CAN message protocol

Ibeo Automobile Sensor GmbH

Version: 1.7.0
Date: 07.08.2007

Contents

1 Introduction

1.1 Disclaimer

This description has been written taking great care. However, this documentation may contain errors that have yet gone unnoticed. If such an error is found, please report it to IBEO Automobile Sensor GmbH, and it will be corrected in the next release. However, please note that IBEO Automobile Sensor GmbH provides this documentation as-is, and cannot be held liable for any consequences resulting from the usage of the contents described herein.

This paper describes the new Ibeo AS CAN specification. All Laserscanners support this protocol. The general idea behind this protocol is for each sensor to communicate using only two CAN identifiers: One for communication *to* the sensor and one *from* the sensor to the host. These identifiers are set to a factory default upon delivery, and can be changed, as well as every other parameter, by the user to adapt the Laserscanner to the individual system, thus allowing a maximum of flexibility.
All Ibeo AS Laserscanners use the CAN v2.0A (11-bit identifiers) for communication.

Note: This document uses "Laserscanner" or "Sensor" meaning the complete Laserscanner system consisting of one or more Laserscanners and the ECU, which provides the user interface.

1.2 General information

Information on the laser scanner system as a whole is contained in the laser scanner manual. You are recommended to read the laser scanner manual first because it explains the system, how the measured values are depicted and the concept of object tracking.

1.3 General explanations

For each Laserscanner, two CAN identifiers are exclusively assigned. With the first one (called the sensor specific CAN Base ID), the sensor receives instructions from the host, and with the second one (which is always CAN Base ID+1) it can send data to the host. In addition, there are some “broadcast“ identifiers which are received by every Laserscanner (e.g. synchronisation).

1.3.1 How to transfer bigger amounts of data

To carry more than 8 bytes of data with one CAN identifier, the principle of mode-signal/sub-identifier is used. In this case, the meaning of the data of a CAN message is coded by some value which is included in the message. E.g., a message with a first data byte of “1“ would be a calibration command, and with a first byte of “2“ would be a parameter setting command. The CAN messages are explained in detail in the following sections.

Note that every CAN message may carry 8 bytes of data, even if the contents themselves are shorter. This convention was made to simplify data transfer inside the application(s).

1.3.2 Data formats

Wherever multi-byte values are to be transmitted, the “Big Endian“ notation is used throughout this document. This means that values are coded MSB-First.
Used data types are:

UINTx — An x-bit unsigned integer value.
Examples: UINT4 = 0x0 .. 0xF, UINT16 = 0x0000 .. 0xFFFF.

INTx — An x-bit signed integer value in 2‘s complementary notation. Examples:
–1 = (INT4) 0xF, –1 = (INT16) 0xFFFF, 7 = (INT8) 0x07, –3 = (INT8) 0xFD

FLOAT8 — An 8-bit decimal value with the value range –12.8 .. +12.7.
Conversion is done like in the following C source code:

```
unsigned char ucInput = ...;    /* 0 <= ucInput <= 255 */
float fOutput = ((int) ucInput - 128) / 10.0;
```

For example, ucInput = 0x62 converts to (98 – 128) / 10 = –3.

FLOAT16 — A 16-bit decimal value with the value range –327.68 .. +327.67. Conversion is done like in the following C source code:

```
unsigned short usInput = ...;    /* 0 <= usInput <= 65535 */
float fOutput = ((int) usInput - 32768) / 100.0;
```

For example, usInput = 0x7448 converts to (29768 - 32768) / 100 = -30.

Used data type codes are:

0: INT16	2: UINT8
1: FLOAT16	3: UINT16
	4: UINT32

1.3.3 Coordinate system

All position and velocity data from and to the Laserscanners is related to a car stationary coordinate system according to DIN 70000. The origin of this coordinate system is placed in the middle of the front axle of the car. The x-axis points in driving direction (pos. values), and the y-axis points to the left side. Angles are defined in the normal mathematical way, positive angles turn counter clockwise (cf. Fig. 1). All angles are sent and received in degrees.

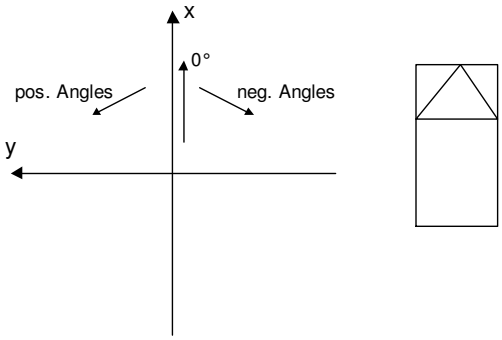


Fig. 1: Used coordinate system.

1.3.4 Mounting position

After power-on, every Laserscanner system starts sending object data immediately. The position offset values and the parameters are automatically restored after power on. If the offset values are not set, the origin of the used coordinate system is the sensor itself. Please use the “ASystemSetup”[1] software to determine the offsets and also the view area of the Laserscanner.
Offsets:

- The Laserscanners position at the vehicle in both x and y direction, and
- its yaw angle offset (offset of the scanners x axis and the vehicle driving direction).

1.3.5 Default parameters

Each Laserscanner system is delivered with an identical set of parameters (“factory setting“). A list of those parameters is given in chapter 6.

[1] ASystemSetup is a part of the Ibeo AS software collection and is used fort setting up the Laserscanner.

2 Identifier allocation

Default CAN identifiers for communication with Ibeo AS Laserscanner systems:

Identifier	Name	Description
0F0h*	Synchronisation	System-wide timestamp
4EAh	AEB++	(CAN Base ID – 6), Application specific, reserved
4EBh	ACC	(CAN Base ID – 5), Application specific, reserved
4ECh	PedestrianProtection	(CAN Base ID – 4), Application specific, reserved
4EDh	CrashPrediction (CP)	(CAN Base ID – 3), Application specific, reserved
4EEh	WarningZones (WZ)	(CAN Base ID – 2), Application specific, reserved
4EFh	AEB	(CAN Base ID – 1), Application specific, reserved
4F0h	Scanner commands	CAN Base ID
4F1h	Scanner data	(CAN Base ID + 1)

Note: *Identifiers marked with a „*“ are fixed and cannot be changed by the user. The CAN Base ID is user configurable.*

3 Broadcast instructions

3.1 Timing synchronisation

Each Laserscanner system uses a millisecond timer. It starts with a value of 0 after start-up. The current time is included in all data sent from the system. To synchronise the system with other components, this clock can be set to any value using the “Timestamp“ message.
Typically, the synchronisation message is sent system-wide by a central control unit with a fixed cycle, e.g. once a second, to avoid the clocks drifting apart.

Message:	Synchronisation broadcast (Timestamp)		
Identifier:	0F0h		
Length:	4 byte	Transmission rate:	-
Receiver:	all	Retransmission time:	1 sec (recommended)

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Timestamp	3..0	31..0	0.. 4294967295 ms	1 x value	UINT32	
	7..4	63..32				(unused)

Note: Since a 32 bit timer is used, the time will be reset to 0 in case of an overflow after 2^{32} ms (about 7 weeks). To avoid this, do not set the time to unnecessarily high values. A contiguous run over multiple weeks is not intended.

4 Sensor instructions

Sensor instructions are 8-byte messages sent on the Base ID. They are used to send commands to the sensor, but also to send configuration data and set parameters. In the following tables, some rows will be grey-coloured. Those rows contain fixed values which cannot be changed in the given configuration, e.g. a command op-code (which obviously cannot be changed for this command).

Note: *The words "instruction" and "command" are used with identical meaning throughout this document.*

Message:	Sensor instructions		
Identifier:	Base ID (default: 4F0h)		
Length:	8 byte	Transmission rate:	-
Receiver:	single Laserscanner	Retransmission time:	once

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Instruction type	0	3..0	0..15		UINT4	Mode signal
Data	7..0	63..4				Mode-depending data

Instructions are:

Instr. type	Instruction	Description
0	Calibration with given values	Set the calibration parameters to the values sent in this message.
1	Calibration in the calibration field	Do the automatic calibration in the calibration field with reference objects.
2	Calibrate with stored values	Restore the values from the last successful calibration.
3	Set Parameter	Set a parameter to the value given in this message.
4	Query parameter	Asks the sensor for the value of a parameter. Should be used to verify the parameter setting.
5	Store current parameter set	Store the currently used parameter set in the non-volatile memory and restore those parameters at ever start-up.

4.1 Calibration (Mounting position setup)

Instruction type: 0..2 → ***Deprecated!*** **See notes below.**

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Instruction type	0	3..0	0..15	0: Calibration with the values of this message 1: Calibration in calibration area 2: Calibration with values from memory	UINT4	Message may be repeated until sensor has set his calibration flag or indicates failure
OffsetX	1..0	13..4	-5,1..5,13 m	0,01 x value – 5,1	UINT10	x-position of the sensor related to the origin
OffsetY	2..1	23..14	-5,1..5,13 m	0,01 x value – 5,1	UINT10	y-position of the sensor related to the origin
OffsetZ	4..3	33..24	-5,1..5,13 m	0,01 x value – 5,1	UINT10	z-position of the sensor related to the origin
Angle Offset	5..4	43..34	-180°..180°	0,5 x value – 180	UINT10	Angle description see DIN 70000
Pitch angle	6..5	53..44	-180°..180°	0,5 x value – 180	UINT10	Angle description see DIN 70000
Roll angle	7..6	63..54	-180°..180°	0,5 x value – 180	UINT10	Angle description see DIN 70000

Notes:

- Calibration is an obsolete name for mounting position detection. Do not use this command anymore, since it is deprecated. Please use the Ibeo "ASystemSetup" instead, as described in section 1.3.4.
- The calibration values can also be set using the Set Parameter command.

4.2 Set Parameter

Instruction type: 3 *Set Laserscanner parameter*

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Instruction type	0	3..0	0..15	3: set parameter	UINT4	
Parameter	1	15..8	0..255	(see chapter 6)	UINT8	(see chapter 6)
Data type	2	23..16	0..255	0: INT16 1: FLOAT16 2: UINT8 3: UINT16 4: UINT32	UINT8	
Data value	4..3	39..24	(see chapter 6)	(see chapter 6)	(see chapter 6)	(see chapter 6)
	7..5	55..40				(unused)
Sensor ID	7	63..56	0..255		UINT8	(see chapter 6)

The Laserscanner does not acknowledge the reception of this command. To ensure that it has accepted the parameter, read the parameter to verify its value.
If the message length is less then 8 bytes, the `SensorID` is assumed to be 0.

4.3 GetParameter

Instruction type: 4 *Request for a Laserscanner parameter*

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Instruction type	0	3..0	0..15	4: request a parameter	UINT4	
Parameter	1	15..8	0..255	(see chapter 6)	UINT8	(see chapter 6)

After reception of this command, the Laserscanner answers with a `Parameter` message (see chapter 5.3 for details).

4.3.1 Laserscanner parameter save command “Store Parameters“

Instruction type: 5 *Store current parameter set*
Saves the current parameter set into the sensor-internal FLASH memory. The previously saved parameters are overwritten. The stored values are now restored automatically as default parameters after every sensor start-up. The calibration parameters are automatically saved *and* restored.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Instruction type	0	3..0	0..15	5: Store parameters	UINT4	
Parameter	1	15..8	0..255	0: Normal save	UINT8	0 = Normal save to FLASH
	7..2	63..16				(unused)

5 Sensor data

This section explains the data which are sent from the Laserscanner to the host. Typically, this will be object data, but also parameters.
For the data transfer, the message-ID “Base ID + 1“ is used.

5.1 Object data (LONG format)

Internally, the Laserscanner builds an object list which is sorted by the current quality criterion, e.g. the radial distance of the objects from the sensor. This object list is then transmitted to the host. This transmission starts with a *list header* message which contains general information. Then, the objects are sent in *object message* messages. Finally, the list ends with a *list end* message. Now, the consistency of the received data should be checked to ensure data integrity by checking that both the *list header* and the *list end* contain the same cycle counter and that the checksum is correct. If this test is omitted, messages from different scans could have been mixed due to message loss on the CAN bus.
As before, the message type is encoded in the first bits of the messages (only in this case, only 3 bits are used).
To transfer other data such as parameters, a special version of the *list header* message is used. In this case, the object number will contain a non-zero value which indicates the contents of the message (sub-coding of the *list header* message).

Message:	Sensor object data		
Identifier:	Base ID + 1 (default: 4F1h)		
Length:	8 byte	Transmission rate:	-
Receiver:	single sensor	Retransmission time:	sensor depending

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	0: reserved 1: List – header 2..6: object message 7 : list – end	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	
Data	7..1	63..8				Depending on Mode and Object number

5.1.1 Message type 1: List header

Note: *This message is only a list header if the object number is 0.*

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	1: list header	UINT3	Mode signal
Object number	0	7..3	0..31	0: normal list header	UINT5	Must be zero for a list header
Object style	1	7..0	0..255	0-127: (reserved) 128: IBEO AS data	UINT8	Must be 128 for this spec
Number of objects	2	7..2	0..31	1 x value	UINT6	
Reserved	2	1	0..1	-	UINT1	Always 1, ignore
Calibration flag	2	0	0..1	0: not calibrated 1: calibrated	UINT1	
Cycle counter	3	7..0	0..255	1 x value	UINT8	Cycle will increase by 1 with each scan. Overflow from 255 to 0
Timestamp	4..7	31..0	0 .. 4.294.967.295 (0 .. ≈ 7 weeks)	1 ms x value	UINT32	Moment of beginning of the measurement

5.1.2 Message type 1: Environment info

Note: *This message is only a Environment info if the object number is 1.*

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	1: list header	UINT3	Mode signal
Object number	0	7..3	0..31	1: Environment info	UINT5	
Sensor dirty	1	7..0	0..255	0 = ok 1 = dirty (must be cleaned) 255 = not available	UINT8	
Rain detection	2	7..0	0..255	0 = No rain visible 1..254 = rain points per 1000 255 = not available	UINT8	
Dirt Detection: Start angle	3	7..0	0..255	0..180: Start angle = 2*value – 180.0, 254: No start angle, 255: Not available	UINT8	Ignore.
Dirt Detection: End angle	4	7..0	0..255	0..180: End angle = 2*value, 254: No end angle, 255: Not available	UINT8	Ignore.
	5-7					(unused)

5.1.3 Message type 2: General object info and velocity

Each object is proceeded by an *General object info* message.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	2: General object info	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Object ID	1	7..0	0..255	1 x value	UINT8	For the life of an object constant number (see below for explanation)
Tracking status	2	7..0	0..15	0: known object 1: unknown object 255: not available	UINT8	1 if this object is sent for the first time
Classification	3	7..0	0..255	1 x value	UINT8	Ref. to chapter 7
Number of points	4	7..0	1..256	1 x value + 1	UINT8	Number of contour points of an object
VelocityX	5	7..0	-63,5.. 62,5 m/s	0: value < -63,5 m/s 1..253: 0,5 x value – 64 254: value >= 62,5 m/s 255: no value available	UINT8	Relative velocity in X direction
VelocityY	6	7..0	-63,5.. 62,5 m/s	0: value < -63,5 m/s 1..253: 0,5 x value – 64 254: value > 62,5 m/s 255: no value available	UINT8	Relative velocity in Y direction
Velocity X ext.	7	7..4	-0.25.. 0,21 m/s	Value / 32	INT4	For better resolution: Add this value to the Velocity X
Velocity Y ext.	7	3..0	-0.25.. 0,21 m/s	Value / 32	INT4	For better resolution: Add this value to the Velocity Y

5.1.4 Message type 3: Object deviation info

Each object is proceeded by an *Object deviation info* message.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	3: Object deviation info	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Relative moment of measure	1	7..0	0..254 ms	1 x value 254: >= 126 ms 255: not available	UINT8	Moment of measure relative to begin of cycle
PositionX sigma	2	7..0	0..25,5 m	0..253: 0,10 x value 254: >= 25,5 m 255: no value available	UINT8	Standard deviation of value PositionX
PositionY sigma	3	7..0	0..25,5 m	0..253: 0,10 x value 254: >= 25,5 m 255: no value available	UINT8	Standard deviation of value PositionY
VelocityX sigma	4	7..0	0..15 m/s	0..25: 0,5 x value 30: >= 16 m/s 255: no value available	UINT8	Standard deviation of value VelocityX
VelocityY sigma	5	7..0	0..15 m/s	0..25: 0,5 x value 30: >= 16 m/s 255: no value available	UINT8	Standard deviation of value VelocityY
Position correlation coefficient	6	7..0	-1..+1	0..200: 0,01 x value – 1 255: no value available	UINT8	
Velocity correlation coefficient	7	7..0	-1..+1	0..200: 0,01 x value – 1 255: no value available	UINT8	

5.1.5 Message type 4: Object classification and age

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	4: Object outline info	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Height	1	7..0	0..12,7m	0..253 : 0,05 x value 254: >= 12,7 m 255: no value available	UINT8	Height of the object
Height sigma	2	7..0	0..12,7m	0..253 : 0,05 x value 254: >= 12,7 m 255: no value available	UINT8	standard deviation of the height
Classification certainty	3	7..0	0..100%	0..100: value 255: no value available	UINT8	Level of certainty that the classification is correct. Ref. to chapter 7 for details. The classification is available in the object header message.
Age of Classification	4	7..0	1..15	0: no value available 1..15 count of tracking scans	UINT8	
Age of Object	5 6	7..0 7..0	0..65535		UINT16	
	7					(unused)

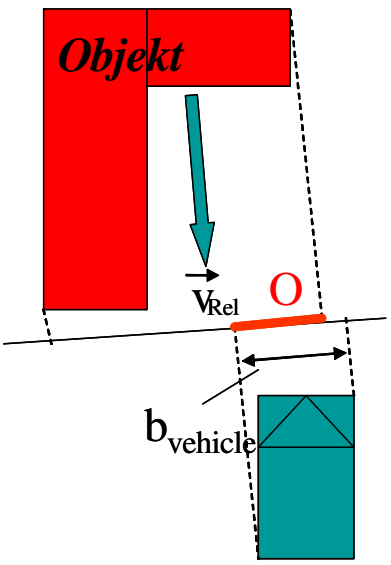
5.1.6 Message type 5: Extended Object info

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	5: Extended Object info	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Info type	1	7..0	0..255	1 x value	UINT8	Defines the interpretation of byte 2..7.
	2..7					depends on Info type

5.1.6.1 Info type 0: Collision info

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	5: Extended Object info	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Info type	1	7..0	0..255	0: Collision info	UINT8	collision info message
ttc	2 3	15..8 7..0	0..65534 ms	0..65533 : 1 x value 65534: >= 65534 ms 65535: no val. available	UINT16	Time to collision
Crash probability	4	7..0	0..100%	0..100: value 255: no value available	UINT8	Probability of a collision with this object
	5					(unused)
	6					(unused)
	7					(unused)

The time to collision (`ttc`) is given for each object corresponding to the time of beginning to send the data stream of this scan.
The `CrashProbability` is calculated from the overlapping area of the vehicle and object contour corresponding to the relative object velocity vector, the `ttc` and an application specific offset Δt (default=100 ms).



$$CP = \frac{O}{TTC - \Delta t}$$

Fig. 2: Overlapping area O and CrashProbability CP.

5.1.6.2 Info Type 1: Absolute velocity

The following message will be transmitted in addition to the Ibeo CANspecAS long format:

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	5: Extended Object info	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Info type	1	7..0	0..255	1: Absolute velocity	UINT8	This ia an absolute velocity message which will only be sent if data are available
Velocity X	2	7..0	-63.0 to 63.0 m/s	0: no value available 1: < -63.0 m/s 2..254: 0.5 x value – 64 255: value > 63.0 m/s	UINT8	Absolute velocity in X direction
Velocity Y	3	7..0	-63.0 to 63.0 m/s	0: no value available 1: < -63.0 m/s 2..254: 0.5 x value – 64 255: value > 63.0 m/s	UINT8	Absolute velocity in Y direction
Velocity X extension	4	7..4	-0.25 .. 0.21 m/s	Value / 32	INT4	For better resolution: Add this value to the abs. velocity X
Velocity Y extension	4	4..0	-0.25 .. 0.21 m/s	Value / 32	INT4	For better resolution: Add this value to the abs. velocity Y
Velocity X (Object number +1)	5	7..0	-63.0 to 63.0 m/s	0: no value available 1: < -63.0 m/s 2..254: 0.5 x value – 64 255: value > 63.0 m/s	UINT8	Absolute velocity in X direction of the 2nd object in this message (take Object number from Byte 0 and increase it by 1)
Velocity Y (object number +1)	6	7..0	-63.0 to 63.0 m/s	0: no value available 1: < -63.0 m/s 2..254: 0.5 x value – 64 255: value > 63.0 m/s	UINT8	Absolute velocity in Y direction of the 2nd object in this message.
Velocity X extension (Object number +1)	7	7..4	-0.25 .. 0.21 m/s	Value / 32	INT4	For better resolution: Add this value to the abs. velocity X of the 2nd object in this message.
Velocity Y extension (Object number +1)	7	4..0	-0.25 .. 0.21 m/s	Value / 32	INT4	For better resolution: Add this value to the abs. velocity Y of the 2nd object in this message.

The message contains the absolute velocity of 2 objects.

5.1.7 Message type 6: Object outline description

This message contains the contour points for objects along their outline. The direction is always from left to right (starts from the leftmost point). The last point for an object being the last point that the Laserscanner has detected.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	6: object points	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	
Number of point	1	7..4	0..15	1 x value	UINT4	
Point Number PositionX	1 2 3	3..0 7..0 7	-200.. 200 m	0..8000 : 0,05 x value – 200 8191: not available	UINT13	Relative to sensor or car coordinate system
Point Number PositionY	3 4	6..0 7..2	-200.. 200 m	0..8000 : 0,05 x value – 200 8191: not available	UINT13	Relative to sensor or car coordinate system
Point Number + 1 PositionX	4 5 6	1..0 7..0 7..5	-200.. 200 m	0..8000 : 0,05 x value – 200 8191: no value available	UINT13	Relative to sensor or car coordinate system
Point Number + 1 PositionY	6 7	4..0 7..0	-200.. 200 m	0..8000 : 0,05 x value – 200 8191: no value available	UINT13	Relative to sensor or car coordinate system

Before the output, redundant information is removed from the CAN outline so that two identical points will appear as one point in the CAN output. This happens e.g. if the leftmost point of an object is at the same time the closest point to the coordinate system origin. However, due to the rounding of position data, two points from the CAN data may be **de**coded to the same coordinates.

5.1.8 Message type 7: List end

This is the last message of an object list. A check should be performed to verify the integrity of the received object data:
Build the checksum by adding all bytes of all object messages, including the *list header* and *list end*, but excluding the checksum itself. Compare this value with the checksum sent in the *list end* message.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	7: list end	UINT3	Mode signal
Object number	0	7..3	0..31	0	UINT5	Must be 0 in the List end message
Error/Status code	1	7..0	0.. 255	1x value, each bit has its own meaning	UINT8	Error/status code 0: Normal operation Bit 1: Tracking error Bit 2: No valid scan Bit 3: No Scanner found Bit 4: Sensor dirty 1: Old: not available
Scan processing Status	2	7..0	0..255	1x value, each bit has its own meaning	UINT8	Bit 0: Angle offset added Bit 1: X-Y offset added Bit 2: Dirt points removed Bit 3: Rain points rem. Bit 4: Ground points rem. Bit 5: Covered points set Bit 6: Movement comp. Bit 7: ScanFusion
Cycle counter	3	7..0	0..255	1x value	UINT8	Cycle will increase by 1 with each scan. Overflow from 255 to 0
Checksum	4..7	31..0	0 .. 4.294.967.295	1 x value	UINT32	

5.2 Object data (SHORT format)

Message: Sensor object data
Identifier: Base ID + 1 (default: 4F1h)
Length: 8 byte Transmission rate: -
Receiver: single sensor Retransmission time: sensor depending

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	0: reserved 1: List – header 2..6: object message 7 : list – end	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	
Data	7..1	63..8				Depending on Mode and Object number

The short format was introduced to allow a more compact representation of the object data, reducing the CAN load while still delivering as much object information as possible. In the minimum setting (`ObjectPoints`=3), the short format requires 3 messages per object, plus 3 messages for list header and list end. The short format can be activated with the parameter setting mechanism (parameter "CAN format").

5.2.1 Message type 1: List header (Short format)

Note: *This message is only a list header if the object number is 0.*

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	1: list header	UINT3	Mode signal.
Object number	0	7..3	0..31	0: normal list header	UINT5	Must be zero for a list header.
Object style	1	7..0	0..255	0-255: (reserved) 32: Special Version 1 64: IBEO AS short data format	UINT8	Special Version 1: Changes of Version 1 are marked in the following description.
Number of objects	2	7..2	0..31	1 x value	UINT6	
Reserved	2	1	0..1	-	UINT1	Always 1, ignore.
Calibration flag	2	0	0..1	0: not calibrated 1: calibrated	UINT1	
Cycle counter	3	7..0	0..255	1 x value	UINT8	Cycle will increase by 1 with each scan. Overflow from 255 to 0.
Timestamp	4 5 6 7	7..0 7..0 7..0 7..0	$0..2^{32}$ -1 ms	1 x value	UINT32	Moment of beginning of the measurement.

5.2.2 Message type 1: Lane detection and collision info (Short format)

Note: *This message is only a Lane detection info #1 if the object number is 2.*

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	1: list header	UINT3	Mode signal
Object number	0	7..3	0..31	2: Lane detection and collision info #1	UINT5	
Left Road width	1	7..0	0..255	0-253: width = value x 0.1m 254 : width >= 25.4 m 255: Unknown	UINT8	
Right Road width	2	7..0	0..255	0-253: width = value x 0.1m 254 : width >= 25.4 m 255: Unknown	UINT8	
Curvature	3 4	7..0 7..4	0..4095	0-4094: curvature = 2 x (value x 1E-5 - 0.02047) 4095: Unknown	UINT12	
View range	4	3..0	0..15	0: Range <= 10 m 1-13: Range=(value * 5) + 10 m 14: Range >= 80 m 15: Unknown	UINT4	
Number of lanes	5	7..5	0..7	0-6: Lanes = value + 1 7: Unknown	UINT3	
Collision object	5	4..0	0..31	0-31: Number of the collision object	UINT5	Only valid if Time-To-Collision is valid!
Time-To-Collision	6	7..0	0..255	0..253: Time = value*10 ms 254: No crash object 255: Unknown, invalid	UINT8	Collision object number is only valid if this value is 0..253 or 255.
CrashProbability	7	7..4	0..15	0..10: CrashProb = value * 10 % 15: Unknown	UINT4	
(reserved)	7	3..0				Reserved for future use.

Note: *For details, please refer to chapter 8.*

5.2.3 Message type 2: General object info 1 (Short Format)

Each object is proceeded by an *General object info 1* message.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	2: General object info1	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Object ID	1	7..0	0..255	1 x value	UINT8	For the life of an object constant number (see below for explanation)
Tracking status	2	0	0,1	0: known object 1: unknown object	UINT1	1 if this object is sent for the first time
Age	2	3..1	0..7	0: New object, 1..5: Age <= value s 6: Age >= 6 s 7 = unknown	UINT3	Time that the object was seen by the scanner. Note that this is not necessarily the time that the object was sent via CAN.
ACC type	2	7..4	0..15	0 = standing still, 1 = coming towards me, 2 = moving away, 15 = unknown	UINT4	Characterizes the object type for ACC applications.
Classification	3	7..0	0..255	1 x value	UINT8	Ref. to chapter 7
Number of points	4	7..0	1..256	1 x value + 1	UINT8	Number of contour points of an object
VelocityX	5	7..0	-63,5.. 62,5 m/s	0: value < -63,5 m/s 1..253: 0,5 x value – 64 254: value >= 62,5 m/s 255: no value available	UINT8	Relative velocity in X direction
VelocityY	6	7..0	-63,5.. 62,5 m/s	0: value < -63,5 m/s 1..253: 0,5 x value – 64 254: value > 62,5 m/s 255: no value available	UINT8	Relative velocity in Y direction
Velocity X ext.	7	7..4	-0.25.. 0,21 m/s	Value / 32	INT4	For better resolution: Add this value to the Velocity X
Velocity Y ext.	7	3..0	-0.25.. 0,21 m/s	Value / 32	INT4	For better resolution: Add this value to the Velocity Y

5.2.4 Message type 3: General Object info 2 (Short format)

Each object is proceeded by a *General Object info 2* message.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	3: General object info2	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	Number 0 as 1st
Rel. Accel. X	1,2	7..0, 7,6	0..1023	0-1000: 0,01 x value – 5 1023: not available	UINT10	Relative to sensor or car coordinate system
Rel. Accel. Y	2,3	5..0, 7..4	0..1023	0-1000: 0,01 x value – 5 1023: not available	UINT10	Relative to sensor or car coordinate system
Lane	3	2..0	0..7	0= left of my lane, 1= on my lane, 2= right of my lane, 7= unknown	UINT3	Position of the object with respect to our lane. Only available in certain configurations.
First Point PositionX	4 5	7..0 7..3	-200.. 200 m	***Ibeo-Short-Format:*** 0..8000 : 0,05 x value – 200 ***Special Version 1:*** 0..8000: 0,0125 x value 8191: not available	UINT13	Relative to sensor or car coordinate system. **Note:** Special Version 1 restricts the range to 0..100 m, while the Ibeo-Format transmits -200..200 m
First Point PositionY	5 6 7	2..0 7..0 7,6	-200.. 200 m	***Ibeo-Short-Format:*** 0..8000 : 0,05 x value – 200 ***Special Version 1:*** 0..8000: 0,0125 x Value - 50 8191: not available	UINT13	Relative to sensor or car coordinate system. **Note:** Special Version 1 restricts the range to +/- 50 m, while the Ibeo-Format transmits -+/- 200 m

5.2.5 Message type 6: Object outline description (Short format)

This message contains contour points for objects which have more than 1 contour point along their outline. The direction is always from left to right. The leftmost point is coded in the header, and these points are added in left-to-right order, the last point for an object being the last point that the Laserscanner has detected.
Note that this message is identical to the `Object point's` message of the standard format.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	6: object points	UINT3	Mode signal
Object number	0	7..3	0..31	1 x value	UINT5	
Number of point	1	7..4	0..15	1 x value	UINT4	
Point Number PositionX	1 2 3	3..0 7..0 7	-200.. 200 m	***Ibeo-Short-Format:*** 0..8000 : 0,05 x value – 200 ***Special Version 1:*** 0..8000: 0,0125 x value 8191: not available	UINT13	Relative to sensor or car coordinate system. **Note:** Special Version 1 restricts the range to 0..100 m, while the Ibeo-Format transmits -200..200 m
Point Number PositionY	3 4	6..0 7..2	-200.. 200 m	***Ibeo-Short-Format:*** 0..8000 : 0,05 x value – 200 ***Special Version 1:*** 0..8000: 0,0125 x Value - 50 8191: not available	UINT13	Relative to sensor or car coordinate system. **Note:** Special Version 1 restricts the range to +/-50 m, while the Ibeo-Format transmits -+/- 200 m
Point Number + 1 PositionX	4 5 6	1..0 7..0 7..5	-200.. 200 m	***Ibeo-Short-Format:*** 0..8000 : 0,05 x value – 200 ***Special Version 1:*** 0..8000: 0,0125 x value 8191: not available	UINT13	Relative to sensor or car coordinate system. **Note:** Special Version 1 restricts the range to 0..100 m, while the Ibeo-Format transmits -200..200 m
Point Number + 1 PositionY	6 7	4..0 7..0	-200.. 200 m	***Ibeo-Short-Format:*** 0..8000 : 0,05 x value – 200 ***Special Version 1:*** 0..8000: 0,0125 x Value - 50 8191: not available	UINT13	Relative to sensor or car coordinate system. **Note:** Special Version 1 restricts the range to +/-50 m, while the Ibeo-Format transmits -+/- 200 m

Before the output, redundant information is removed from the CAN outline so that two identical points will appear as one point in the CAN output. This happens e.g. if the leftmost point of an object is at the same time the closest point to the coordinate system origin. However, due to the rounding of position data, two points from the CAN data may be decoded to the same coordinates.

5.2.6 Message type 7: List end (Short format)

This is the last message of an object list. A check should be performed to ensure the integrity of the received object data:
Build the checksum by adding all bytes of all object messages, including the *list header* and *list end*, but excluding the checksum itself. Compare this value with the checksum sent in the *list end* message.
Note that this message is identical to the `List end` of the standard format.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	7: list end	UINT3	Mode signal
Object number	0	7..3	0..31	0	UINT5	Must be 0 in the List end message
Error/Status code	1	7..0	0.. 255	1x value, each bit has its own meaning	UINT8	Error/status code 0: Normal operation Bit 1: Tracking error Bit 2: No valid scan Bit 3: No Scanner found Bit 4: Sensor dirty 1: Old: not available
Scan processing Status	2	7..0	0..255	1x value, each bit has its own meaning	UINT8	Bit 0: Angle offset added Bit 1: X-Y offset added Bit 2: Dirt points removed Bit 3: Rain points rem. Bit 4: Ground points rem. Bit 5: Covered points set Bit 6: Movement comp. Bit 7: ScanFusion
Cycle counter	3	7..0	0..255	1x value	UINT8	Cycle will increase by 1 with each scan. Overflow from 255 to 0
Checksum	4 5 6 7	7..0 7..0 7..0 7..0	$0..2^{32}-1$	1 x value	UINT32	(see description above)

5.3 Parameter reading

Parameter values can be requested by sending a `Get Parameter` command to the sensor. The sensor will then answer with this `Parameter` message. This message is coded as a `list header` with the *Message type* = 1, but with an *object number* of 31 to indicate that this is a `Parameter` message.

Signal name	Byte	Bit	Domain	Conversion	Data type	Comment
Message type	0	2..0	0..7	1: List header	UINT3	Mode signal
Object number	0	7..3	0..31	31: Parameter	UINT5	Must be 31
Parameter	1	7..0	0..255		UINT8	
Data type	2	7..0	0..255	0: INT16 1: FLOAT16 2: UINT8 3: UINT16 4: UINT32	UINT8	-32768..32767 -327,68..327,67 0..255 0..65536 0..4294967296
Data value	3 4 5 6	7..0 7..0 7..0 7..0	(see chapter 6)	(see chapter 6)	(see chapter 6)	(see chapter 6)
	7	7..0				(unused)

6 Parameters

Whenever parameters are to be sent to the Laserscanner or read from the scanner, the following set of parameters is available. The parameter is identified by a 8-bit value, the `parameter index`. It also has a fixed data type which is used for transmission.
All parameters can be read, however, those marked with a „W“ in the R/W column can also be written with the `Set Parameter` command.
Some of the parameter values are scanner-specific (such as the start and end angles). If more than one scanner is attached to the ECU, the scanner for which the parameter should apply is set with byte 7 (the last byte) of the `SetParameter` message (0 for first scanner (=lowest ARCnet ID) and so on). If a message with less than 8 bytes is used, the Sensor number is assumed to be 0, so the settings apply to the first sensor (this is always ok for single-sensor systems).

Parameter	Index	Data type	Default value	R/W	Description
Start angle	0	FLOAT16	80.0	R/W	Start angle of the scan/object detection area. Scanner-specific.
End angle	1	FLOAT16	-80.0	R/W	End angle of the scan/object detection area. This angle is assumed to lie counter clockwise behind the `Start angle`. Scanner-specific.
XYFactor	2	FLOAT16	3.0	R/W	Factor by which the value `MinDist` is multiplied in x direction.
MinDist	3	FLOAT16	0.5	R/W	Segmentation factor (y direction). Scan points which are closer together than this value are fused together into one segment.
ObjectPoints	4	UINT16	0	R/W	Max number of contour points for each object. Minimum is 3 and maximum 16. If set to 0, automatically the optimal number of object contour points is selected.
CAN Baudrate	5	UINT16	20	R/W	20 = 1 Mbit/s, 28 = 500 kBit/s
CAN Base ID	6	UINT16	0x4F0	R/W	Base ID for CAN communication.
OffsetX	7	FLOAT16	0.0	R	(set via calibration - Scanner-specific.)
OffsetY	8	FLOAT16	0.0	R	(set via calibration - Scanner-specific.)
Sensor ID	9	UINT16	0x10	R	ID of the processing system (either IPC or sensor); used for ARCnet transmission.
Quality criterion	10	UINT16	0	R/W	0: Radial, 1: Look Ahead, 5: ACC-filter
HorizontalAngleOffset	11	FLOAT16	0.0	R	(set via calibration)
Calibration-Counter	12	UINT16	0	R	The calibration counter will be increased after every successfully calibration event. Using this value, the external application can monitor the calibration success.
SendObjects	13	UINT16	20	R/W	The maximum number of objects that the Laserscanner may send in one scan. The CAN bandwidth can be limited by decreasing this value.
OffsetZ	14	FLOAT16	0.0	R	(set via calibration - Scanner-specific.)
VerticalAngleOffset	15	FLOAT16	0.0	R	(set via calibration - Scanner-specific.)
LastCalibrationStatus	16	UNIT16	0	R	Shows the status of the current calibration
OutputAreaX1	17	FLOAT16	0.0	R/W	Only objects are inside of this area will be send. If all parameters are default, all objects will be sent.
OutputAreaY1	18	FLOAT16	0.0	R/W	
OutputAreaX2	19	FLOAT16	0.0	R/W	
OutputAreaY2	20	FLOAT16	0.0	R/W	

Parameter	Index	Data type	Default value	R/W	Description
Rotation frequency	21	UINT16	-	R/W	Change or read the rotation frequency of the specified Laserscanner. Scanner-specific. Unit depends on value: value ≤ 100: unit is Hz, e.g. 12 = 12 Hz value > 100: unit is mHz, 12500 = 12.5 Hz
	22			R/W	reserved
TimeRequest	23	UINT32	-	R	returns the current system time at the time of sending the reply message
CAN Format	24	UINT8	0	R/W	Sets the CAN output format. Currently allowed is: 0 = Standard format ("Long") 1 = Short format
Sensor ID list	25	4 x UINT8	-	R	ID List of all (max 4) connected sensors. Values greater zero represent a valid sensor. This ID(s) can be used for the sensor specific parameter setting.

7 Object classes

If the classification module is active, the parameter “class“ in the object header is one of the following:

Parameter	Class	Description
0	Unknown small	Object isn't (not yet) classified but it's small.
1	Pedestrian	Object meets the criteria of the class "Person", like size and velocity.
2	Car	Object meets the criteria of the class "Car", like size and velocity.
3	Truck/Bus	Object meets the criteria of the class "Truck/Bus", like size and velocity.
4*	Bike	Object meets the criteria of the class "Bike", like size and velocity.
5	Unknown big	Object isn't (not yet) classified but it's big.
6*	Possible pedestrian	The class of the object isn't really known, but it's possible that this is a person.
...	--	unused
15	Unclassified	Unclassified object

*) not yet available.
If the classification module is not active, all objects are set to „unclassified“.

8 Lane detection

The data from the Lane Detection system is sent in the CAN-message “Lane detection info". The Lane Detection uses the raw and object data of the Laserscanner to predict the position and curvature of the road in front of the Laserscanner. Its output parameters are:

Parameter	Description
Left Lane Offset	Offset in [m] between the Reference point and the left lane boundary, at the x-position of the reference point. In the figure below, this is the parameter C_{0_l} (not shown).
Right Lane Offset	Same as Left Lane Offset, only for the right side. In the figure below, this is the parameter C_{0_l}.
Heading Angle	The basic heading of the Lane in [rad]. In the figure below, this is the parameter C_1.
Curvature	The curvature of the road, in [1/m]. In the figure below, the parameter C_2 is twice the curvature. Note that this Parameter is sent via CAN.
Confidence Level	The level of confidence of the current Lane Detection parameters. This level depends on the number and type of objects which are available, and the ground which is monitored by the Laserscanner.
View range	The approximate range of the lane detection. This is the range in which the street parameters are considered to be reasonably reliable.

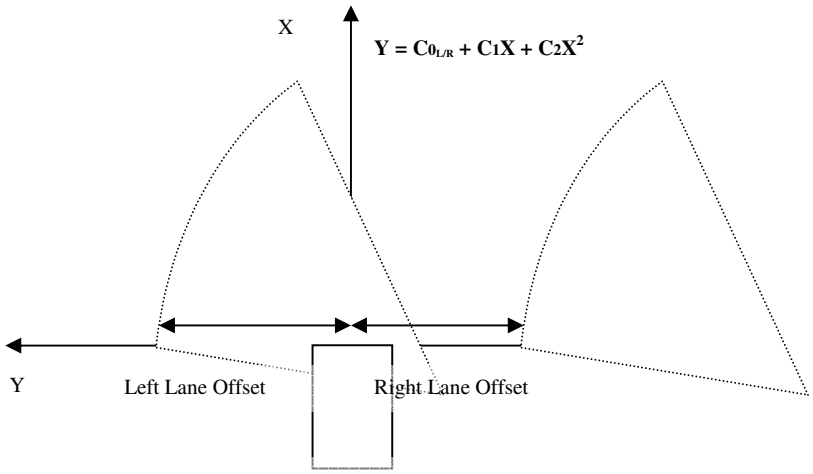


Fig. 3: Lane Detection (schematic overview over output parameters)

9 Examples

This chapter contains examples how to send data to a Laserscanner and receive data from a Scanner.

9.1 Start-up

Assuming that the sensor is in its default configuration, it will immediately start sending object information after starting the application, which will take some seconds.
This object information consists of:

1. A list header message
2. For each object:
   - An object header message (General object info, see chapter 5.1.3)
   - Maybe an object header (ext. 1) message (Object deviation info, see chapter 5.1.4)
   - Maybe an object header (ext. 2) message (Object outline info, see chapter 0)
   - Maybe a list of contour points, depending on the object shape, see chapter 5.1.7
3. A list end message

These messages are sent on the CAN ID "Base ID + 1", which is 0x4F1 in default configuration.

Here is an **example** for a list header – CAN message:

CAN message ID = 4F1h (CAN Base ID + 1 in default configuration)
DLC (Length) = 8 (8 bytes of data)
Object number = 0 (normal list header)
Object style = 128 (IBEO AS data)
Number of objects = 0 (if there are objects the sensor wants to send, this value is NOT 0)
Sensor status = 1 (OK)
Calibration flag = 0 (not calibrated)
Cycle counter = x (Set to 0 at start-up)
Timestamp = x (Set to 0 at start-up)

→ CAN message: Byte 0 = 01h
Byte 1 = 00h
Byte 2 = 40h
Byte 3 = xxh (Current value of the Cycle counter)
Byte 4 = xxh (MSByte of Timestamp)
Byte 5 = xxh
Byte 6 = xxh
Byte 7 = xxh (LSByte of the Timestamp)

9.1.1 Changing the CAN Base ID

If more than one Laserscanner is to be used on one CAN bus, the CAN Base ID (with which the communication to/from the sensor is carried out) of the sensors must be changed from the identical factory setting to unique values.
Do not connect any different Laserscanners/ ECUs with the same Base ID! Then, use the `SetParameter` command to set the new Base ID. After setting the new ID, the sensor will store the value and immediately switch to the new CAN ID for communication.

If we assume that the current Base ID is 0x4F0, and that we want to set the ID 0x123 as the new Base ID, the command is:

MsgID = 0x4F0, DLC=5, Data =	0x03	(Command: Set Parameter)
	0x06	(Parameter "CAN Base ID")
	0x02	(Data type UINT16)
	0x01	(MSByte of new Base ID)
	0x23	(LSByte of new Base ID)

10 Message overview

Some explanations:
MT: Message type
Byte 1, Bit 1: Sensor status
Byte 1, Bit 0: Calibration flag

0		1	2	3	4	5	6	7
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
List Header								
Object Nr.	MT	Object style	Number of obj.	Cycle counter	MSB - Timestamp - LSB			
Environment info								
Object Nr.	MT	Sensor dirty	Rain detection	Dirt det. start angle	Dirt det. end angle	*unused*		
Lane detection info #1								
Object Nr.	MT	MSB - Left lane offset - LSB		MSB - Right lane offset - LSB		MSB - Curvature - LSB		Heading angle
Lane detection info #2								
Object Nr.	MT	Confidence level	View range	*unused*				
General object info								
Object Nr.	MT	Tracking number	Tracking status	Classification	Number of points	VelocityX	VelocityY	*unused*
Object deviation info								
Object Nr.	MT	el. Moment of measu	PositionX sigma	PositionY sigma	VelocityX sigma	VelocityY sigma	Pos. Correl. Sigma	Vel. Correl. Sigma
Object outline info								
Object Nr.	MT	Height	Height sigma	Class. Certainty	Age of class.	Age of object		*unsused*
Object points								
Object Nr.	MT	Point Nr.	MSB - Point Nr. pos. X - LSB	MSB - Point Nr. pos. Y - LSB	MSB - Point Nr.+1 pos. X - LSB	MSB - Point Nr.+1 pos. Y - LSB		
List end								
Object Nr.	MT	MSB - Error / Status code - LSB		Cycle counter	MSB - Checksum - LSB			

11 History

v1.0.6, 10.10.02
- Changed description of parameter "View Range" in the Street Detection section
- History added

v1.0.7, 06.02.03
- Message type 5 added

v1.0.8, 10.03.03
- Collision info as object attribute added.

v1.0.9, 22.05.03
- Crash prediction added.
- Higher resolution of the relative velocity.
- Parameter “sector points” means the maximum object points of each object.

v1.1.0, 03.06.03
- New configurable Vehicle data Parser.

v1.1.1, 18.06.03
- `TTCMinTime` removed.

v1.2.0, 18.10.03
- Parameter `SectorPoints` changed to `ObjectPoints`, and description updated.
- CAN point output message description updated (info about double points with identical coordinates).
- Ini-file settings added to parameter section
- Multi-Scanner parameter setting description added, and list updated ("Sensor specific" entries)

v1.2.2, 19.11.03
- QualityCriterion "ACC" added to parameter list.

v1.2.3, 14.12.03
- New IBEO AS Logo added

v1.3.0, 20.01.04
- Broadcast messages to receive vehicle data removed
- Object classes modified

v1.4.0, 26.01.04
- Short message format added
- Parameter "CAN Format" added

v1.5.2, 15.06.04
- List Header: Sensor Status removed.

v1.5.3, 30.07.04
- CrashPrediction info changed.
- Identifier for Applications defined.

v1.5.5, 03.03.05
- Ini-File description removed
- Converted to word
- Calibration removed

v1.5.7, 01.08.05
- Description of parameter “rotation frequency” extended (unit Hz or mHz)
- Replaced “SystemSetup” by the correct name “ASystemSetup”

v1.5.8, 08.08.2005
- List Header and List End description: Replaced “bit 32..0” by “bit 31..0”

v1.6.0, 04.04.2006
- Added Mode "Special Version 1" in the Short-Format

v1.6.1, 16.05.2006
- Bugfix in the description of " Special Version 1" (see v1.6.0). Now, Message Type 6 also uses the new format.

v1.6.2, 26.09.2006
- Bugfix in the description chapter 5.1.10

v1.7.0, 07.08.2007
- Added absolute velocity as extended object information (info type 1)
- Delete old Parameter