

Ibeo CAN data protocol description

Release date of this document: 19.09.2008.

1 Introduction.....	1
2 General information	2
3 Object Data.....	3
3.1 List header: CAN Base ID (e.g. 0x500).....	3
3.2 Time stamp: CAN Base ID + 0x1 (e.g. 0x501)	3
3.3 Tracking1: CAN Base ID + 0x2 (e.g. 0x502)	4
3.4 Tracking2: CAN Base ID + 0x3 (e.g. 0x503)	4
3.5 Class and box1: CAN Base ID + 0x4 (e.g. 0x504)	6
3.6 Box2: CAN Base ID + 0x5 (e.g. 0x505).....	6
3.7 Contour header: CAN Base ID + 0x6 (e.g. 0x506).....	8
3.8 Contour points: CAN Base ID + 0x7 (e.g. 0x507).....	8
4 Command.....	10
5 Reply	10
6 Ibeo LUX error/warning	11

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).

3 Object Data

Object data can be transmitted via CAN.

The ID range is CAN base ID ... CAN base ID + 0x7, first and last ID included.

For a base ID 0x500 the range is [0x500, 0x507].

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 1.
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
Reserved	bytes 5...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 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:

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Object ID							
1	Position X							
2	Position X							
3	Position Y							
4	Position Y							
5	Velocity X							
6	Velocity X							
7	Velocity Y							

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 x	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 $(\text{NumOfContourPoints}+1) \text{ Div } 3$. 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.
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).
Reserved	byte 3	-	-
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. Divide 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	

4 Command

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.

5 Reply

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.

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 or warnings persist.

The ID is CAN base ID + 0xF.

For a base ID 0x500 the command ID is 0x50F.

Content	Data area	Data type	Description
Error 1	bytes 0...1	bit field 16 bits	Please refer to the document 'ibeo LUX errors and warnings'.
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	

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 AppBase2 have a configurable CAN data parser which is capable of decoding almost all kinds of messages containing ego motion data. However, ibeo LUX sensors need ego motion data transmitted in predefined CAN messages:

0x303	Data area	Data type	Description
Version	byte 0	UINT8	Set to 2.
Velocity	bytes 1...2	INT16	Velocity in 0.01 m/s. Two's complement. Forward: positive values.

0x304	Data area	Data type	Description
Version	byte 0	UINT8	Set to 2.
Cross acceleration	bytes 1...2	INT16	Cross acceleration in 0.001 m/s ² . Two's complement. To the left: positive values.

0x305	Data area	Data type	Description
Version	byte 0	UINT8	Set to 2.
Steering wheel angle	bytes 1...2	INT16	Steering wheel angle in 0.001 rad. Two's complement. To the left: positive values.

0x306	Data area	Data type	Description
Version	byte 0	UINT8	Set to 2.
Yaw rate	bytes 1...2	INT16	Yaw rate in 0.0001 rad/s. Two's complement. To the left: positive values.