** [ms]. The latter one is the time between triggering an AEB alarm and the beginning of deceleration (default: 300 ms, must not be greater than 10.000 ms).

$a_{x,\max}$ is the maximum longitudinal acceleration when fully braking. $a_{x,\max}$ is set by the parameter **dMaxAcceleration** [m/s^2] and must be a negative value (i. e. deceleration). The default value is –10 m/s^2. Note that $a_{x,\max}$ is constant so that this parameter does *not* reflect the maximum possible deceleration given the tyres' current grip—simply because this grip is unknown. Since an AEB application brakes only if a crash is unavoidable $a_{x,\max}$ must be set to the maximum deceleration possible at best grip.

6. Keep only those objects where the vehicle can escape neither to the left nor to the right of the object. To check this, the current left and right escape radius is computed. The *escape radius* is the smallest radius of a (circular) curve that the vehicle can take at the current velocity (see ❹, ❻, and ❼ in Fig. 47). It is computed using the formula $r_{\text{escape}} = v^2 / a_{y,\max}$ where $a_{y,\max}$ is the maximum lateral acceleration. This constant can be specified by the parameter **dMaxCrossAcceleration** [m/s^2] (default: 5 m/s^2). As in the previous step here again a constant acceleration is assumed that is available only at best grip. Since the escape radius cannot be less than the radius of the vehicle's turning circle this limit can be configured using the parameter **dTurningCircle** [m] (default: 11 m).

The following lines show an exemplary ANB section of the "AppBase.ini" file with the default parameters:

```
; Parameters for the AEB application
[ANB]       ← not AEB for historical reasons
dEgoMinSpeed = 2.78
dMaxObjectSpeed = 1.0
uMinObjectAge = 5
dVehicleWidth = 1.75
uSystemDelayTime = 100
nEscapeWayOffset = 25
dDistOriginFront = 0.6
uSystemPreFiredTime = 300
dMaxAcceleration = -10.0
dMaxCrossAcceleration = 5.0
dTurningCircle = 11.0
```

Two requirements must be met for the AEB application to work:

1. The ANB section of the "AppBase.ini" file must contain valid settings, i. e. name spelling must be correct and values must be plausible. If parsing of the ANB section fails, a warning will appear in the *AppBase* dialog window and in the log file, and the AEB application is aborted. Care must be taken that *all* parameters are set correctly, since there is no error message in case of missing or redefined parameters. Missing parameters will be set to their default values without notice.

2. Vehicle data must be sent regularly, at least once a second. Otherwise an error message appears in the *AppBase* dialog window and AEB will not work. It is recommended to

update the vehicle data at 10 Hz. Higher update rates do not improve the quality, but may cause unwanted delay because all incoming messages must be evaluated.

If any object passes all AEB filters a collision with this object is unavoidable and the AEB application initiates an emergency braking by sending a CAN message. After the application is started successfully, AEB sends CAN messages periodically with the scan frequency. All messages have the ID = (*CAN_base_ID* – 1) and are two bytes long. The following messages are available:

AEB_ALL_CLEAR (Message: ID = *CAN_base_ID* – 1; length = 2; data = 0x00, 0xFF): This is the standard AEB message. It indicates that the application is alive and that the AEB filters have detected no potential crash object.

AEB_WARNING (Message: ID = *CAN_base_ID* – 1; length = 2; data = 0x55, 0xAA): AEB has detected an object inside the driving path that is closer than the way-to-stop but there is still a chance of escaping to the left or right of the object. This warning informs that a crash might happen soon. Therefore, measures should be taken that reduce the reaction time in the case of an actual crash; e.g. the brake booster can be pretensioned. This message is repeated as long as the potential crash object is detected.

AEB_ALARM (Message: ID = *CAN_base_ID* – 1; length = 2; data = 0xAA, 0x55): AEB has detected a crash object so that an emergency braking must be initiated. This message may be sent without a prior AEB warning because of a sudden change in the traffic situation. This message is repeated as long as the crash object is detected.

Note that if AEB is not active, e.g. because the parameter read has failed or no vehicle data is available, no messages will be sent. In this case, a warning will be given both in the dialog window of the *AppBase* and in the log file.

7.3.2 Other applications

As stated at the beginning of the previous section, AEB is only one exemplary application for the ALASCA laserscanner. Other applications are available from Ibeo, too:

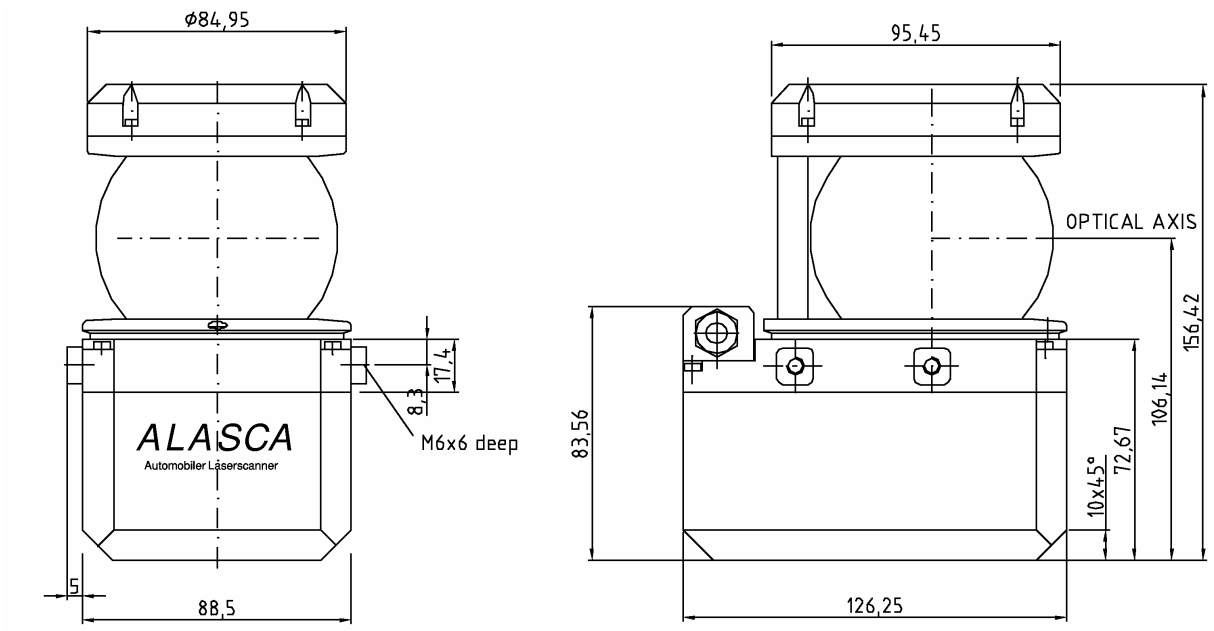
- Automatic Stop & Go
- PreCrash
- Pedestrian Protection
- Collision Warning
- Turing Assist
- Park Assist
- …

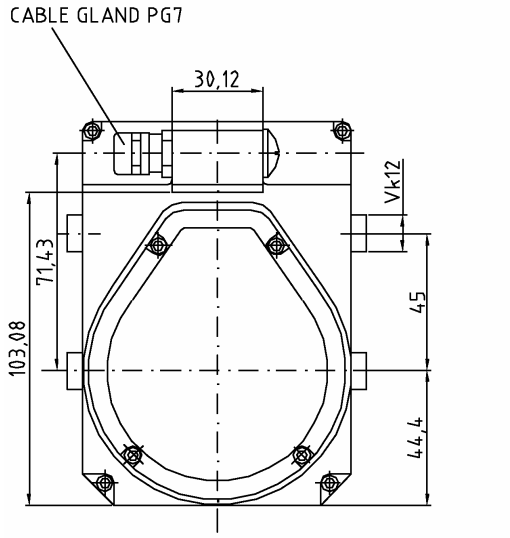
Please visit www.ibeo-as.com to get informed about the latest applications for your laserscanner.

8 Physical dimensions

8.1 ALASCA

All measures are given in millimetres.



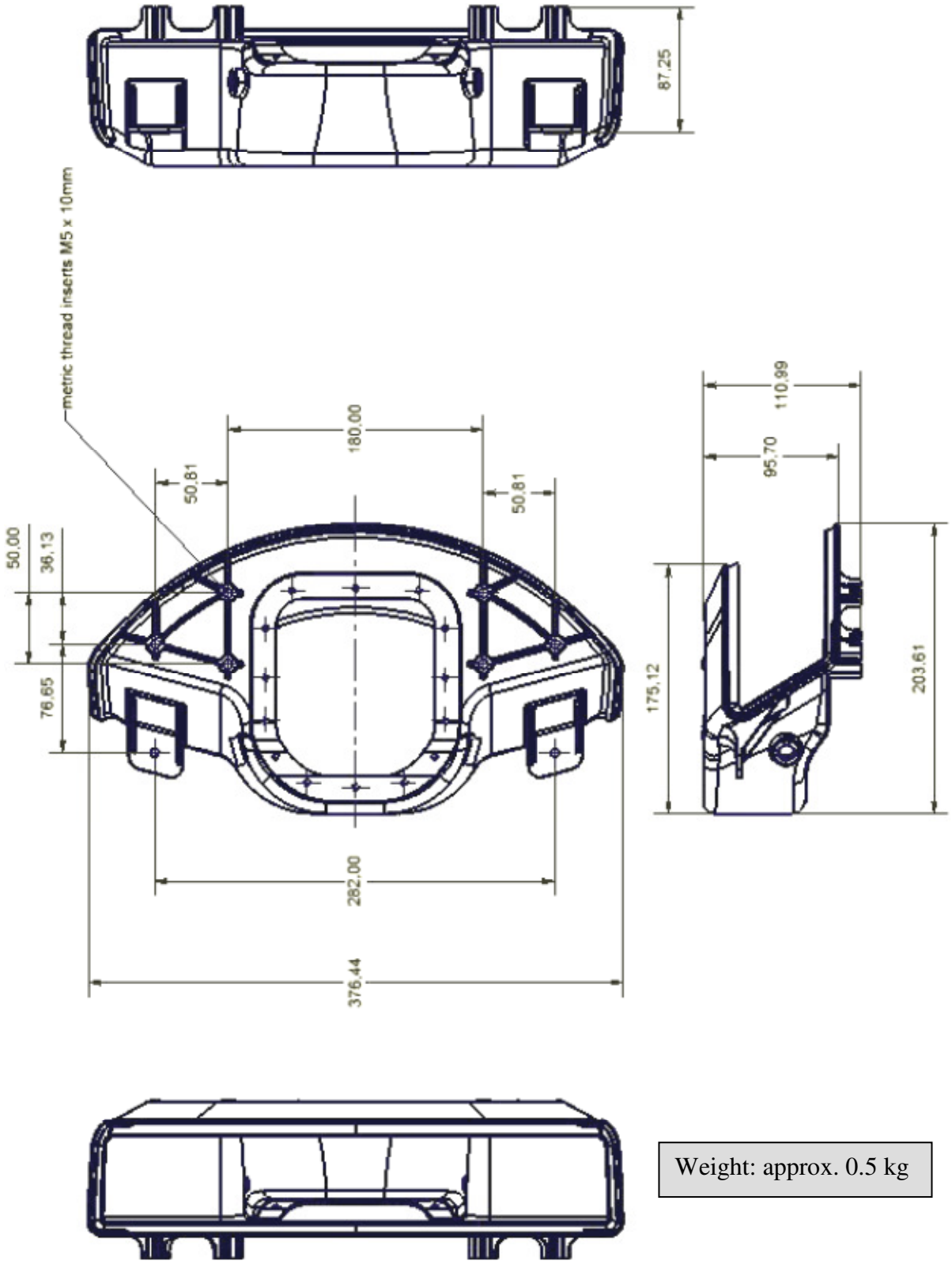


Weight: approx. 1.5 kg, including the flexible rubber seal (not shown here)

8.2 Standard integration chamber

8.2.1 Housing

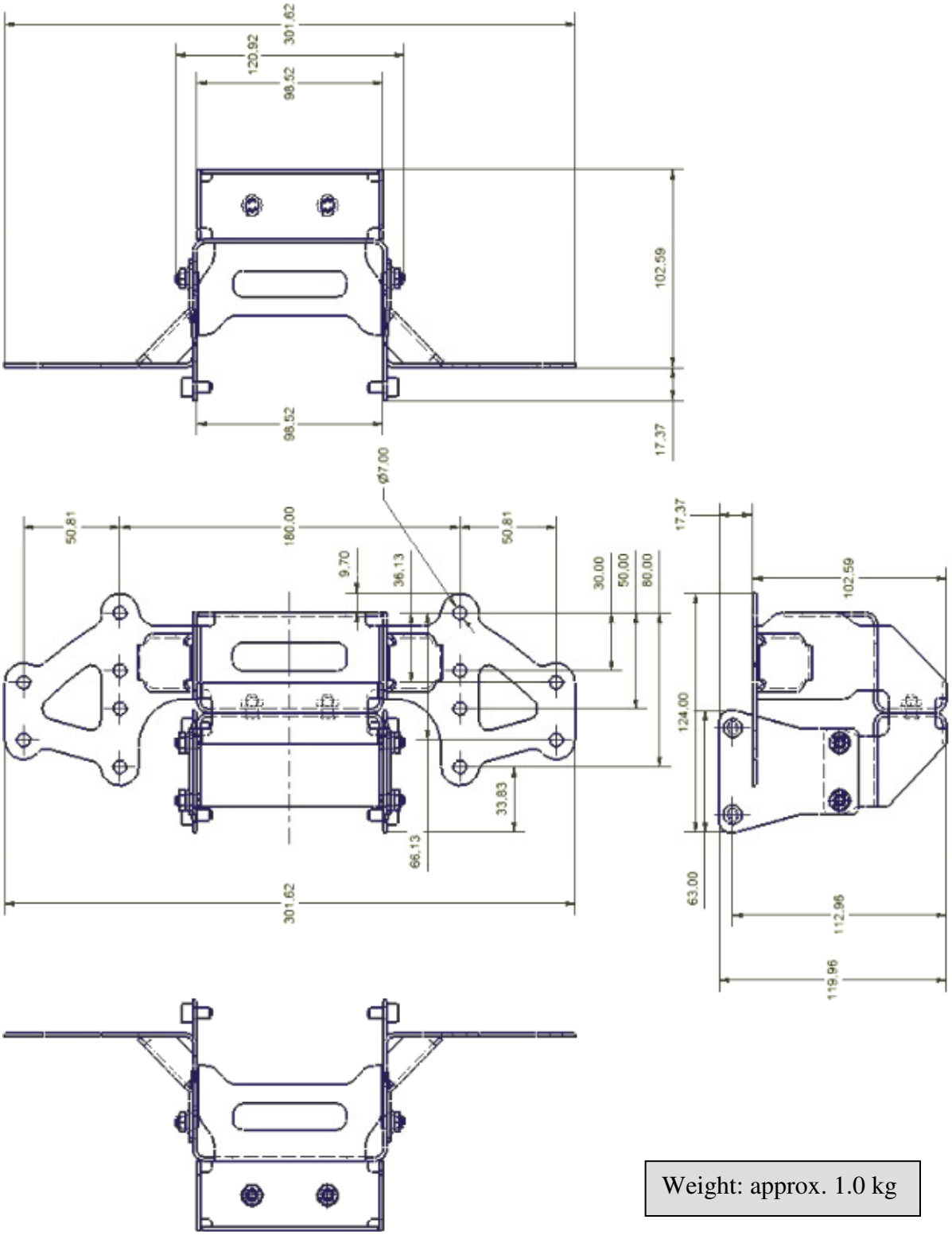
All measures are given in millimetres.



Weight: approx. 0.5 kg

8.2.2 Mounting shoe (holder)

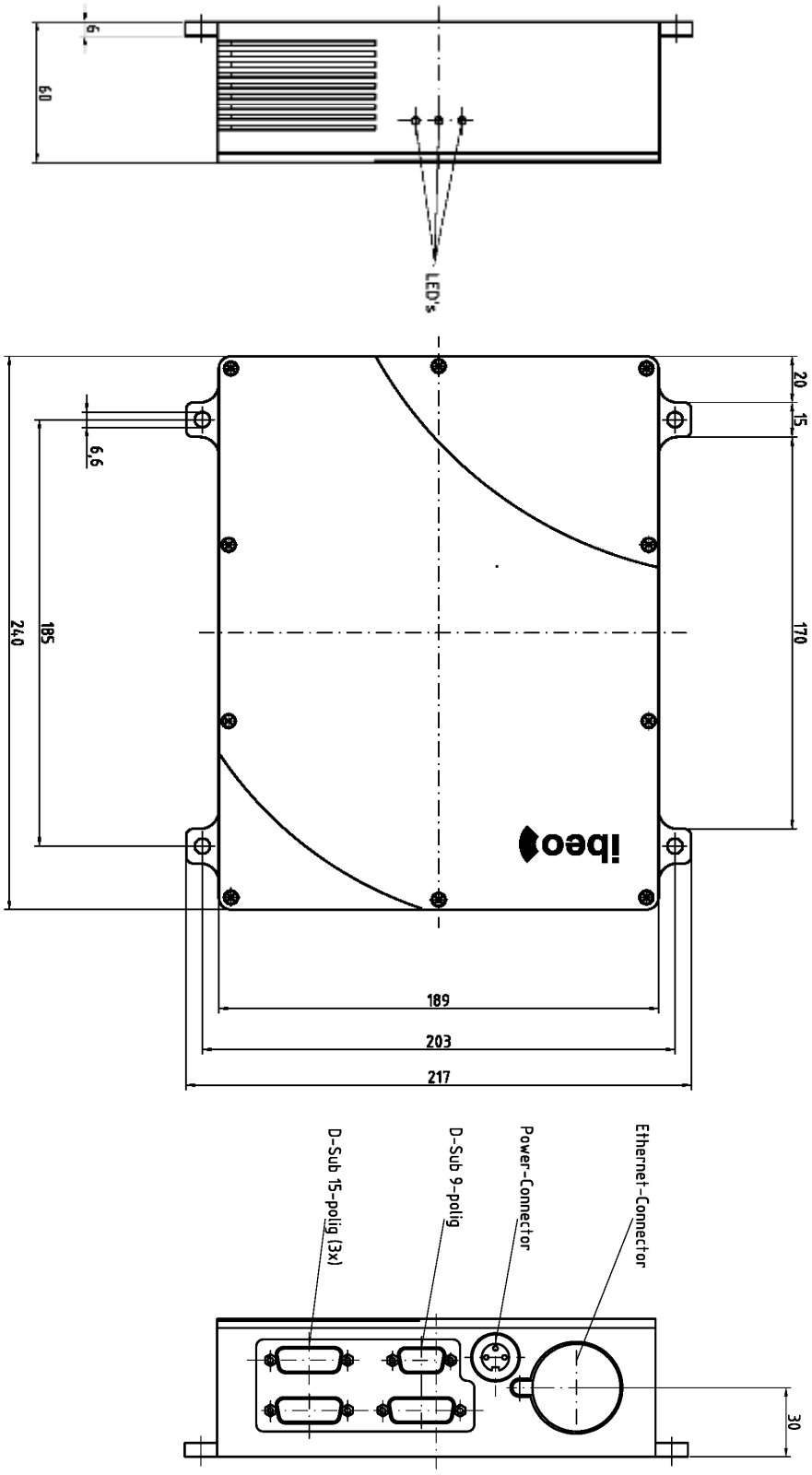
All measures are given in millimetres.



Weight: approx. 1.0 kg

8.3 ECU

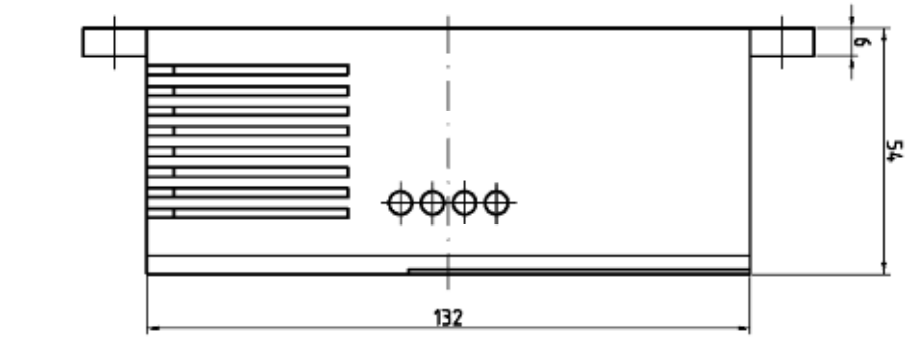
All measures are given in millimetres.

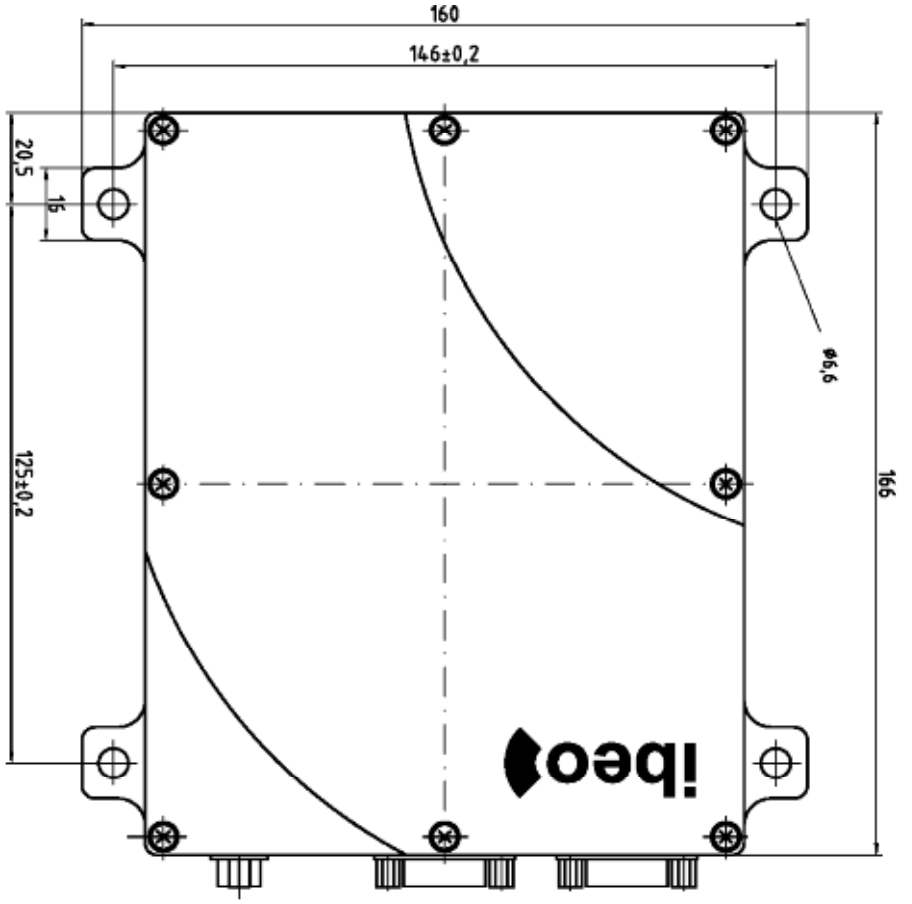


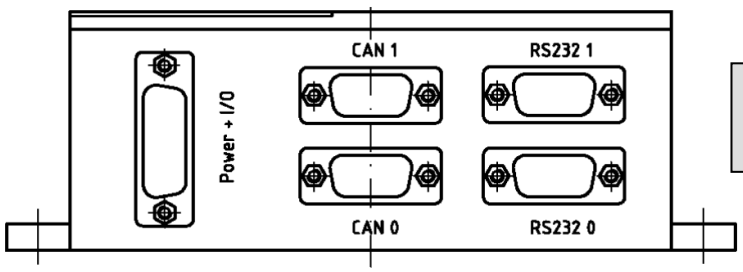
Weight: approx. 2.6 kg

8.4 SyncBox

All measures are given in millimetres.







Weight:
approx. 1.1 kg

9 References

[1a] European Committee for Electrotechnical Standardization: “EN 60825-1: Safety of laser products – Part 1: Equipment classification, requirements and user’s guide” (IEC 60825-1:1993 + A2:2001); Internet: www.cenelec.org

[1b] Deutsches Institut für Normung e.V.: „DIN EN 60825-1: Sicherheit von Laser-Einrichtungen – Teil 1: Klassifizierung von Anlagen, Anforderungen und Benutzer-Richtlinien“; Internet: www.din.de

[2] Ibeo Automobile Sensor GmbH: “ARCnet Documentation”

[3] Ibeo Automobile Sensor GmbH: “Specification of the CAN message protocol for Ibeo Automobile Sensor GmbH Laserscanners”

[4] Till Heinrich: „Bewertung von technischen Maßnahmen zum Fußgängerschutz am Kraftfahrzeug“, Technische Universität Berlin, ILS, August 2003

[5a] International Organization for Standardization: “ISO 8855:1991: Road vehicles – Vehicle dynamics and road-holding ability – Vocabulary”; Internet: www.iso.org

[5b] Deutsches Institut für Normung e.V.: „DIN 70000: Straßenfahrzeuge; Fahrzeugdynamik und Fahrverhalten; Begriffe“; edition 1994-01 (ISO 8855:1991, modified); Internet: www.din.de