ibeo LUX and ibeo LUX Systems

CAN protocol description

Version History

Date	Version	Changes
19.11.2008	1.0	Initial release
16.06.2009	1.1	Parameter name vehicle width fixed. Fixed ego motion data format removed.
02.10.2009	1.2	Object list trailer message (e.g. 0x508) added. Counter in list header message (e.g. 0x500) added. Protocol version changed to 2.
05.10.2010	1.3	Time sync message (e.g. 0x50C) added.
26.10.2010	1.4	Time sync message ID change to 0x100. Add parameter Customer Processing Switch 0 and Interface Flags.
09.05.2012	1.5	Correction: Box size y (chapter: 3.6)
05.12.2013	1.6	Updated description for contour/closest point.
26.02.2014	1.7	Object flags added, updated to CAN version 2.1

Table of Content

1 Introduction

This document describes how data is received and transmitted via CAN. Addressed systems are ibeo LUX sensors and ECUs or applications using the current Ibeo API/software versions.

2 General information

CAN messages always have an identifier (ID). Ibeo systems use a parameter called CAN base ID. This parameter defines a range of 16 IDs.
With a base ID 0x500 the ID range is [0x500, 0x50F], first and last ID included.
A message ID defines the message and its contents.
Thus each ID may only be used by one device or in one matter.
For the device with base ID 0x500 the message 0x50A (base ID + 0xA) is used to send commands to this device.
The device answers with message ID 0x50B (base ID + 0xB).

Message length is always 8 bytes as long as not declared differently.
Message byte and bit numbering is zero based.
Data encoding is big endian (Motorola) for all messages instead of command, command reply and errors and warnings, which use little endian (Intel).

This document describes protocol version 2.1. This version is downwards compatible to versions 2.0 and 1. Information was only added: there are extra values in some messages, and the object list trailer was added in version 2.0.

3 Object Data

Object data can be transmitted via CAN.
The ID range is CAN base ID … CAN base ID + 0x8, first and last ID included.
For a base ID 0x500 the range is [0x500, 0x508].

3.1 List header: CAN Base ID (e.g. 0x500)

Content	Data area	Data type	Description
Version	byte 0	UINT8	Version of the object data format. This document describes version 2.
Number of objects	byte 1	UINT8	Number of objects transmitted in this cycle.
Sensor view range	byte 2	UINT8	Estimated maximum sensor view range on typical vehicles.
Sensor temperature	byte 3	INT8	Current sensor temperature in °C. 0x80 indicates an invalid value.
Object data info flags	byte 4	bit field 8 bits	bit 0: 0 = absolute velocities, 1 = relative velocities. bit 1: 0 = boxes are object boxes, 1 = boxes are bounding boxes. bits 2…7: reserved
Counter	byte 5	UINT8	This value can be used to check if list header and trailer match.
Reserved	bytes 6…7	-	-

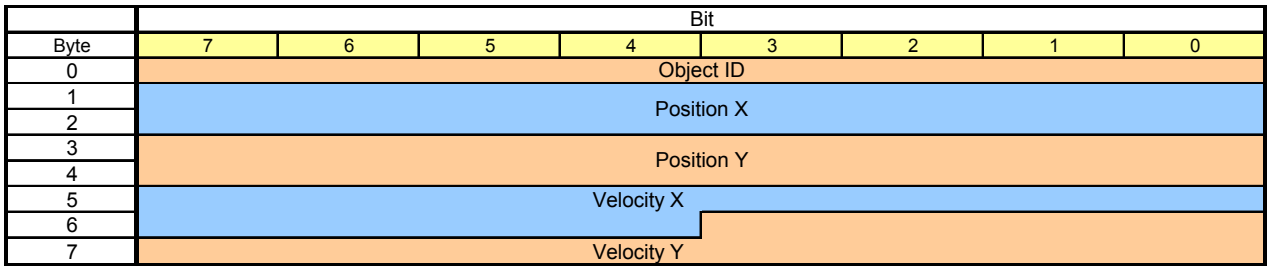
3.2 Time stamp: CAN Base ID + 0x1 (e.g. 0x501)

Content	Data area	Data type	Description
NTP seconds	bytes 0…3	UINT32	Start time of the scan these objects are based on.
NTP fractional seconds	byte 4…7	UINT32	Fractional seconds of the scan start time.

3.3 Tracking1: CAN Base ID + 0x2 (e.g. 0x502)

Content	Data area	Data type	Description
Object ID	byte 0	UINT8	ID of this object from the tracking. Use this ID to refer messages to an object.
Position x	bytes 1…2	INT16	Position of the object (reference point, e.g. center of gravity) in the reference coordinate system in cm.
Position y	bytes 3…4	INT16	
Velocity x	byte 5 bits 0…7, byte 6 bits 4…7	INT12	Object velocity in 0.1 m/s in the reference coordinate system. See list header for absolute or relative velocities. 0x800 indicates an invalid velocity.
Velocity y	byte 6 bits 0…3, byte 7 bits 0…7	INT12	

Please refer to this image for clarification of the bits and bytes used for each information:



3.4 Tracking2: CAN Base ID + 0x3 (e.g. 0x503)

Content	Data area	Data type	Description
Object ID	byte 0	UINT8	ID of this object from the tracking. Use this ID to refer messages to an object.
Object age	byte 1	UINT8	Number of scans this object has been tracked for. Saturates at 0xFF.
Object prediction age	byte 2	UINT8	Number of scans this object has only be predicted without measurement update. Saturates at 0xFF. Is reset to 0 after measurement update.
Object time offset	byte 3	UINT8	Detection time of this object as offset to the reference time stamp in ms.
Position x sigma	byte 4	UINT8	Standard deviation of the position estimation in cm.
Position y sigma	byte 5	UINT8	
Velocity x sigma	byte 6	UINT8	Standard deviation of the velocity estimation in cm.
Velocity y sigma	byte 7	UINT8	

3.5 Class and box1: CAN Base ID + 0x4 (e.g. 0x504)

Content	Data area	Data type	Description
Object ID	byte 0	UINT8	ID of this object from the tracking. Use this ID to refer messages to an object.
Object classification	byte 1	UINT8	Most likely class of this object: 0: unclassified 1: unknown small 2: unknown big 3: pedestrian 4: bike 5: car 6: truck 7…: reserved
Object classification certainty	byte 2	UINT8	The higher this value is the more reliable is the assigned object class.
Object classification age	byte 3	UINT8	Number of scans this object has been classified with its current class. Saturates at 0xFF.
Box center x	bytes 4…5	INT16	Center position of the box in cm. See list header for object box / bounding box.
Box center y	bytes 6…7	INT16	

3.6 Box2: CAN Base ID + 0x5 (e.g. 0x505)

Content	Data area	Data type	Description
Object ID	byte 0	UINT8	ID of this object from the tracking. Use this ID to refer messages to an object.
Box size x	bytes 1…2	UINT16	Size of the box in cm in the object coordinate system. The box orientation is only available for an object box. A bounding box is an unrotated rectangle in the reference coordinate system. In this case the box size is always given in the reference coordinate system. 0x8000 indicates an invalid orientation.
Box size y	bytes 3…4	UINT16	
Box orientation	byte 5…6	INT16	Object box orientation in the reference coordinate system in 1/100°. 0x8000 indicates an invalid orientation.
Reserved	byte 7	-	-

3.7 Contour header: CAN Base ID + 0x6 (e.g. 0x506)

Content	Data area	Data type	Description
Object ID	byte 0	UINT8	ID of this object from the tracking. Use this ID to refer messages to an object.
Number of contour points	byte 1	UINT8	Number of contour points including start point transmitted for this object. The number of following ObjectDataContour messages can be calculated by (NumOfContourPoints+1) Div 3. If this value is set to 0xFF (invalid), the contour of this object was not calculated correctly (e.g. too many contour points). In this case, the ContourStartpoint contains the closest distance to the object. No more contour point messages are sent for this object. In case the CAN output is configured such that not the contour is sent out, but only the closest point is setnt out, this value is 0xFF.
Closest contour point number	byte 2	UINT8	The closest object distance can be found in the contour point with this number. The numbering starts with 0 (start point).
Object motion flags (only available in CAN spec 2.1 or higher)	byte 3		bit 0: 1 = object is currently tracked using a stationary model 0 = object is currently tracked using a dynamic model bit 1: 1 = object has been detected as dynamic at some time (maybe in the past) 0 = object has never been dynamic bit 2: 1 = object motion model (stationary or dynamic) has been validated 0 = object motion model has not been validated, i.e. it is a new object and it is not yet certain whether it is moving or not
Start point x	bytes 4…5	INT16	Position of the first contour point in cm in the reference coordinate system. This is the first point of the contour (or the closest distance, see above). The following contour points are only given by offsets to the previous points.
Start point y	bytes 6…7	INT16	

3.8 Contour points: CAN Base ID + 0x7 (e.g. 0x507)

Content	Data area	Data type	Description
Object ID	byte 0	UINT8	ID of this object from the tracking. Use this ID to refer messages to an object.
Contour message number	byte 1	UINT8	Number of this contour message. Zero based.
x offset (e.g. point 1)	byte 2	INT8	Add these offsets to the position of the previous point. Calculate the position for each contour point (besides the start point) using the offsets. First contour point is the start point sent in the contour header message. Note that these offsets have a resolution of 4 cm. Multiply these values by 4 to convert to cm.
y offset (e.g. point 1)	byte 3	INT8	
x offset (e.g. point 2)	byte 4	INT8	
y offset (e.g. point 2)	byte 5	INT8	
x offset (e.g. point 3)	byte 6	INT8	
y offset (e.g. point 3)	byte 7	INT8	

3.9 List trailer: CAN Base ID + 0x8 (e.g. 0x508)

Content	Data area	Data type	Description
Number of object list CAN messages sent	bytes 0…1	UINT16	This value can be used to check, if all messages describing the current object list were received. The value includes header and trailer (this) messages.
Number of warning/ error messages sent	byte 2	UINT8	This value indicates how many warning/ error messages were sent since the last object list trailer. 0 indicates that there were no reported problems. The value 255 (0xff) means, that there occurred 255 or more warning/error messages since last list trailer.
Counter	byte 3	UINT8	This value can be used to check if list header and trailer match.
Reserved	bytes 4…7	-	-

4 Command Interface

Commands can be transmitted via CAN.
The ID is CAN base ID + 0xA.
For a base ID 0x500 the command ID is 0x50A.
Attention: The data is encoded in little endian byte order in this message!

Content	Data area	Data type	Description
Command ID	bytes 0…1	UINT16	See detailed list of commands and according options/parameters.
Command data	bytes 1…7	-	Depending on command. May be unused for some commands.

After receiving a command a reply is always sent.
The ID is CAN base ID + 0xB.
For a base ID 0x500 the command ID is 0x50B.
Attention: The data is encoded in little endian byte order in this message!

Content	Data area	Data type	Description
Reply ID	bytes 0…1	UINT16	If a command succeeded, the reply ID is equal to the corresponding command ID. If a command failed, the reply ID is the command ID + 0x8000. Thus, the most significant bit indicates a failed command.
Command data	bytes 1…7	-	Depending on command this reply is related to. May be completely missing for some replies and if a command failed.

4.1 Ibeo LUX commands and command replies

4.1.1 Reset

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0000	UINT16	Reset DSP

In case of command Reset no reply is sent.

4.1.2 Get Status

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0001	UINT16	Status request

Bytes	Offset	LUX reply: content	Content type	Description
2	0	0x0001	UINT16	Status request
2	2	Firmware version	UINT16	e. g. 0x1230 = version 1.2.3, 0x123B = version 1.2.3b
2	4	FPGA version	UINT16	e. g. 0x1230 = version 1.2.3, 0x123B = version 1.2.3b
2	6	Scanner status	UINT16	Bit field, with the following meaning for every bit: Bit 15 ...6: reserved / internal Bit 5: phase locked Bit 4: external sync signal available Bit 3: frequency locked Bit 2: reserved / internal Bit 1: laser on Bit 0: motor on
4	8		UINT32	reserved / internal
2	12	temperature	UINT16	T[°C] = - (temperature - 579.2364) / 3.63
2	14	serial number 0	UINT16	YYCW (z. B. YYCW = 0x0740 = year '07, calendar week 40)
2	16	serial number 1	UINT16	Counter of serial number
2	18		UINT16	reserved / internal
6	20	FPGA time stamp	[3] * UINT16	YYYY MMDD hhmm (FPGA version state)
6	26	DSP time stamp	[3] * UINT16	YYYY MMDD hhmm (Firmware version state)

4.1.3 SaveConfig

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0004	UINT16	Current sensor configuration will be saved permanently. Multiple SetParameter commands may be sent before saving the changes permanently.

The command SaveConfig will be acknowledged by the same command ID without command reply data.

4.1.4 Set Parameter

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0010	UINT16	Set a single Parameter by its index to the sensor memory. Parameter is set only temporarily until a SaveConfig command (see 4.1.3) is sent.
2	2	Parameter index	UINT16	Refer to ibeo LUX parameter list.
4	4	Parameter	UINT32	Set parameter accordingly to parameter list. If e.g. a 2 byte value is set, use the first 2 bytes. Fill the remaining 2 bytes with 0.

The command Set Parameter will be acknowledged by the same command ID without any command reply data.

4.1.5 Get Parameter

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0011	UINT16	Read a single Parameter with its index from the LUX.
2	2	Parameter index	UINT16	Refer to ibeo LUX parameter list (4.2)

Bytes	Offset	LUX reply: content	Content type	Description
2	0	0x0011	UINT16	Read a single Parameter by its index from the LUX.
2	2	Parameter index	UINT16	Refer to Ibeo LUX parameter list (4.2)
4	4	Parameter	UINT32	

4.1.6 Reset Default Parameters

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x001A	UINT16	Resets all parameters to the factory defaults.

The command Reset Default Parameters will be acknowledged by the same command ID without any command reply data.

4.1.7 Start Measure

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0020	UINT16	Starts the measurement with the current settings.

The command Start Measure will be acknowledged by the same command ID without any command reply data.

4.1.8 Stop Measure

Bytes	Offset	LUX command: content	Content type	Description
2	0	0x0021	UINT16	Stops the measurement.

The command Stop Measure will be acknowledged by the same command ID without any command reply data.

4.2 ibeo LUX parameter list

This table gives an overview of available ibeo LUX parameters. Please refer to 4.1.4 and 4.1.5 for details on getting and setting these parameters.

IP address, subnet mask and standard gateway encode the data as UINT32 value which is built like that: aa.bb.cc.dd = 0xaabbccdd. Due to little endian byte order this value must be sent as 0xddccbbaa.

Bytes	Parameter index	LUX parameter: content	Content type	Description
4	0x1000	IP address	UINT32	Valid: all
2	0x1001	TCP Port	UINT32	Valid: all
4	0x1002	Subnet Mask	UINT32	Valid: all
4	0x1003	Standard gateway	UINT32	Valid: all
2	0x1004	Customer Processing Switch 0	UINT16	bit true: process false: do not process. bit0: dirt + rain detection. bit3/bit2: timeSyncMode: - 00: NMEA over RS232 - 01: ETH - 10: CAN - 11: AUTO
4	0x1010	CAN Base ID	UINT32	Valid: value <= 0x7F0
2	0x1011	CAN Baud Rate	UINT16	in kBaud - next matching value (1000 kBaud, 500 kBaud, 250 kBaud, 125 kBaud) will be used.
2	0x1012	Data Output Flag	16 bit field	Bit true: disable output, false: enable output. 0xFFFF is invalid. bit0: ETH scan data bit1: reserved/internal bit2: ETH object data bit3: ETH vehicle data bit4: ETH errors/warnings bit5: CAN errors/warnings bit6: CAN object data bit7...15: reserved/internal
2	0x1013	maxObjectsViaCAN	UINT16	<= 65 (max. number of objects) limited by tracking and CAN bus capacity.
2	0x1014	ContourPointDensity	UINT16	Valid: < 3 0: closest point only 1: low density 2: high density

Bytes	Parameter index	LUX parameter: content	Content type	Description
2	0x1015	ObjectPriorizationCriterion	UINT16	Valid: < 2 Used to reduce transmitted objects via CAN. Decision which objects are discarded is based on this criterion. 0: Radial 1: Look ahead
2	0x1016	CAN object data options	16 bit field	Valid: all bit 0: 0 = absolute velocities, 1 = relative velocities bit 1: 0 = boxes are object boxes, 1 = boxes are bounding boxes bits 2…15: reserved
2	0x1017	Minimum Object Age	UINT16	Valid: all Minimum tracking age (number of scans) of an object to be transmitted.
2	0x1018	Maximum Prediction Age	UINT16	Valid: all Maximum prediction age (number of scans) of an object to be transmitted.
2	0x1019	Interface Flags	UINT16	RS232 Baud rate: 0 = default (currently 57600 Baud) 1 = 2400 Baud 2 = 4800 Baud 3 = 9600 Baud 4 = 19200 Baud 5 = 38400 Baud 6 = 57600 Baud 7 = 115200 Baud 8 = 921600 Baud 9 = 6250000 Baud >9 = default
2	0x1100	Start angle	INT16	In 1/32°, in the sensor coordinate system. Valid: 1600…-1919. Start angle > end angle!
2	0x1101	End angle	INT16	In 1/32°, in the sensor coordinate system. Valid: 1599…-1920. Start angle > end angle!

Bytes	Parameter index	LUX parameter: content	Content type	Description
2	0x1102	Scan frequency	UINT16	In 1/256 Hz. Valid: 3200 (12.5 Hz) 6400 (25.0 Hz) 12800 (50.0 Hz)
2	0x1103	Sync angle offset	INT14 (!) 16 bits used	In 1/32° in the sensor coordinate system. Valid: -5760…+5759 (-180°…+180°). Bits 14 and 15 are ignored!
2	0x1104	angular resolution type	UINT16	0: focused 1: constant 2: reserved
2	0x1105	angleTicksPerRotation	UINT16	11520 (read only), constant for ibeo LUX
2	0x1200	SensorMounting_X	INT16	In cm, related to vehicle reference point, rear axle. Order of translation and rotation is essential (Rotation -> Translation).
2	0x1201	SensorMounting_Y	INT16	In cm, related to vehicle reference point, rear axle. Order of translation and rotation is essential (Rotation -> Translation).
2	0x1202	SensorMounting_Z	INT16	In cm, related to vehicle reference point, rear axle. Order of translation and rotation is essential (Rotation -> Translation).
2	0x1203	SensorMounting_Yaw	INT16	In 1/32°, order of translation and rotation is essential (Yaw->Pitch->Roll-> Translation).
2	0x1204	SensorMounting_Pitch	INT16	In 1/32°, order of translation and rotation is essential (Yaw->Pitch->Roll-> Translation).
2	0x1205	SensorMounting_Roll	INT16	In 1/32°, order of translation and rotation is essential (Yaw->Pitch->Roll-> Translation).
2	0x1206	VehicleFrontToFrontAxle	UINT16	valid: all; in cm
2	0x1207	FrontAxleToRearAxle	UINT16	valid: all; in cm
2	0x1208	RearAxleToVehicleRear	UINT16	valid: all; in cm
2	0x1209	VehicleWidth	UINT16	valid: all; in cm

Bytes	Parameter index	LUX parameter: content	Content type	Description
2	0x120A	steerRatioType	UINT16	0: TBD 1: TBD
4	0x120C	SteerRatioPoly0	Float32	valid: all
4	0x120D	SteerRatioPoly1	Float32	valid: all
4	0x120E	SteerRatioPoly2	Float32	valid: all
4	0x120F	SteerRatioPoly3	Float32	valid: all
2	0x1210	Vehicle Motion Data Flags	16 bit field	Bit 0: Vehicle Motion data expected: 1=true, 0=false Bits 1 to 15: reserved

4.3 Example

This example shows how to set the IP address via CAN to 10.152.36.200.

Bytes	Message ID: CAN base ID + 0xA e.g. 0x50A	Content type	Content
0	Command ID: 0x0010	UINT16	0x10
1			0x00
2	Parameter index: 0x1000 (IP Address)	UINT16	0x00
3			0x10
4	Parameter data (here: 10.152.36.200, resp. 0x0A9824C8)	UINT64	0xC8
5			0x24
6			0x98
7			0x0A

5 Time sync message

This message is for time synchronization of the Ibeo LUX laser scanner.
The time is sent to the sensor in a 64 bit NTP time format.
The CAN ID is 0x100.
Attention: The data is encoded in big endian byte order in this message!

Content	Data area	Data type	Description
Seconds	bytes 0…3	unsigned int 32 bit	Send in big endian format
Fractional Seconds	bytes 4…7	Unsigned int 32 bit	

The sensor only respects receiving time sync messages if the time sync mode bits in the parameter Customer processing switch 0 are configured for receiving time sync messages over CAN.

6 Ibeo LUX error/warning

As soon as an ibeo LUX laserscanner detects an error or wants to emit a warning, this message is sent. Errors and warning bits are reset after sending this message.
This message will be sent periodically as long as errors of warnings persist.
The ID is CAN base ID + 0xF.
For a base ID 0x500 the command ID is 0x50F.
Attention: The data is encoded in little endian byte order in this message!

Content	Data area	Data type	Description
Error 1	bytes 0…1	bit field 16 bits	See detailed description below.
Error 2	bytes 2…3	bit field 16 bits	
Warning 1	bytes 4…5	bit field 16 bits	
Warning 2	bytes 6…7	bit field 16 bits	

6.1 Error register 1

Bytes	LUX error	Description	Comment
Bit 0	E-SP	internal error	contact support
Bit 1	E-Motor_1	motor fault	contact support
Bit 2	E-Buffer_1	scan buffer transmitted incompletely	decrease scan resolution/frequency/range; contact support
Bit 3	E-Buffer_2	Scan buffer overflow	decrease scan resolution/frequency/range; contact support
Bit 4	E-Meas_1	APD voltage failed	contact support
Bit 5		reserved	
Bit 6		reserved	
Bit 7		reserved	
Bit8…9	E-Temp	Bit 9: APD Over Temperature Bit 8: APD Under Temperature Bit 8 and 9: APD Temperature Sensor defect	provide cooling provide heating contact support
Bit 10	E-Motor_2	motor fault	contact support
Bit 11	E-Motor_3	motor fault	contact support
Bit 12	E-Motor_4	motor fault	contact support
Bit 13	E-Motor_5	motor fault	contact support
Bit14…15		reserved	

6.2 Error register 2

Bytes	LUX error	Description	Comment
Bit 0	E-IF_internal_1	no scan data received.	contact support
Bit 1	E-IF_internal_2	internal communication error	contact support
Bit 2	E-IF_internal_3	incorrect scan data	contact support
Bit 3	E-Configuration_1	FPGA not configurable	contact support
Bit 4	E-Configuartion_2	incorrect configuration data	load correct configuration values
Bit 5	E-Configuration_3	configuration contains incorrect parameters	load correct configuration values
Bit 6	E-Timeout_1	data processing timeout	decrease scan resolution or scan frequency
Bit 7	E-Timeout_2	reset the computation of the environmental model	contact support
Bit8…15	reserved		

6.3 Warning register 1

Bytes	LUX warning	Description	Comment
Bit0	W-CMD	internal communication error	
Bit1	W-Range_1	internal warning	
Bit2	W-Range_2	internal warning	
Bit3	W-low_temperature	temperature too low	warning of insufficient temperature
Bit4	W-high_temperature	temperature too high	warning of exceeding temperature
Bit5	W-Motor_1	internal warning	
Bit6	W-Motor_2	internal warning	
Bit 7	W-Sync	synchronization error	check synchronization- and scan frequency
Bit7…15	RES 7…15	reserved	

6.4 Warning register 2

Bytes	LUX warning	Description	Comment
Bit0	W-IF_CAN	CAN Interface blocked	check CAN bus and CAN connection
Bit1	E-IF_ETH	Ethernet Interface blocked	check Ethernet connection
Bit2	W-CANdata	incorrect CAN message received	check CAN data
Bit3	W-IF_internal_1	incorrect scan data	contact support
Bit4	W-ETHdata	unknown or incomplete data	check Ethernet data
Bit5	W-Command	incorrect or forbidden command received	check command
Bit6	W-Flash	memory access failure	restart ibeo LUX, contact support
Bit7	W-Overflow_1	internal overflow	contact support
Bit8	W-EgoMotion	vehicle data update missing	check CAN vehicle data
Bit9	W-Mounting Position	incorrect mounting parameters	correct mounting position according to OM
Bit10	W-CalcFrequency	no object computation due to scan frequency.	set the scan frequency to 12.5 Hz to receive objects
Bit11…15	reserved		

7 Ego motion information

Algorithms working in ibeo LUX and Ibeo applications are designed to use ego motion information if the sensor is mounted e.g. on a vehicle.
Generally it is not mandatory but data quality will decrease if sensors are moving and there is no ego motion information available via CAN. Data update rate must be at least the scan frequency of the ibeo LUX sensors. Better is twice or more the sensors frequency.
Applications based on the Ibeo API like AppBase have a configurable CAN data parser which is capable of decoding almost all kinds of messages containing ego motion data.
The ibeo LUX also has a configurable CAN data parser.